



Michigan Technological University
Create the Future Digital Commons @ Michigan Tech

Dissertations, Master's Theses and Master's
Reports - Open

Dissertations, Master's Theses and Master's
Reports

2009

Long-Range Pollution Transport: Trans-Atlantic Mechanisms and Lagrangian Modeling Methods

Robert Christopher Owen
Michigan Technological University

Follow this and additional works at: <https://digitalcommons.mtu.edu/etds>


 Part of the [Environmental Engineering Commons](#)

Copyright 2009 Robert Christopher Owen

Recommended Citation

Owen, Robert Christopher, "Long-Range Pollution Transport: Trans-Atlantic Mechanisms and Lagrangian Modeling Methods", Dissertation, Michigan Technological University, 2009.
<https://doi.org/10.37099/mtu.dc.etds/819>

Follow this and additional works at: <https://digitalcommons.mtu.edu/etds>

 Part of the [Environmental Engineering Commons](#)

**Long-range pollution transport: Trans-Atlantic
mechanisms and Lagrangian modeling methods**

By

Robert Christopher Owen

A DISSERTATION

Submitted in partial fulfillment of the requirements

for the degree of

DOCTOR OF PHILOSOPHY

(Environmental Engineering)

MICHIGAN TECHNOLOGICAL UNIVERSITY

2009

Copyright © 2009 Robert Christopher Owen
All rights reserved.

This dissertation, “Long-range pollution transport: Trans-Atlantic mechanisms and Lagrangian modeling methods,” is hereby approved in partial fulfillment of the requirements for the degree of DOCTOR OF PHILOSOPHY (Environmental Engineering).

Geological and Mining Engineering and Sciences

Dr. Richard E. Honrath, Dissertation Adviser

Dr. Judith Perlinger, Non-Departmental Program Chair

Date

This work is dedicated to my father, Bob Owen, for igniting the spark that inspired me to begin, and to the memory of my adviser, Richard Honrath, who was taken from us much too early, but fueled the flame that helped me to finish.

Abstract

Over the past several decades, it has become apparent that anthropogenic activities have resulted in the large-scale enhancement of the levels of many trace gases throughout the troposphere. More recently, attention has been given to the transport pathway taken by these emissions as they are dispersed throughout the atmosphere. The transport pathway determines the physical characteristics of emissions plumes and therefore plays an important role in the chemical transformations that can occur downwind of source regions. For example, the production of ozone (O_3) is strongly dependent upon the transport its precursors undergo. O_3 can initially be formed within air masses while still over polluted source regions. These polluted air masses can experience continued O_3 production or O_3 destruction downwind, depending on the air mass's chemical and transport characteristics. At present, however, there are a number of uncertainties in the relationships between transport and O_3 production in the North Atlantic lower free troposphere.

The first phase of the study presented here used measurements made at the Pico Mountain observatory and model simulations to determine transport pathways for US emissions to the observatory. The Pico Mountain observatory was established in the summer of 2001 in order to address the need to understand the relationships between transport and O_3 production. Measurements from the observatory were analyzed in conjunction with model

simulations from the Lagrangian particle dispersion model (LPDM), FLEX-PART, in order to determine the transport pathway for events observed at the Pico Mountain observatory during July 2003. A total of 16 events were observed, 4 of which were analyzed in detail. The transport time for these 16 events varied from 4.5 to 7 days, while the transport altitudes over the ocean ranged from 2-8 km, but were typically less than 3 km. In three of the case studies, eastward advection and transport in a weak warm conveyor belt (WCB) airflow was responsible for the export of North American emissions into the FT, while transport in the FT was governed by easterly winds driven by the Azores/Bermuda High (ABH) and transient northerly lows. In the fourth case study, North American emissions were lofted to 6-8 km in a WCB before being entrained in the same cyclone's dry airstream and transported down to the observatory. The results of this study show that the lower marine FT may provide an important transport environment where O_3 production may continue, in contrast to transport in the marine boundary layer, where O_3 destruction is believed to dominate.

The second phase of the study presented here focused on improving the analysis methods that are available with LPDMs. While LPDMs are popular and useful for the analysis of atmospheric trace gas measurements, identifying the transport pathway of emissions from their source to a receptor (the Pico Mountain observatory in our case) using the standard gridded model output, particularly during complex meteorological scenarios can be difficult can be difficult or impossible. The transport study in phase 1 was limited to only 1 month out of more than 3 years of available data and included only 4 case studies out of the 16 events specifically due to this confounding factor. The second phase of this study addressed this difficulty by presenting a method to clearly and easily identify the pathway taken by only those emissions that arrive at a receptor at a particular time, by combining the standard gridded

output from forward (i.e., concentrations) and backward (i.e., residence time) LPDM simulations, greatly simplifying similar analyses. The ability of the method to successfully determine the source-to-receptor pathway, restoring this Lagrangian information that is lost when the data are gridded, is proven by comparing the pathway determined from this method with the particle trajectories from both the forward and backward models. A sample analysis is also presented, demonstrating that this method is more accurate and easier to use than existing methods using standard LPDM products. Finally, we discuss potential future work that would be possible by combining the backward LPDM simulation with gridded data from other sources (e.g., chemical transport models) to obtain a Lagrangian sampling of the air that will eventually arrive at a receptor.

Acknowledgements

There are a number of individuals who I would like to acknowledge for their support of this work. I would like to my adviser, Dr. Richard Honrath, for his patience, guidance, flexibility, and support throughout the course of this research. I would also like to thank my committee, Drs. Judith Perlinger, Will Cantrell, and Alex Mayer, for their commitment and guidance. I thank my research group, Dr. María Val Martín, Kateryna Lapina, and Dr. Jan Kleissl for their contributions to my research and allowing me to be a part of their research as well. I would like to extend my thanks to Dr. Andreas Stohl (Norwegian Institute for Air Research) for providing the FLEXPART model and his extensive technical and analytical contributions to my work and to Dr. Owen Cooper (Cooperative Institute for Research in Environmental Sciences) for his guidance in the meteorological interpretations presented in Chapter 2 of this work. I would like to thank Mike Dziobak for his efforts in installing and maintaining the Pico Mountain observatory, Dr. Paulo Fialho (Azores University, Portugal) for his support in establishing and maintaining the Pico Mountain observatory, and Dr. Detlev Helmig and David Tanner (Institute of Arctic and Alpine Research) for their contributions to the observatory and for allowing me to be a part of their non-methane hydrocarbon research. Additionally, I would like to gratefully acknowledge Sabine Eckhardt (Norwegian Institute for Air Research), Nicole Spichtinger, and Diamantino Henriques

(Portuguese Meteorological Institute) for providing ECMWF meteorological data used to drive FLEXPART and Paul Novelli and Sam Oltmans (NOAA Climate Monitoring and Diagnostics Laboratory) for the referencing of our measurements to CMDL and NIST standards.

I would like to thank my wonderful wife Amy for her patience, understanding, and support during the course of my research. I would like to thank my parents, Bob and Jane Owen, for their loving support during the course of this research, but more so for providing a fulfilling childhood and for their roles in helping make me who I am today. I wish to thank my many friends who have helped make my time at Michigan Tech enjoyable and contributed to my physical and spiritual well being.

This work was supported by NOAA, Office of Global Programs, grants NA16GP1658, NA86GP0325 and NA03OAR4310002, the National Science Foundation, grants INT-0110397, ATM-0535486 and ATM-0720955, and the US Department of Agriculture, Azores Cooperative Initiative Program grant 58-3625-5-127.

Contents

Abstract	iv
Acknowledgments	vii
List of Figures	xv
List of Tables	xviii
1 Introduction	1
1.1 Background	1
1.2 Research approach and objectives	12
1.3 Dissertation overview	15
2 Summertime transport of anthropogenic emissions to the central North Atlantic[†]	16
2.1 Introduction	16
2.2 Methods	19
2.2.1 PICO-NARE station measurements	19
2.2.2 FLEXPART	21
2.2.3 Event selection	23
2.3 Results	23

2.3.1	Event 1: Direct transport from North America associated with the decaying tropical storm Bill	28
2.3.1.1	Export out of the U.S. boundary layer in a low pressure system	29
2.3.1.2	Transport to the station within geostrophic winds	32
2.3.2	Event 2: Lofting and subsidence associated with a mid-latitude cyclone	34
2.3.2.1	Export out of the U.S. boundary layer in the westerly wind	34
2.3.2.2	Rapid lofting and subsidence	37
2.3.3	Events 3 and 4: Export in a weak warm conveyor belt followed by transport around the Azores High	39
2.3.3.1	Export out of the U.S. boundary layer in a weak warm conveyor belt	42
2.3.3.2	Transport to the station in gradient winds	43
2.4	Discussion	44
2.5	Summary and Conclusions	48

3 Lagrangian tracking of pollution plumes using gridded model output[†] 50

3.1	Introduction	50
3.2	Method overview	54
3.2.1	Formulation of the model output and folded retroplume	54
3.2.1.1	Standard output from the forward mode	54
3.2.1.2	Standard output from the backward mode	55

3.2.1.3	The folded retroplume – combining model output to determine the source-to-receptor transport pathway	56
3.2.2	Illustrative case	59
3.3	Method evaluation	61
3.3.1	Approach and methods for the detailed evaluation . . .	61
3.3.2	Detailed evaluation results	64
3.3.2.1	Detailed evaluation of the folded retroplume pathway	64
3.3.2.2	Detailed evaluation of the UMRs	67
3.3.3	The impact of various model settings on the folded retroplume	73
3.3.3.1	Folded retroplume pathway	73
3.3.3.2	Folded retroplume UMRs	75
3.3.3.3	Spatial grid sizes	75
3.3.3.4	Averaged or instantaneous values and the length of averaging period	76
3.3.4	Summary of evaluation	77
3.4	Sample application	78
3.4.1	Comparison of the folded retroplume pathway with standard LPDM products	79
3.4.2	Folded retroplume UMRs	82
3.5	Alternate backward LPDM combinations	89
3.5.1	Chemical transport models	89
3.5.2	Meteorological fields, measurement and satellite data, and other applications	90
3.5.3	Wet and dry removal	91
3.6	Practical considerations	92

3.6.1	FLEXPART output time stamp	92
3.6.2	Output unit options	92
3.7	Summary and conclusions	93
4	Summary and conclusions	97
4.1	Mechanisms for transport of North American pollution to the central North Atlantic lower free troposphere	97
4.2	New methods for tracking plumes from their source to a recep- tor using gridded model output	99
4.3	Application and future analysis	101
	References	117
A	A users guide for FLEXPART as implemented at MTU	127
A.1	Introduction	127
A.2	Running FLEXPART at MTU	128
A.2.1	Forward simulations	129
A.2.1.1	Part 1: Running FLEXPART	129
A.2.1.2	Part 2: Initial ASCII output to intermediate files	131
A.2.1.3	Part 3: Intermediate to final files	132
A.2.2	Backward simulations	133
A.2.3	Meteorological data	134
A.2.4	GRIB decoder	134
A.2.5	Computational considerations	135
A.3	Custom modifications to the FLEXPART Fortran code	136
A.3.1	FLEXPART version 6.2 modified files	137
A.3.1.1	FLEXPART bug fixes	137
A.3.1.2	FLEXPART modifications for folding	137

A.3.1.3	FLEXPART modifications for easier processing at MTU	138
A.3.2	FLEXPART version 8.0 modified files	139
A.3.2.1	FLEXPART bug fixes	139
A.4	IDL programs for processing and plotting FLEXPART	140
A.4.1	Programs for loading data	140
A.4.2	Programs for processing data	140
A.4.3	Programs for plotting data	141
A.5	FLEXPART settings for standard MTU products	142
A.5.1	Retroplumes	142
A.5.2	Retroplume exceptions	143
A.5.3	Forward runs	144
A.5.4	Recommended settings	145

B IDL programs for running, processing, and plotting FLEXPART and products 149

B.1	Plotting programs	149
B.1.1	color_key_for_dist.pro	149
B.1.2	make_std_4_panel_retro_plot.pro	150
B.1.3	oplot_density_field2.pro	155
B.1.4	oplot_density_field.pro	162
B.1.5	oplot_distf.pro	168
B.2	Programs for loading and manipulating FLEXPART output	176
B.2.1	get_avg_flex_height.pro	176
B.2.2	get_flex_density.pro	181
B.2.3	get_flex_height_density.pro	185
B.2.4	get_retro_folded_conc.pro	188
B.2.5	load_flex_gridded_data.pro	199

B.2.6	load_flex_pico_ppb_data.pro	203
B.2.7	load_flex_retro_data.pro	208
B.3	Programs for running and processing forward FLEXPART simulations	212
B.3.1	convert_forward_temps_to_gridded_final.pro	212
B.3.2	read_forward_flex_output.pro	221
B.3.3	run_forward_flexpart.pro	229
B.4	Programs for running and processing backward FLEXPART simulations	245
B.4.1	convert_retro_temps_to_gridded_final.pro	245
B.4.2	doit_dry_auto_retro.pro	249
B.4.3	doit_wet_auto_retro.pro	252
B.4.4	read_retro_flex_output.pro	255
B.4.5	run_retro_flex.pro	260
B.4.6	write_available.pro	275
C	Copyright permissions and information	277
C.1	Documentation for Chapter 2	277
C.1.1	Email requesting for reproduction permission	277
C.1.2	Email granting reproduction permission	279
C.2	Documentation for Chapter 3	281
C.2.1	ACP copyright policy	281

List of Figures

1.1	Map of the location of the Pico Mountain observatory	14
1.2	Map of US anthropogenic CO emissions	15
2.1	Time series of the CO measured at the PICO-NARE station and CO from FLEXPART retroplumes	24
2.2	FLEXPART retroplume results for event 1	30
2.3	Snapshots of the retroplume for Event 1, plotted on top of infrared satellite images	31
2.4	FLEXPART retroplume results for event 2	35
2.5	Snapshots of the retroplume for Event 2, plotted on top of infrared satellite images	36
2.6	FLEXPART retroplume results for events 3 and 4	40
2.7	Snapshots of the retroplumes for Event 3 and 4, plotted on top of infrared satellite images	41
3.1	Illustrative snapshots of the forward, sensitivity, and folded retroplumes	62
3.2	Time integrated forward CO and backward sensitivity plumes for the folded retroplume evaluation	68
3.3	Comparative views of the positive particle trajectories and the folded retroplume	69

3.4	Upwind mixing ratios (<i>UMRs</i>) for the folded retroplume evaluation	72
3.5	Comparative views of the standard sensitivity plume and the folded retroplume for the sample application	83
3.6	Comparative snapshots of the standard sensitivity plume and the folded retroplume with forward CO contours for the sample application	84
3.7	The <i>UMRs</i> at each upwind time for the sample evaluation	88
4.1	Time series of CO and O ₃ from Pico and FLEXPART CO enhancements, tracer age, and transport times	103
4.2	Correlations between the observed O ₃ /CO slopes and the average of the modeled forward and backward FLEXPART CO enhancements for all events	106
4.3	Correlations between the observed O ₃ /CO slopes and the average of the modeled forward and backward FLEXPART CO enhancements for only those events that were sufficiently captured by FLEXPART	107
4.4	Correlations between the observed O ₃ /CO slopes and the average trans-Atlantic transport height of the folded retroplume for all event	109
4.5	Correlations between the observed O ₃ /CO slopes and the average trans-Atlantic transport height of the folded retroplume for only those events that were sufficiently captured by FLEXPART	110
4.6	The vertical view of folded retroplumes initiated during event 3111	
4.7	The horizontal view of folded retroplumes initiated during event 3	112

4.8	The vertical view of folded retroplumes initiated during select events	113
4.9	The horizontal view of folded retroplumes initiated during select events	114

List of Tables

2.1	Events of elevated CO observations during July 2003	27
3.1	Model settings used for evaluation	74

Chapter 1

Introduction

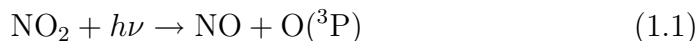
1.1 Background

Anthropogenic emissions have a significant impact on the composition of the atmosphere. Many of these emissions, such as CO_2 and methane, are relatively inert, having atmospheric lifetimes in excess of 10 years. As a result, these emissions will become well mixed into the troposphere before they are removed. Other emissions, such as NO , NO_2 , and volatile organic carbons (VOCs), are relatively reactive, with lifetimes of hours to days. These short-lived compounds can initiate and participate in chemical reactions that affect their concentrations and the concentrations of other trace gases. The transport during the first few hours and days after emission of plumes containing short-lived species determines the physical conditions experienced by these plumes and thus plays a major role in the chemical reactions that occur.

Tropospheric O_3 is one chemical species whose concentration is highly dependent upon the initial transport experienced by pollution plumes, not only because it is a short-lived compound, but also because it is a secondary pollutant and its production is dependent on chemical reactions between other

short lived species. Tropospheric O_3 plays several important roles in the chemical and physical properties in the atmosphere. It is a strong oxidant which can negatively affect plant and animal life at levels not far above current background concentrations (e.g., *Fowler et al.* [1999]; *Mauzerall and Wang* [2001]; *Karnosky et al.* [2005]). Ozone is the primary source of the hydroxyl radical in the atmosphere, which is the primary oxidant in the atmosphere [Levy, 1971] and thus O_3 plays a crucial role in the atmosphere's ability to clean itself of other pollutants. Finally, O_3 is a strong greenhouse gas, accounting for approximately 14% of the total global warming *IPCC* [2007]. The background levels of tropospheric O_3 have increased significantly since the industrial age [Mickley et al., 2004] and have continued to increase over the last two decades [Parrish et al., 2004; Lelieveld et al., 2004; Lin et al., 2000]. These increases in background O_3 concentrations can make it difficult for subsequent downwind regions to attain O_3 standards. For example, it has been estimated that 20% of the European O_3 violations in 1997 would not have occurred without the emissions from North America [Li et al., 2002]. Episodes of long-range transport of concentrated pollution plumes can also occur, resulting in significantly elevated levels of O_3 in the absence of local sources (e.g., [Trickl et al., 2003; Guerova et al., 2006]. The importance of O_3 and its ability to be transported long distances indicates the need for a thorough understanding of its production and transport.

The specific circumstances that result in elevated O_3 concentrations are controlled by a cycle of production and destruction. Ozone is produced primarily by the photolysis of NO_2 to NO and $O(^3P)$, followed by the combination of $O(^3P)$ and diatomic oxygen:

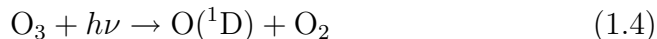


where M represents any molecule that absorbs the excess energy from the reaction (usually N₂ or O₂). Ozone can be lost by reacting with NO to reform NO₂:



Reactions 1.2 and 1.3 are quite fast, so O₃ production and loss through these reactions quickly reaches a steady state, where the destruction and production of O₃ are equal. In the absence of other reactions to convert NO to NO₂, reactions 1.1, 1.2, and 1.3 form a null cycle that results in neither O₃ destruction or production.

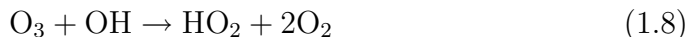
In addition to reaction 1.3, O₃ can be lost by photolysis followed by reaction with water:



Note that this reaction is the primary loss mechanism for O₃ and is also the primary source for the hydroxyl radical in the atmosphere [Levy, 1971]. O(^1D) can also be deactivated to participate in reaction 1.2 to reform O₃:



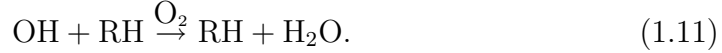
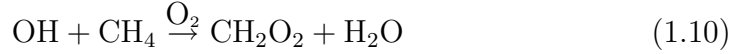
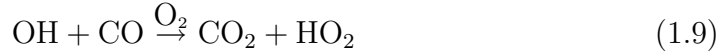
Additional O₃ loss reactions include:



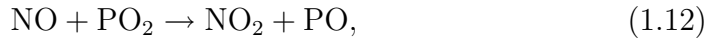
As a result of the reactions above, tropospheric O₃ has a average lifetime of about 22 days during summer in the mid-latitudes [Stevenson *et al.*, 2006].

The OH formed in reactions 1.5 and 1.7 can react with CO, methane (CH₄), and VOCs (RH in the following equations) to form peroxy radicals

($\text{PO}_2 = \text{HO}_2 + \text{CH}_3\text{O}_2 + \text{RO}_2$) *Monks* [2005]:



These peroxy radicals can react with NO, giving, in addition to reaction 1.3, a pathway for the conversion of NO to NO_2 . These reactions take the general form:



Reaction 1.12 allows for O_3 production because it provides a way to convert NO back to NO_2 , which would otherwise be accomplished by O_3 destruction in the cycle of production and destruction described by reactions 1.1, 1.2, and 1.3.

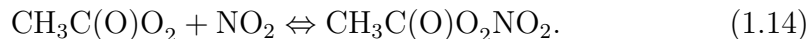
While each VOC molecule (and its subsequent oxidized forms) can only participate in reactions 1.9 and 1.12 a limited number of times (until ultimately oxidized to CO_2), each NO and NO_2 acts as a catalyst in the presence of VOCs for a net production of O_3 . As a result, relatively low levels of NO_x ($\text{NO} + \text{NO}_2$) can produce significant amounts of O_3 [*Carpenter et al.*, 1997; *Reeves et al.*, 2002]. The rate of O_3 production, however, has a non-linear dependence on the concentration of NO_x . In fact, above a threshold concentration of NO_x and with sufficient VOCs, the ozone production efficiency decreases as the NO concentration increases [*Liu et al.*, 1987]. Consequently, the ozone production efficiency can be higher in downwind regions than in source regions, where the levels of NO_x are lower than in source regions [*Jacob et al.*, 1993]. Ultimately, the rate of O_3 production and destruction is controlled by a number of factors, including VOC/ NO_x ratios, the reactivity

of the available VOCs, and the rates of meteorological dispersion, though the concentrations of NO_x are frequently a controlling factor [*Sillman*, 1999].

The sources and sinks of NO_x in the atmosphere are important because of the catalytic role NO_x plays in O_3 production. During the summertime, NO_x has a lifetime of about 2 days in the mid-latitudes before being removed, mainly by wet removal after being oxidized by OH:



HNO_3 can then be rapidly removed by uptake on water droplets or aerosols followed by washout in precipitation or by dry deposition on surfaces. If HNO_3 is not removed, however, it has a lifetime of about 18 days in mid-latitudes during the summertime before it is photolyzed or oxidized by OH to reform NO_x [*Neuman et al.*, 2006]. Alternately, NO_x can form peroxyacetyl nitrate (PAN):



PAN has a lifetime that is highly temperature dependent, it will quickly decomposing back to NO_2 and $\text{CH}_3\text{C}(\text{O})\text{O}_2$ in warmer temperatures. However, under colder conditions, PAN can be transported long distances downwind, potentially reforming NO_2 downwind, which can then contribute to further O_3 formation. As a result of the dependence of NO_x chemistry on temperature, the presence of precipitation, or contact with surfaces, the availability of NO_x to produce O_3 is highly dependent upon the transport, which control these physical characteristics of the pollution plume.

This review of O_3 chemistry shows that O_3 production is dependent upon the availability of NO_x and that the availability of NO_x is dependent on the physical settings of pollution plumes dictated by the transport. As a result of its importance for human health and the environment, the links

between transport and O_3 production have thus for several decades, been the focus of many studies. Early studies focused more on regional production and transport (e.g., *Fishman et al.* [1985]; *McKeen et al.* [1991]). Continued research has shown that O_3 can be transported long distances and affect ambient O_3 levels in downwind regions (e.g., *Parrish et al.* [1993]; *Dickerson et al.* [1995]). It has since been shown that O_3 produced from anthropogenic precursors (NO_x and VOC) and the precursors themselves can be observed in remote, clean environments, thousands of km downwind of the precursor source regions (e.g., *Parrish et al.* [1998]; *Jaffe et al.* [1999]; *Honrath et al.* [2004]; *Lelieveld et al.* [2004]; *Val Martín et al.* [2008a]).

Since most anthropogenic emissions occur in the continental boundary layer (BL) at or near the surface, subsequent transport must either carry them further in the BL or into the free troposphere (FT). If these emissions are successfully exported out of the BL into the FT, where many removal processes are less effective and wind speeds are high, their lifetimes are significantly longer and are able to be transported long distances, though they may be less likely to impact surface concentrations downwind [*Stohl et al.*, 2002c]. Uplift within warm conveyor belts (WCBs), which are airstreams associated with cold fronts in mid-latitude cyclones, has been identified as an important mechanism for the rapid transport of emissions from the BL to the middle and upper FT (e.g., *Stohl and Trickl* [2001a]; *Cooper et al.* [2004]). Emissions exported through WCBs to the FT have been reported to impact the middle and upper FT downwind of source regions, resulting in enhanced levels of O_3 , CO, and nitrogen oxides [*Stohl and Trickl*, 1999; *Trickl et al.*, 2003; *Auvray and Bey*, 2005; *Huntrieser et al.*, 2005]. Rapid uplift of emissions to the FT can also occur through convection [*Dickerson et al.*, 1987], significantly affecting the FT above the source (e.g., *Pickering et al.* [1990]; *Thompson et al.* [1994]; *Jacob et al.* [1993]; *Li et al.* [2005]).

Export out of the continental BL can also occur at low altitudes, either as the result of episodic ventilation in the post cold front, another air stream associated with mid-latitude cyclones [*Cooper et al.*, 2002], or via horizontal advection in the geostrophic winds, not associated with frontal passages [*Li et al.*, 2002]. When transported to the marine boundary layer (MBL), the lifetime of these emissions are relatively short and they are less likely to impact downwind regions than are those exported to the free troposphere. Even when above the BL, transport in the lower FT is slower than transport in the middle and upper FT. Consequently, the amount of mixing and the degree of photochemical processing is expected to be greater. As a result, the impact of emissions transported in the lower FT is not anticipated to be significant. Nonetheless, transport through the lower FT, but above the BL, appears to be important. *Li et al.* [2005] determined that more than one third of the CO exported from North America in July GOES-CHEM simulations occurred in the lowest 3 km. Using STOCHEM simulations, *Derwent et al.* [2008] found that all of the air parcels that carried North American emissions less than 20 days old arriving at Mace Head during 1998 remained at altitudes below 4 km during transport. *Neuman et al.* [2006] reported that polluted air containing relatively high levels of nitric acid (10s of ppb), can be transported hundreds of km downwind of source regions in layers as low as a few hundred meters altitude.

Measurements at the Pico Mountain observatory, located in the lower FT in the central North Atlantic, have also indicated that the transport in the lower FT can play an important role on the downwind concentrations of anthropogenic emissions. *Val Martín et al.* [2008b] reported that plumes originating from North America that were transported in the MBL had elevated levels of CO but little or no enhancements of O₃ or NO_x. However, North American pollution plumes transported in the lower FT, at altitudes from 2-4

km, had elevated levels of CO, O₃, and NO_x. *Honrath et al.* [2004] reported that North American pollution plumes sampled during the summer at the Pico Mountain observatory frequently exhibited elevated levels of CO and O₃, many of which were transported primarily in the lower FT. They found that the slope of the O₃/CO correlations for these plumes were significantly higher than had been reported before.

The O₃/CO relationship has been the subject of a number of studies. Measurements in the early and mid 1990's along the eastern US coast found typical slopes between 0.3–0.4 [*Parrish et al.*, 1994, 1998; *Chin et al.*, 1994]. More recent measurements in the US boundary layer made during the summer 2004 ICARTT campaign [*Fehsenfeld et al.*, 2006] produced slopes of 0.41 for a surface site at Chebogue Point and 0.47 for aircraft sampled plumes [*Hudman et al.*, 2008]. Significantly higher slopes, however, were found in the North Atlantic mid-troposphere (600-650 hPa). *Zhang et al.* [2006] reported a slope of 0.81 from July 2005 measurements from the TES satellite and a slope of 0.72 from ICARTT aircraft data. *Honrath et al.* [2004], however, reported the highest slopes, with an average slope around 1.0 for the summers of 2001 and 2003 at the Pico Mountain observatory, in the lower FT.

A number of explanations have been offered for these large variations in observed O₃/CO slopes. Decreases in CO emissions and increased O₃ production have been suggested as the cause of the higher O₃/CO slopes found in the 90's by *Parrish et al.* [1994, 1998] and *Chin et al.* [1994] and the more recent values reported by [*Hudman et al.*, 2008]. However, *Honrath et al.* [2004] determined that reduced CO emissions and CO oxidation could not explain the difference between the Pico slopes and the slopes from the 1990's. Instead, *Honrath et al.* [2004] suggested that longer transport times to the mid-Atlantic allowed for continued O₃ production, thus moderately aged anthropogenic plumes could have higher O₃/CO slopes. The higher slopes

found in the mid-troposphere over the Atlantic by *Zhang et al.* [2006] could support this argument. *Zhang et al.* [2006] used concentrations averaged over $10^\circ \times 10^\circ$ boxes, so the 0.7-0.8 could represent an average of a range of slopes in air masses of different characteristics. *Helmig et al.* [2008] found that the North Atlantic lower FT is dominated by O_3 destruction, so highly aged air could have a smaller O_3/CO slope. Similarly, lower slopes would be expected in fresher air masses near the coast that have not had time to produce much O_3 . While moderately aged polluted air masses that may have experienced transport that has not removed NO_x or O_3 would have higher slopes, like those reported by *Honrath et al.* [2004]. At present, however, the reason for the differences in the slopes found by these studies remains unclear and indicates the need for an improved understanding of the transport and production of O_3 .

Numerical model simulations are an important component in the interpretation of the transport associated with atmospheric measurements. The simplest modeling tool is the atmospheric trajectory, which is defined as the transport pathway of infinitesimally small particle of air. Trajectories can be traced either forward or backward in time. While forward trajectories show where an air parcel will go, backward trajectories show where an air parcel came from. The backward trajectory can therefore be used to establish relationships between atmospheric measurements and their sources [*Stohl*, 1998]. Trajectories have thus been part of almost every air measurement campaign [*Stohl et al.*, 2002b]. The uses of trajectories are widely varied and include wind sector analysis (e.g., *Moody et al.* [1995], residence time analysis (e.g., *Sirois and Bottenheim* [1995]), and Lagrangian box models (e.g., *Methven et al.* [2003]). Cluster analysis is a statistical method that groups trajectories with similar histories and has also been an important analysis approach using trajectories. Measurements are grouped according to the corresponding tra-

jjectory’s cluster and thereby classified according to the transport associated with each measurement (e.g., *Moy et al.* [1994]; *Moody et al.* [1995]; *Dorling and Davies* [1995]; *Kahl et al.* [1997]; *Cape et al.* [2000]; *Traub et al.* [2003]).

Trajectories remain popular because they are easy to use and interpret. However, since they only describe the transport of an infinitesimally small particles of air, they cannot adequately describe the deformation of an air mass. Lagrangian particle dispersion models (LPDMs) have been developed to overcome this deficiency of trajectories. LPDMs use many trajectories initiated over a source or receptor volume (for forward and backward simulations, respectively) in order to model the deformation of the air mass that occurs during transport. Each trajectory is usually allocated a mass and is thus referred to as a particle. While individual particle trajectories can be saved, the output from LPDMs are usually gridded. That is, concentrations (for forward runs) or residence times (for backward runs) are computed on a regular grid using the masses carried by the particles (see *Stohl et al.* [2005] and *Lin et al.* [2003] for full descriptions of two LPDMs).

LPDMs are increasing in popularity and have become common tools for the analysis of atmospheric measurements (e.g., *Stohl et al.* [2004a]). They have primarily been used in three ways. One is the usage of the backward LPDM to determine the transport history of an air mass associated with a measurement, including both the transport pathway and residence time analysis (e.g., *Stohl et al.* [2003]). Second, they are frequently used to determine source-receptor relationships by calculating the concentration of a tracer in a receptor (e.g., the concentration of CO at the Pico Mountain observatory from biomass burning in North American [*Lapina et al.*, 2008]). Finally, the forward mode of LPDMs are frequently used to understand the large scale transport of pollution or other tracers (i.e., *Stohl* [2001] and *Stohl et al.* [2002a]).

The output from LPDMs is significantly more complex than trajectories.

A single backward trajectory has only 3 parameters are each output time (the latitude, longitude, and altitude). In contrast, a single backward LPDM simulation will have a 3 dimensional grid of the sensitivity or residence time, and possibly matching grids of wet deposition, dry deposition, and uncertainty. Thus, in order to analyze the transport over 10-days, with model output every 6 hours and 10 output levels, a trajectory would require two maps, one of the horizontal and one of the vertical position at each time. For the LPDM, one would potentially need to analyze one residence time map for each output time and level; a total of 400 maps! As a result of this increased complexity, it is difficult to perform statistical techniques, such as cluster analysis, in order to analyze large sets of measurements. Even the analysis required for a few individual events can be difficult. For example, *Huntrieser et al.* [2005] provided detailed descriptions of transport for only two events.

Stohl et al. [2002b] recognized the need to simplify the LPDM output and introduced a method to summarize the results from backward model simulations. This method reduced the backward model simulation essential down to a few trajectories, whose longitude, latitude, and altitude were found by clustering the particles, using their location, at each upwind time. This method, however, only simplifies the backward output. It can be difficult to bring the information provided by the forward model simulations (i.e., the location of tracer or pollution) and the backward model simulations (i.e., the location of air that will eventually be transported to the receptor) together in order to adequately determine the source-to-receptor transport pathway (like that given by *Huntrieser et al.* [2005]). Presently, the only way to use both the forward and backward model output to determine the source-to-receptor transport pathway is to view maps of the tracer concentrations (from the forward mode) and residence time (from the backward mode) at many output levels and times. Thus, again assuming the LPDM output is saved

every 6 hours and at 10 vertical levels, in order to analyze the transport of an event observed in Europe that originated in the US 10 days earlier, one would potentially need to view 400 concentration and 400 residence time maps! Thus, there is still the need for additional methods to simplify and summarize the output from LPDMs.

1.2 Research approach and objectives

The Pico Mountain observatory was established in the summer of 2001 atop Pico Mountain, Azores (2225 masl, Figure 1.1) The observatory is located in the central North Atlantic, far downwind from any major emission regions, it is typically in the FT [Kleissl *et al.*, 2007], and is frequently impacted by emissions of anthropogenic origin [Honrath *et al.*, 2004; Val Martín *et al.*, 2008b,a]. The observatory operated year-round through August 2005, then during the summers of 2006-2008. Measurements of O₃ and CO have been nearly continuous since the establishment of the observatory. Nitrogen oxide measurements began in the summer of 2002 and non-methane hydrocarbon measurements began in the spring of 2004. Full documentation of the measurements are given by Honrath *et al.* [2004], Tanner *et al.* [2006], and Val Martín *et al.* [2008b].

In the previous section, it was shown that the measurements collected at the observatory, specifically the slope of the O₃/CO correlation, exhibited a behavior that was not well understood. This behavior clearly highlights the need for additional study of these measurements. Thus, as a step towards an improved understanding of the CO and O₃ behavior observed at the observatory, **the first objective of this research is to determine the meteorological mechanisms responsible for the export of North American anthropogenic emissions from the continental boundary layer and**

the subsequent transport to the Pico Mountain observatory.

In order to accomplish this objective, simulations with the LPDM FLEXPART have been made to support the the interpretation of these measurements. Forward simulations modeling the emission and transport of anthropogenic CO were made. Anthropogenic emissions are based on the EDGAR Fast Track 2001 emissions inventory [*Olivier et al.*, 1996, 1999], which has global coverage at a $1^\circ \times 1^\circ$ resolution. The US CO emissions are shown in Figure 1.2. These simulations are useful for understanding the overall transport situation, particularly when linking the transport of a plume with meteorological conditions. Additionally, backward simulations, initiated at the observatory, were made. These simulations are useful for determining source regions and transport pathways for air that arrives at the observatory.

It was also shown in the previous section that current modeling techniques with LPDMs are quite complex. As a result, it can be difficult to determine the source-to-receptor pathway for emissions for large data sets, like those collected at the Pico Mountain observatory. For example, it is difficult to determine what relationships exist between the observed O_3/CO slopes and the transport characteristics for more than a few events. Thus, **the second and third objectives of this research are to develop a new method to track pollution plumes from their source to a receptor using gridded model output and to evaluate the new tracking method in order to determine its accuracy and limitations.** In order to accomplish these objectives, a new analysis technique has been developed that couples the numerical output fields from both the forward and backward LPDM simulations. This technique was thoroughly tested and evaluated using the LPDM FLEXPART.

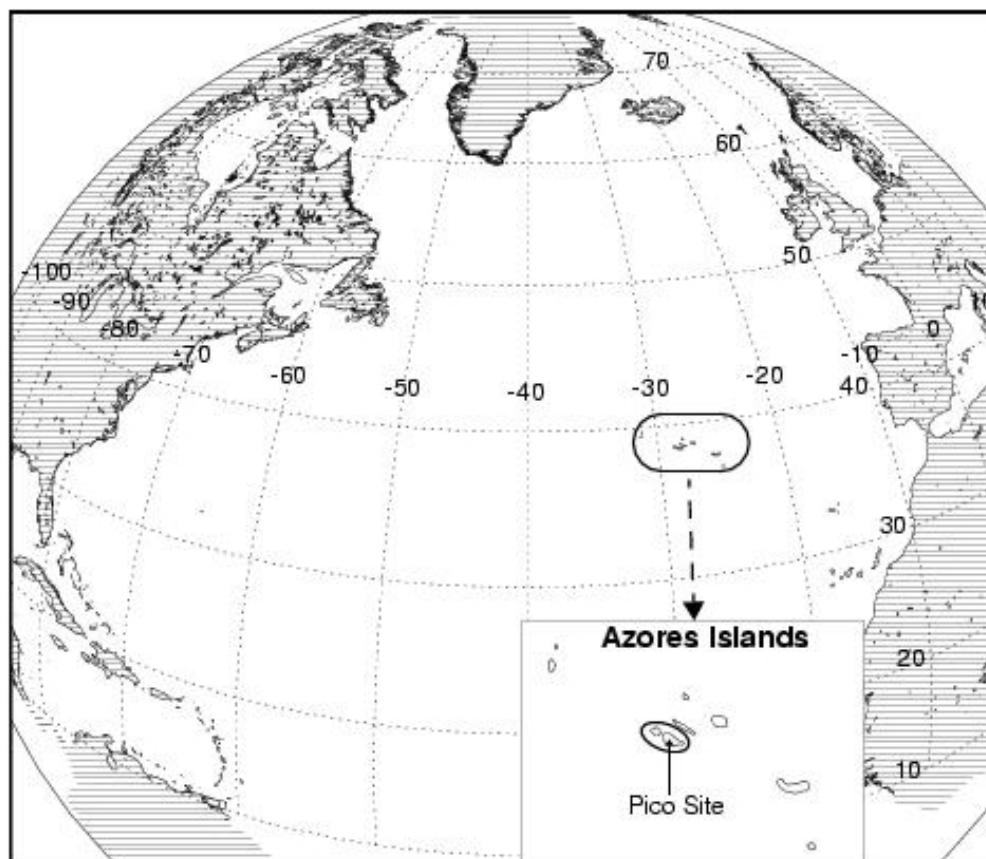


Figure 1.1 Location of the Pico Mountain observatory measurement site on Pico Island, Azores, Portugal.

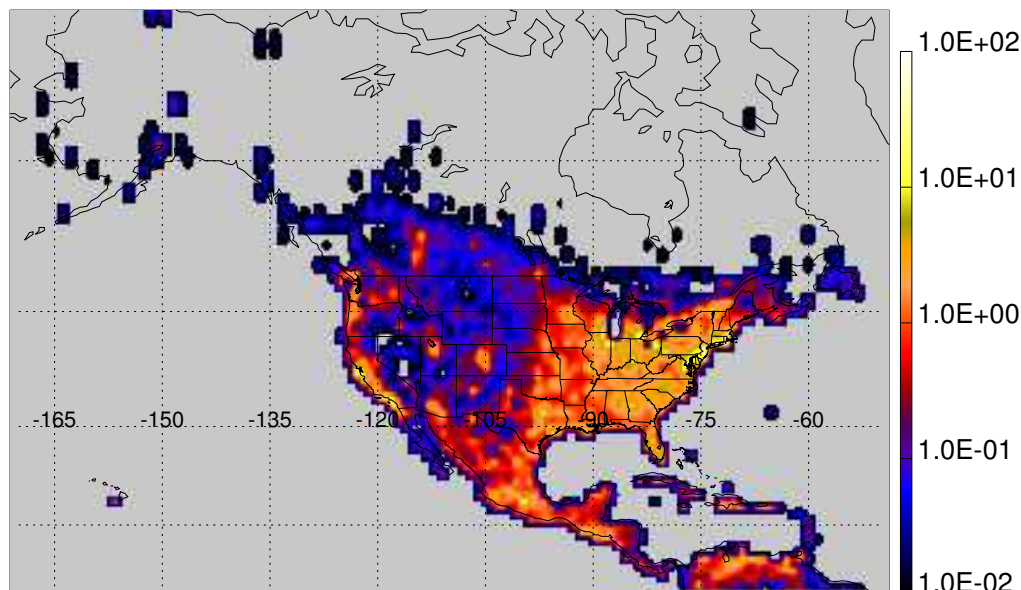


Figure 1.2 Map of US anthropogenic CO emissions, based on the EDGAR Fast Track 2001 emissions inventory.

1.3 Dissertation overview

The following chapters detail the analyses, results, and conclusions made to address the three research objectives. Chapter 2 presents the results of an analysis of the long-range transport of anthropogenic emissions to the Pico Mountain observatory, which addresses the first research objective. Chapter 3 addresses the second and third research objectives by presenting the new plume tracking method, providing an evaluation of the method, and presenting a sample analysis that demonstrates its advantages over traditional plume tracking analyses. The final chapter summarizes these findings and discusses potential future work, which would be based on the findings of the transport analysis in Chapter 2 and the new method presented in Chapter 3. The appendix provides documentation for the implementation of FLEXPART at Michigan Technological University.

Chapter 2

Summertime transport of anthropogenic emissions to the central North Atlantic[†]

2.1 Introduction

Based on extensive study over the past two decades, it is recognized that emissions exported from large urban/industrial regions like the eastern U.S., Europe, and Asia have large-scale impacts on levels of O₃, particles, and other species over downwind continents and in the remote atmosphere [e.g., [*Parrish et al.*, 1998; *Hoell et al.*, 1997; *Trickl et al.*, 2003; *Pochanart et al.*, 2003],

[†]This chapter is based on Owen, R. C., O. R. Cooper, A. Stohl, and R. E. Honrath, An analysis of the mechanisms of North American pollutant transport to the central North Atlantic lower free troposphere, *J. Geophys. Res.*, *111*, D23S58, 2006. Reproduced by permission of American Geophysical Union. Copyright permission details given in Appendix 3.

[*Jaffe et al.*, 1999; *Ryall et al.*, 1998]]. These impacts are important because of their effect on the ability of downwind nations to attain air quality standards [*Li et al.*, 2002; *Park et al.*, 2004], the important role of O_3 as a source of OH radical, and the roles of O_3 and particles in the earth’s energy budget [*Houghton et al.*, 2001].

Most anthropogenic emissions occur at or near ground level over the continents. The details of the export of these emissions from the continental boundary layer affect pollutant lifetimes and transformations downwind. For example, emissions transported to the marine boundary layer (MBL), where photochemical lifetimes are typically shorter, are less likely to exert a large-scale impact than are those exported to the free troposphere (FT). In contrast, emissions rapidly transported to the upper troposphere, where many loss processes are less effective and wind speeds are high, are more likely to have detectable large-scale impacts [*Stohl et al.*, 2002c], but may be less likely to directly affect downwind surface concentrations.

Uplift in the warm conveyor belt (WCB) airstream ahead of cold fronts associated with midlatitude cyclones provides an effective mechanism for rapid transport to the middle and upper troposphere. Episodic WCB export associated with passing cyclones has been identified as a dominant mechanism for rapid transport of both North American and Asian emissions to the middle and upper troposphere [*e.g.*, *Stohl and Trickl*, 2001b; *Cooper et al.*, 2004]. The combination of rapid uplift and rapid transport in the upper troposphere results in impacts on middle to upper tropospheric composition well downwind, and significant enhancements of O_3 , CO, and nitrogen oxides over Europe resulting from WCB export of North American emissions have been documented [*Stohl and Trickl*, 1999; *Trickl et al.*, 2003; *Auvray and Bey*, 2005; *Huntrieser et al.*, 2005]. An alternative mechanism for even more rapid uplift of pollutants to the upper troposphere is provided by convection [*Dickerson*

et al., 1987], leading to significant impacts above the pollutant source region [e.g., *Pickering et al.*, 1990; *Thompson et al.*, 1994], although transport to distant regions may be less rapid [*Li et al.*, 2005].

Export out of the continental boundary layer can also occur at low altitudes, either as the result of episodic ventilation in the post cold front, an airstream associated with passing midlatitude cyclones [*Cooper et al.*, 2002] or via advection in the mean winds, not associated with frontal passages [*Li et al.*, 2002]. Transport speeds in the lower troposphere are less than those in the upper troposphere. As a result, the time for dilution and photochemical transformations is greater, and enhancements in mixing ratios well downwind are expected to be smaller. Perhaps for this reason, lower tropospheric export events have received less attention than WCB export and convection. However, lower tropospheric export fluxes are significant. For example, *Li et al.* [2005] determined that more than one-third of the CO exported from North America in July GOES-CHEM simulations occurred in the lowest 3 km. As noted above, the fate of emissions exported in these events is dependent, in part, on whether transport occurs in the MBL or FT. Although most of the lower 3 km column is within the daytime continental boundary layer, downwind transport is likely to occur in the lower FT as well as in the MBL. This is the result of the differing boundary layer structure over the ocean, relative to that over land. As discussed by *Angevine et al.* [2004], once exported continental air reaches the Atlantic Ocean, a statically stable near-surface layer develops, growing over time and isolating the polluted air mass from the ocean surface. Eventually, a steady-state MBL must develop, but central North Atlantic MBL heights are much smaller than 3 km (e.g., typically less than 1 km during summer over the central and eastern North Atlantic [*Rodrigues et al.*, 2004]). For example, recent measurements have shown that polluted air containing relatively high levels of nitric acid (10s of ppb), can be transported

hundreds of km downwind of source regions. These plumes were observed in layers as low as a few hundred meters altitude, decoupled from the surface and above the marine boundary layer [*Neuman et al.*, 2006].

In order to study the photochemical evolution of North American emissions transported downwind in the lower FT, we established the 2.2 km-altitude PICO-NARE station in the Azores Islands in July 2001. Analyses of CO and O₃ measurements made there during the summers of 2001–2003 demonstrate that events of long range transport in the lower FT occur and are detectable at this location, typically ~ 3 –6 days downwind [*Honrath et al.*, 2004]. Here, we use these measurements in combination with simulations by the FLEXPART particle dispersion model [*Stohl et al.*, 1998] to identify events during which eastern North American emissions significantly impacted the central North Atlantic lower FT during July 2003. The focus of this paper is a descriptive analysis of the meteorological scenarios responsible for four representative events. Three of these originated as lower tropospheric export events and are expected to be typical of the events responsible for lower tropospheric pollution export from eastern North America. The fourth involves an interesting combination of uplift in a WCB followed by subsidence in the same cyclone’s dry airstream.

2.2 Methods

2.2.1 PICO-NARE station measurements

Measurements of CO, O₃, and relative humidity (RH) made at the PICO-NARE station during July 2003 were used to identify periods apparently impacted by upwind emissions of CO and of O₃ precursors, and the CO measurements were used with FLEXPART simulations (described below) to identify

periods of North American emission transport to the station.

The PICO-NARE station is located on the summit of Pico mountain, an inactive volcano on Pico Island in the Azores, Portugal (38.47 degrees north latitude, 28.4 degrees west longitude). Station altitude (2.2 km) is well above the regional MBL, which is typically less than 1 km in height during summer. Upslope flow on Pico mountain is not frequent, even during summer, occurring on 39% of the days during a period of intensive meteorological measurements in summer 2004, and bringing MBL air to the summit on only a fraction of those days. During July 2003, the period analyzed here, about one-half of the days experienced the low synoptic winds and strong insolation necessary for upslope flow on Pico mountain. Of the events discussed in detail below, only the last 3 hours of event 4 are potentially affected by buoyant upslope flow. However, relative humidity remained below 50% during that period, indicating a lack of significant MBL impact.

Carbon monoxide was determined using a modified commercial nondispersive infrared absorption instrument (Thermo Environmental, Inc., Model 48C-TL), calibrated daily with standards referenced to the NOAA CMDL standard. Ozone was obtained with two commercial ultraviolet absorption instruments (Thermo Environmental Instruments Inc., Franklin, Massachusetts; Model 49C). Based on comparisons between these instruments and a NOAA CMDL network ozone standard conducted in 2001 and 2004 and between the two instruments in 2003 and 2004, the raw measurements from the first instrument (used prior to July 19) were reduced by 0.2%, while those from the second instrument (used after July 19) were increased by 2.6%. The resulting O₃ mixing ratios are 5 to 7% lower than those reported previously [*Honrath et al.*, 2004], as a result of the additional intercomparisons now available. The 30-minute average CO measurements used here have 2- σ precision of ± 3 ppbv or better; accuracy is $\pm 6\%$. For O₃, precision is 1 ppbv or better, and ac-

curacy is estimated as ± 3 %, the maximum adjustment applied to the July 2003 raw O₃ measurements.

The July 2003 CO and O₃ measurements have been presented previously [Honrath *et al.*, 2004]. Further details on the PICO-NARE station, the measurements made there, and the frequency of upslope flow to the station are provided elsewhere [Honrath *et al.*, 2004].

2.2.2 FLEXPART

We used the FLEXPART (version 5.1) particle dispersion model [Stohl *et al.*, 1998; Stohl and Thomson, 1999] to simulate CO enhancements at the PICO-NARE station resulting from the transport of North American emissions. Forward simulations were used for visualizing the advection and dispersion of CO due to the synoptic scenario. Backward simulations, which produce a cloud of particles that is traced backward in time (termed a retroplume) were primarily used in the same manner as backward trajectories: to trace the flow of air, helping to identify important source regions. However, a major advantage over simple back trajectories is that they can be used to calculate a time series of the contribution of North American CO to the total CO at the station by multiplying the residence times of retroplume particles in the lowest 300 m over North America, a measure of the sensitivity to emission input, by the emissions flux at each location, as described by Seibert and Frank [2004] and used in previous studies [Stohl *et al.*, 2003; Huntrieser *et al.*, 2005]. Simulations were conducted for the month of July 2003.

FLEXPART was driven with data from the European Centre for Medium Range Weather Forecasts (ECMWF) [ECMWF, 1995] with 1° horizontal resolution, 60 vertical levels and a temporal resolution of 3 hours, using meteorological analyses at 0000, 0600, 1200, and 1800 UTC, and ECMWF 3-hr

forecasts initialized at these times. It has been found that the use of analyzed fields can result in an overestimation of mixing between airmasses [*Stohl et al.*, 2004b]. While forecast fields do not experience this overestimation, they will drift away from the real meteorological situation and so the use of analyzed fields is a necessity for any simulation beyond a few hours. In general, the backward simulations show that our events have a fairly coherent meteorological situation and exhibit relatively little mixing between the source and the station. Therefore, we do not believe this effect is significant enough to change the conclusions of this paper.

In the forward mode, CO emissions were released into the lowest 150 m of the atmosphere over North America. CO emissions were based on the EDGAR Version 3.2 inventory [*Olivier and Berdowski*, 2001], which uses 1995 as the base year and has a 1° resolution. A total of 11 million particles were released per 20 days over North America with particles released over 3-hour intervals. The number of particles released into each grid cell was determined by scaling the total number of particles released by the emissions in each grid cell. Particles were dropped from the simulation after 20 days and were conserved up to that time. Thus, the simulations model only CO enhancements caused by North American emissions over the previous 20 days.

Retroplumes were initiated every three hours with 25,000 particles released over a three-hour time interval into a $1^\circ \times 1^\circ$ grid box centered on the PICO-NARE station, over an altitude range of 1750 m asl to 2750 m asl. Retroplume simulations were run for 20 days backward in time. To calculate CO mixing ratios at the PICO-NARE station resulting from North American emissions, the same EDGAR emission inventory was used with the retroplume results, using the method of *Seibert and Frank* [2004]. No attempt was made to adjust the emission inventory for the CO emissions reductions that have occurred since 1995 [*Parrish et al.*, 2002], as moderate biases in the

FLEXPART-predicted mixing ratios would not affect our interpretation.

2.2.3 Event selection

Only periods with observed CO above 95 ppbv during the study period of July 2003 were identified as potentially impacted by North American outflow. This cutoff value was selected as the approximate dividing point between apparent background observations and enhanced-CO observations in the summer 2003 PICO-NARE measurements [Honrath *et al.*, 2004]. These periods of elevated CO were then further subdivided based on evidence of changes in flow pathways, as indicated by the FLEXPART retroplumes, and apparent changes in airmasses, as indicated by correlated changes in the CO and O₃ mixing ratios and relative humidity. For example, the period from 1900 UTC on July 2 to 1230 UTC July 6 was identified as potentially affected by North American outflow, as CO levels were well above the 95 ppb cutoff. However, this period was divided into two events based off an abrupt change in CO levels (Figure 2.1) and a change in the flow path indicated by the FLEXPART retroplumes (not shown).

2.3 Results

CO measurements and FLEXPART-simulated North American CO at the PICO-NARE station during the July 2003 study period are shown in Figure 2.1. The 16 events selected as described above are summarized in Table 2.1. These 16 events account for 52% of the 416 CO measurement hours during the study period. Mean observed CO and O₃ levels during each event may be compared to the most common CO level observed at the PICO-NARE station during apparent background periods in summer 2003—approximately

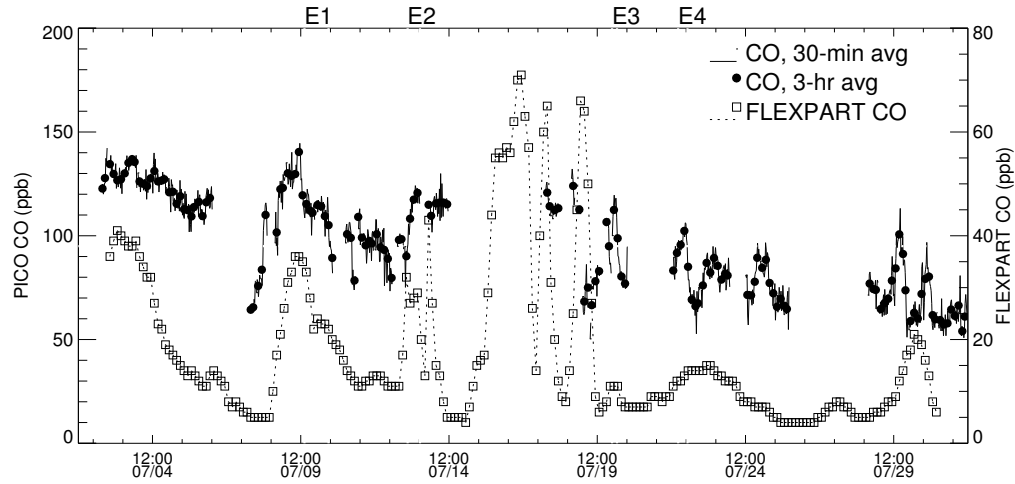


Figure 2.1 Time series of 30-min (solid line) and 3-hr averaged (filled circles) CO measured at the PICO-NARE station, and CO from FLEXPART retroplumes (open boxes and dotted line) initialized over a 3-hr period. The case study events discussed in the text are marked with black vertical lines.

80 ppb (the “low-CO mode” [Honrath *et al.*, 2004])—and to the median O_3 mixing ratio during periods when CO was below 80 ppbv—24 ppbv.

Also shown in Table 2.1 are three columns based on the FLEXPART simulations: the mean North American contribution to the CO mixing ratio calculated using FLEXPART during each event, and the travel time and transport height between the responsible source region and the station. Travel times were estimated using 24-hr-average retroplume locations. Transport heights were estimated using retroplume clusters. (More precise information on transport heights is provided for the four case studies below.) The North American CO contributions summarized in Table 2.1 average 20 ppbv, and contributions of at least 10 ppbv are simulated during all events but one. Importantly, the events listed in Table 2.1, which were actually detected at the PICO-NARE station, account for all but four of the periods when FLEXPART-simulated North American CO contributions exceeded 10 ppbv. Of the four FLEXPART events not detected in the measurements, one was the result of biomass-burning emissions present in the EDGAR emission in-

ventory in a region of Canada with no active fires during July 2003, and two had only moderate FLEXPART CO enhancements of less than 15 ppbv. This high degree of consistency between CO events simulated by FLEXPART and those detected at the PICO-NARE station indicates that FLEXPART is adequately simulating the transport responsible for North American pollution transport to the PICO-NARE station.

The events listed in Table 2.1 include each of the July 2003 events identified in our previous analysis [*Honrath et al.*, 2004], in which events were identified solely based on the occurrence of CO levels above 114 ppbv. However, some of these events have been further subdivided here based on changes in flow, and Table 2.1 includes three additional events on July 11–12 as a result of the reduced CO cutoff used here. Our previous analysis identified North American anthropogenic emissions as a probable contributor to the enhanced CO observations during each of the July 2003 events also identified here, but also noted potentially significant impacts from biomass-burning emissions during several: from Siberian fires during events a–b and from fires in the western U.S. during events c–n. Our analysis here focuses on the transport of anthropogenic emissions, but is not inconsistent with the occurrence of upwind biomass-burning emissions.

Based on a preliminary analysis of the events shown in Table 2.1, we selected for detailed analysis four periods for which the mechanisms of export from North America and transport to the Azores were relatively clear and that appeared to be representative of the events during the study period. These four events are numbered in Table 2.1 and indicated in Figure 2.1.

The remainder of this section provides a descriptive analysis of each of these four events. The presentation of each event is divided into two parts: the export of emissions out of the North American boundary layer, including relevant aspects of the preceding meteorological situation, and the subse-

quent transport to the PICO-NARE station. This discussion makes use of the FLEXPART forward and retroplume simulations, GOES-EAST and METEOSAT infrared satellite images, and sea-level pressure from NOAA NCEP Aviation Model analyses, which are presented in plots described in detail in the following section.

Table 2.1 Events of elevated CO observations during July 2003.

Event	Start Time ^a	End Time ^a	Observed CO ^b	Observed O ₃ ^b	N. American CO ^c	Transport time ^d	Transport Height ^d
a	7/02 1900	7/04 2230	129 ppb	48 ppb	34 ppb	5-7 days	2-4 km
b	7/04 2300	7/06 1230	116 ppb	NA	14 ppb	5.5 days	2-4 km
c	7/08 0600	7/08 0900	109 ppb	NA	6 ppb	N/A	N/A
d	7/08 1530	7/08 1700	101 ppb	NA	15 ppb	4.5 days	3-4 km
e	7/08 1800	7/09 1500	128 ppb	NA	31 ppb	4.5 days	2-5 km
f: 1	7/09 1530	7/10 1030	113 ppb	NA	25 ppb	5 days	2-6 km
g	7/11 0030	7/11 0530	102 ppb	NA	15 ppb	6 days	2-6 km
h	7/11 1100	7/12 0730	101 ppb	30 ppb	14 ppb	5 days	2-4 km
i	7/12 1900	7/12 2300	98 ppb	28 ppb	15 ppb	5 days	2-3 km
j: 2	7/13 0300	7/13 1230	114 ppb	56 ppb	30 ppb	4.5 days	2-7 km
k	7/13 2000	7/14 1130	114 ppb	68 ppb	22 ppb	5.5 days	2-8 km
l	7/17 1800	7/18 0430	115 ppb	NA	45 ppb	7 days	2-4 km
m	7/18 1530	7/18 1800	124 ppb	NA	14 ppb	5 days	2-4 km
n: 3	7/20 0030	7/20 0400	113 ppb	50 ppb	11 ppb	5 days	2 km
o: 4	7/22 0600	7/22 1300	100 ppb	51 ppb	15 ppb	7 days	2 km
p	7/29 1430	7/29 1930	101 ppb	49 ppb	14 ppb	5 days	2 km

NA, Not available. ^aTimes are in UTC, which is approximately 2 hours later than local solar time. ^bMean mixing ratios observed during

the indicated periods. ^cMean North American contribution to observed CO, calculated using FLEXPART as described in the text. ^dTravel

time and approximate altitude of FLEXPART retroplume particles released during the event that reached eastern North America.

2.3.1 Event 1: Direct transport from North America associated with the decaying tropical storm Bill

The results of the FLEXPART retroplume simulation initialized during event 1 (0000–0300 UTC July 10) are displayed in Figure 2.2. This figure is divided into three parts. Figure 2.2a shows the column-integrated residence time of retroplume particles. Plots of this type are most useful for identifying the overall horizontal transport pathway during the event. Figure 2.2b shows the residence time of retroplume particles in the lowest 300 m, termed the footprint layer. Footprint layer plots are most useful for identifying regions in which ground-level sources have the potential to influence pollutant levels during the event. Finally, Figure 2.2c shows the product of the footprint layer residence time and the CO emission inventory. This is the contribution of sources in each grid cell to the FLEXPART-calculated North American CO at the PICO-NARE station during this event. The color scales in Figure 2.2a and Figure 2.2b are scaled to the maximum residence time, which is indicated below the color scale. For Figure 2.2c, the color scale indicates the mixing ratio contribution of each grid cell, in units of ppbv.

The overall transport pathway during event 1 is apparent in Figure 2.2a, which indicates a coherent retroplume back to the U.S. east coast. (The blue colors surrounding the red, straight path result from recirculation, not direct transport.) Figure 2.2b shows that the plume resided in the footprint layer primarily over the heavily populated regions in the eastern U.S., with smaller footprint residence times covering most of North America, the Atlantic south of the Azores and east of the U.S., and western Africa. Figure 2.2c confirms that the majority of the FLEXPART-simulated CO came from the northeastern U.S.

Results from the retroplume simulation are presented in a second way in

the left column of Figure 2.3 (a, c, e, and g). These figures show the location of retroplume particles on a single day, overlain on sea-level pressure isobars and infrared GOES-EAST and METEOSAT images. These plots indicate that the air sampled during this event was over the region with the highest CO contribution on July 3 (Figure 2.3a) and 4 (Figure 2.3c), 6–7 days prior to arrival at the station.

The right column of Figure 2.3 (b, d, f, and h) shows the results of the forward FLEXPART simulations, overplotted on the same infrared images, but this time in color, with reds and yellows indicating colder higher altitude cloud tops, and greens and dark blue indicating mid-level clouds and the Earth’s surface, respectively. Isolines of North American CO columns integrated over 0–3 km (mg/m^2) are plotted over infrared images. These plots are useful for putting the events detected at the PICO-NARE station into the context of low-level CO export from North America. These images are discussed further below.

2.3.1.1 Export out of the U.S. boundary layer in a low pressure system

The meteorological situation that led to the export for this event began on July 1, 9 days prior to its arrival at the PICO-NARE station, when tropical storm Bill made landfall over southern Mississippi. The low pressure system associated with the storm was relatively weak, with a core sea level pressure of 998 hPa at 0000 UTC on July 1. From July 1 to July 3 the cyclone weakened, reaching 1009 hPa at 0000 UTC on July 3, as it tracked northeast over the southeastern U.S., reaching the Mid-Atlantic states on July 3 (–7 days).

During these three days, a general region of high pressure persisted over the eastern U.S. and Atlantic ocean. The high over the U.S. dampened convection in regions not affected by the decaying tropical storm. The high

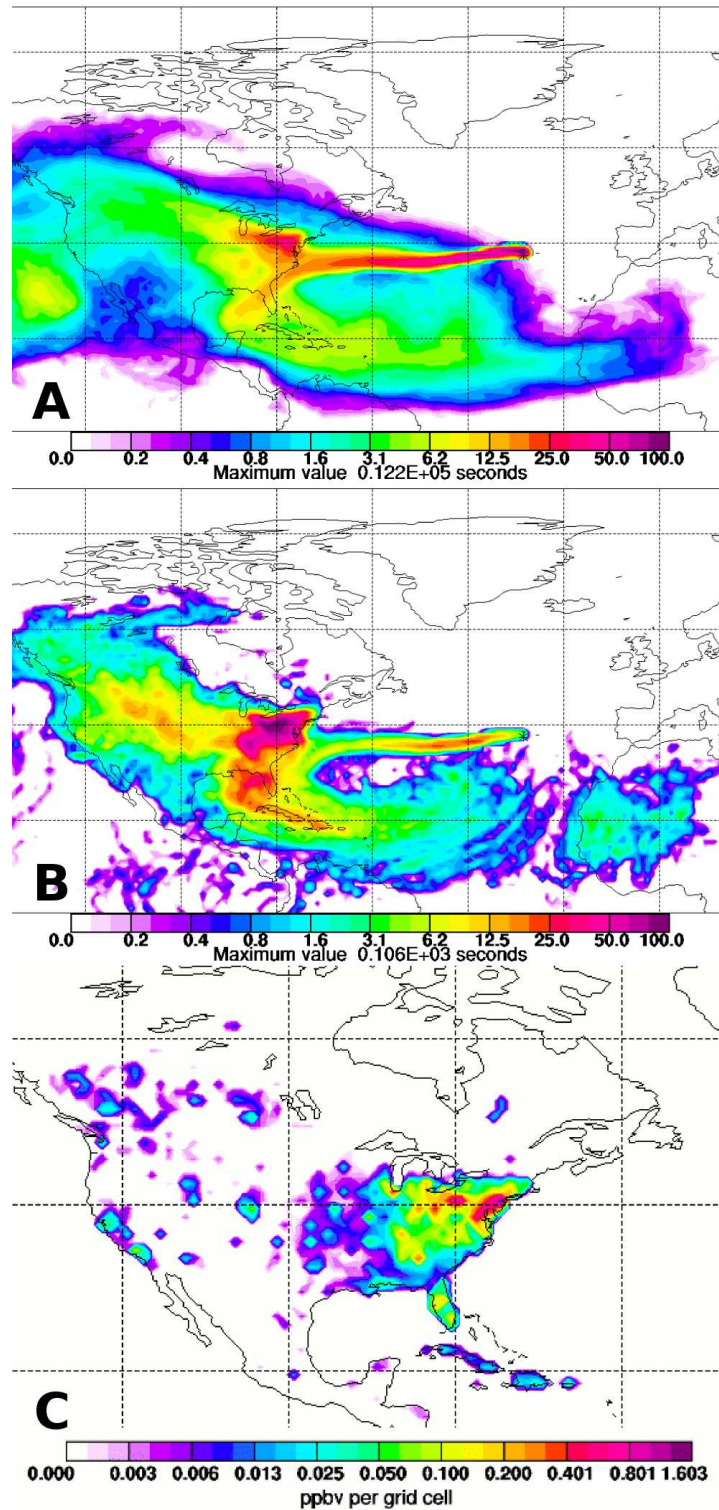


Figure 2.2 FLEXPART retroplume results for event 1. (a) column integrated residence time, (b) residence time in the 0–300 m footprint layer, and (c) source contributions to CO mixing ratio calculated during this event. The retroplume was initiated from 0000–0300 UTC on July 10.

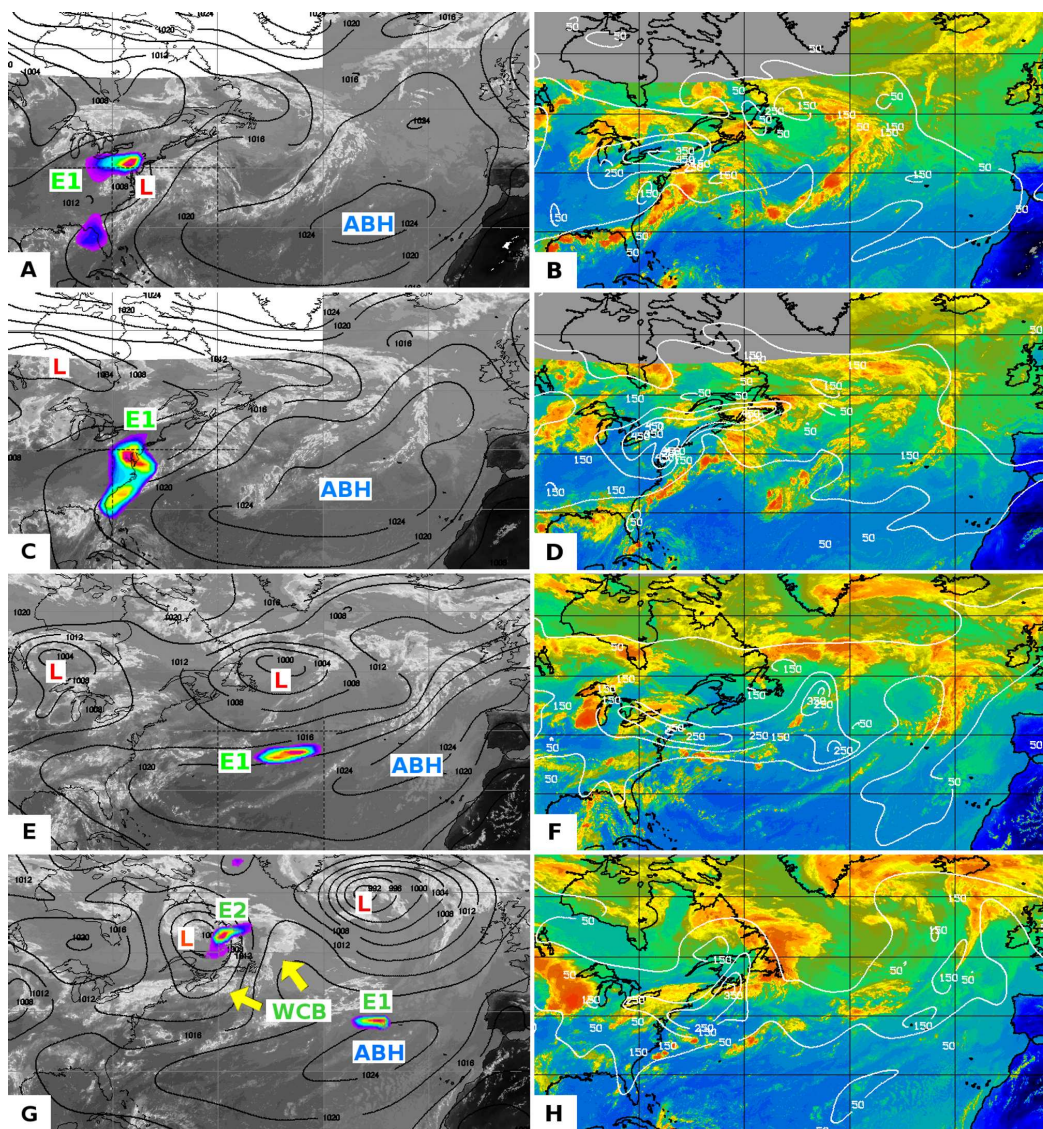


Figure 2.3 The 24-hr average location of the retroplume for Event 1, plotted on top of combined GOES-EAST and METEOSAT infrared image and sea level isobars (left column) and 0–3 km CO column isolines (right column) plotted on top of the same infrared image shown in false color. The retroplume shown in the left column was initialized during event 1 (0000–0300 UTC July 10). Times shown are 1200 UTC July 3 (a and b), 1200 UTC July 4 (c and d), 1200 UTC July 7 (e and f), and 1200 UTC July 9 (g and h). Labels on the left-hand plots indicate the retroplume for event 1 (E1) and pertinent meteorological features (low pressure centers, L, associated warm conveyor belts, WCB, and the Azores-Bermuda High, ABH). The event 2 retroplume (E2) is also indicated in part g.

pressure over the Atlantic formed the western edge of the Azores-Bermuda High (ABH); its presence reduced the zonal export of pollution out of the region. As the low tracked northeast, the high over the U.S. diminished, though strong convection was still limited to the regions of the decaying storm. By July 3, the high over the U.S. had dissipated, leaving only the ABH off shore (Figure 2.3a), where it continued to prevent significant zonal pollution transport (Figure 2.3b).

Between 1200 UTC on July 3 and 0000 UTC on July 4, the low remained situated over Maryland and Delaware. It then tracked north, reaching Massachusetts by 1200 UTC on July 4 (−6 days, Figure 2.3c) This is the region where export of pollution had been dampened during the previous 3 days.

During the first half of July 4, a region of elevated CO developed in the area of the low. The region of intense CO over southern New England is visible in the forward FLEXPART simulation, with column densities exceeding 500 mg/m^2 for the 0–3 km column (Figure 2.3d). (The total column had only slightly higher maximum column densities in the region of high CO: 550 mg/m^2 , not shown.)

Early on July 4, a new low pressure system began to develop over northeastern Canada. As the low tracked east toward the Canadian east coast, the northwestern edge of the ABH began to retreat. This allowed increased zonal transport and, by July 5, the column of pollution moved off shore, with the plume located about midway between the ABH and the Canadian low.

2.3.1.2 Transport to the station within geostrophic winds

From July 5 (−5 days) to July 7 (−3 days), as the low tracked eastward over and across the Atlantic (Figure 2.3e), the plume also tracked eastward in the geostrophic wind between the ABH and the low. A large, coherent airstream, or river of CO is clearly visible in the forward FLEXPART results on July

7 (-3 days) as North American emissions were channelled out of the U.S. boundary layer through the two pressure features, with the portion of the plume for this event at the leading edge of the river of CO (Figure 2.3f).

On July 7, the low intensified, and by July 8 (-2 days) it had begun to track northeast. As the low moved northeast, away from the Azores, the ABH expanded northwest into the region west of the low. The retreat of the low to the northeast and the advance of the ABH to the northwest left the plume situated close to the center of the ABH early on July 9 (-1 day) (Figure 2.3g and h). During the last day of travel, the plume experienced slower transport than during the previous 4 days, but continued eastward to the Azores in the relatively weak westerly winds near the northern edge of the ABH, arriving at the PICO-NARE station while it was near the center of the ABH.

In summary, the elevated CO levels observed at the Azores during this event were the result of a series of events beginning with an accumulation of pollution emissions over the eastern U.S. caused by a stagnant high-pressure system. The high over the U.S. diminished as a weak low (the result of a decaying tropical storm) moved into the region, and the accumulated emissions were mixed through the boundary layer near the center of the low. The pollution was then transported offshore in a weak westerly wind, and then to the PICO-NARE station in the geostrophic wind between the ABH and the low to the north. The path of transport was governed by the relative location of the ABH and the transient northern low. The CO enhancement sampled at the PICO-NARE station during this event was the result of the southern edge of a river of CO, which was channeled across the Atlantic between the ABH and the northern low (Figure 2.3g and h).

2.3.2 Event 2: Lofting and subsidence associated with a midlatitude cyclone

The retroplume pathway for this event, shown in Figure 2.4a, differs significantly from that of event 1. The air sampled during this event traveled directly from the northeastern Canadian coast, but originated in the footprint layer (0–300 m) over the northern U.S. and southern Canada (Figure 2.4b). The FLEXPART-simulated North American CO during this event, shown in Figure 2.4c, included contributions from a rather large region of the northern U.S. and southern Canada, with particularly strong signals from Boston, Toronto, Detroit, and Chicago. The retroplume originated over this region on July 9, 4.5 days before arriving at the PICO-NARE station.

2.3.2.1 Export out of the U.S. boundary layer in the westerly wind

The features of this event are distinctly different from those of event 1, beginning with the meteorological scenario that led to the export. On July 7 at 0000 UTC (−6 days) a low pressure system was located over southern Manitoba and Ontario. Unlike the case prior to event 1, however, a high pressure system was not present and the eastern U.S. did not experience dampened export. Instead, pollution was being actively exported out of the U.S. boundary layer. Figures 2.3e and f, discussed above in the context of event 1, show continuing CO export on July 7, with a CO plume being channelled out of the U.S. between the ABH and the low responsible for the transport of event 1, located over the northwestern Atlantic at this time.

The next day, the Canadian low tracked east, and by 0000 UTC on July 8 (−5 days), it was located over northern Quebec and Newfoundland. At this time the low, in conjunction with the ABH, was still zonally channelling CO into the Atlantic. Later on July 8, a region of high pressure began to form

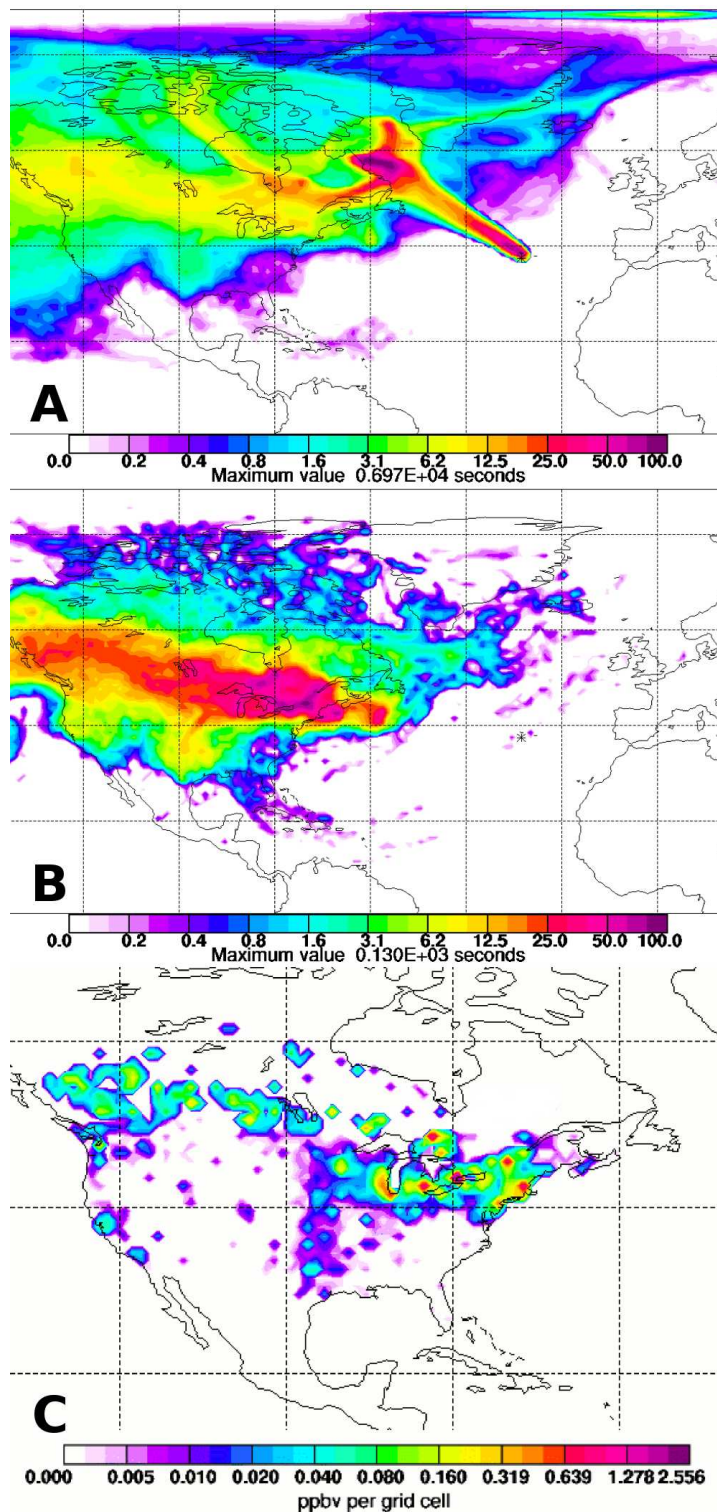


Figure 2.4 FLEXPART retroplume results for event 2. (a) column integrated residence time, (b) residence time in the 0–300 m footprint layer, and (c) source contributions to CO mixing ratio calculated during this event. The retroplume was initiated from 1200–1500 UTC on July 13.

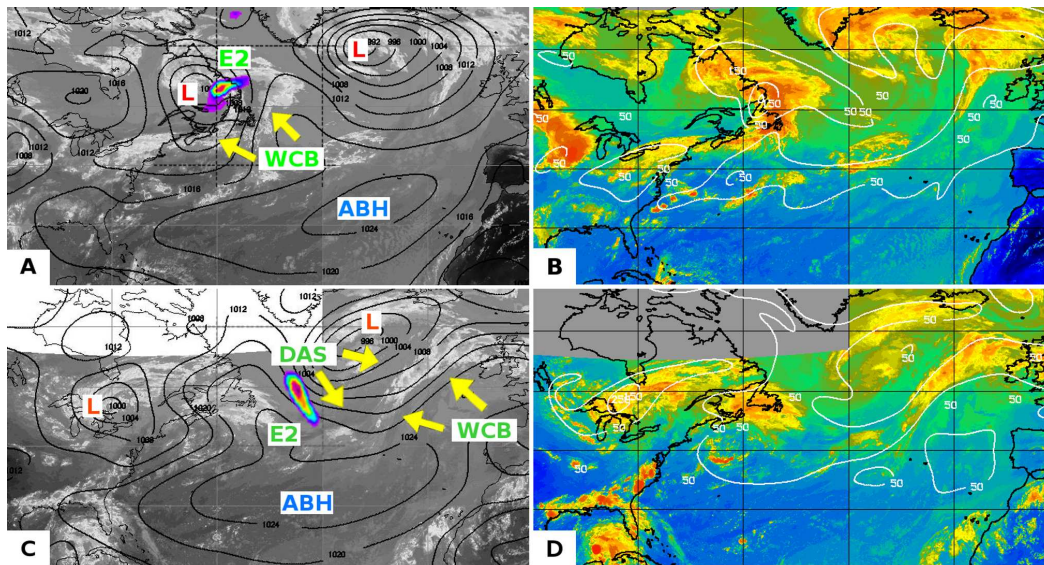


Figure 2.5 The 24-hr average location of the retroplume for Event 2, plotted on top of combined GOES-EAST and METEOSAT infrared image and sea level isobars (left column) and 4–20 km CO column isolines (right column) plotted on top of the same infrared image shown in false color. The retroplume shown in the left column was initialized during event 2 (1200–1500 UTC July 13). Times shown are 1200 UTC on July 9 (a and b) and 0000 UTC on July 12 (c and d). Labels on the left-hand plots indicate the retroplume for event 2 (E2) and pertinent meteorological features (low pressure centers, L, associated warm conveyor belts, WCB, and dry airstreams, DAS, and the Azores-Bermuda High, ABH).

between the Canadian and Atlantic lows. This region of high pressure, coupled with the intensification of the Canadian low, shifted most of the low-level flow of CO to the north-northeast, toward the center of the low, early on July 9 (-4.5 days, shortly before the time shown in Figure 2.5a). By this time, the low was over northeastern Quebec and Newfoundland. As a result, the most intense portion of the CO plume had moved northeast from its source over the Mid-Atlantic states and southern New England. At 0000 UTC on July 9, part of the plume was located over northern Maine and Nova Scotia. The larger portion was over the western Atlantic, off the east coast of Maine, distributed from the surface up to about 3 km, with the bulk in the lower kilometer.

2.3.2.2 Rapid lofting and subsidence

Shortly after the shift in the flow regime, the cold front associated with the Canadian low passed over the northeastern U.S., southeastern Canada, and the adjacent portion of the Atlantic, including a large section of the CO plume. As a result, the CO plume was lofted in the warm conveyor belt ahead of the front. This frontal passage began at approximately 0000 UTC on July 9 (-4.5 to -4 days). The location of the retroplume at 1200 UTC on July 9, shortly after the cold front's passage, is shown in Figure 2.5a. Figure 2.5b shows the 4–20 km CO column calculated by the forward FLEXPART simulation at this time; the lofted CO plume is apparent near the center of the comma cloud associated with the WCB uplift. (In contrast, Figure 2.3h shows much less CO present in the 0–3 km column at the retroplume's location at this time.)

Over the next three days, the Canadian low continued to track eastward, with the pressure center moving over the Atlantic early on July 10 (-3 days). Around July 12 (-1.5 days), the low began to weaken and track northeastward. During this time, the cold front continued to sweep eastward, eroding the warm sector.

During the cyclone’s trek over the Atlantic, the retroplume of North American CO emissions destined to reach the PICO-NARE station also tracked eastward, traveling cyclonically around the center of the pressure system from the eastern side on July 9 (−4 days), to the northern side on July 10 (−3 days), and finally to the western side on July 11 (−2 days). The plume continued to rise during July 9 and 10, reaching a maximum height of approximately 6.5 km at 0000 UTC on the 11th (−2.5 days). After arriving on the western side of the low, the retroplume became entrained in the cold descending air behind the cold front (the dry airstream [*Cooper et al.*, 2001]). Once this happened, the retroplume rapidly descended toward the southeast, and by 0000 UTC on July 12 (−1.5 days) it was travelling rapidly toward the Azores (Figure 2.5c). It arrived at the station on July 13, immediately behind the cold front.

In summary, the events that led to the elevated CO and O₃ levels observed over the Azores during this event began with the low-level export of emissions from the U.S. boundary layer in westerly winds. This was followed by the lofting of the pollution to 6–8 km altitude in the warm conveyor belt ahead of the cold front associated with a passing northerly cyclone. However, despite being lofted to high altitude, the plume was observed at 2.2 km over the Azores, as a result of incorporation into the same cyclone’s descending dry air stream. A similar process was observed by *Cooper et al.* [2001] over the western North Atlantic Ocean, but was not observed to descend as strongly as in the present case study.

2.3.3 Events 3 and 4: Export in a weak warm conveyor belt followed by transport around the Azores High

Two additional episodes of elevated CO were observed at the station in the period of July 20–22, 2003. The first (event 3) arrived during 0000–0400 UTC July 20 and the second (event 4) arrived during 0600–1300 UTC July 22. Though these events arrived two days apart, they were exported out of the U.S. boundary layer in the same frontal system and had similar transport pathways until July 18, when their paths diverged.

Using FLEXPART retroplumes initiated during 0000–0300 UTC July 20 for event 3, and 1200–1500 UTC July 22 for event 4, the retroplumes can be traced back to the U.S. east coast, where the bulk of the CO originated over New England and the Mid-Atlantic states on July 15 (–5 days and –7 days for events 3 and 4, respectively). The two events are described together here because they were exported at the same time and the subsequent transport is closely linked.

Column-integrated retroplume residence times are shown for both events in Figures 2.6a and b. Their general features are very similar, though event 3 (Figure 2.6a) exhibits a greater degree of dispersion. The footprint layer residence times (Figures 2.6c and d) indicate that most time spent in the footprint layer was over the eastern and midwestern U.S. for both plumes, though the plume for event 4 (Figure 2.6d) also spent time in the marine boundary layer south of the Azores. Figures 2.6e and f show broad CO contributions from the eastern U.S., with the Boston area being particularly important for event 3.

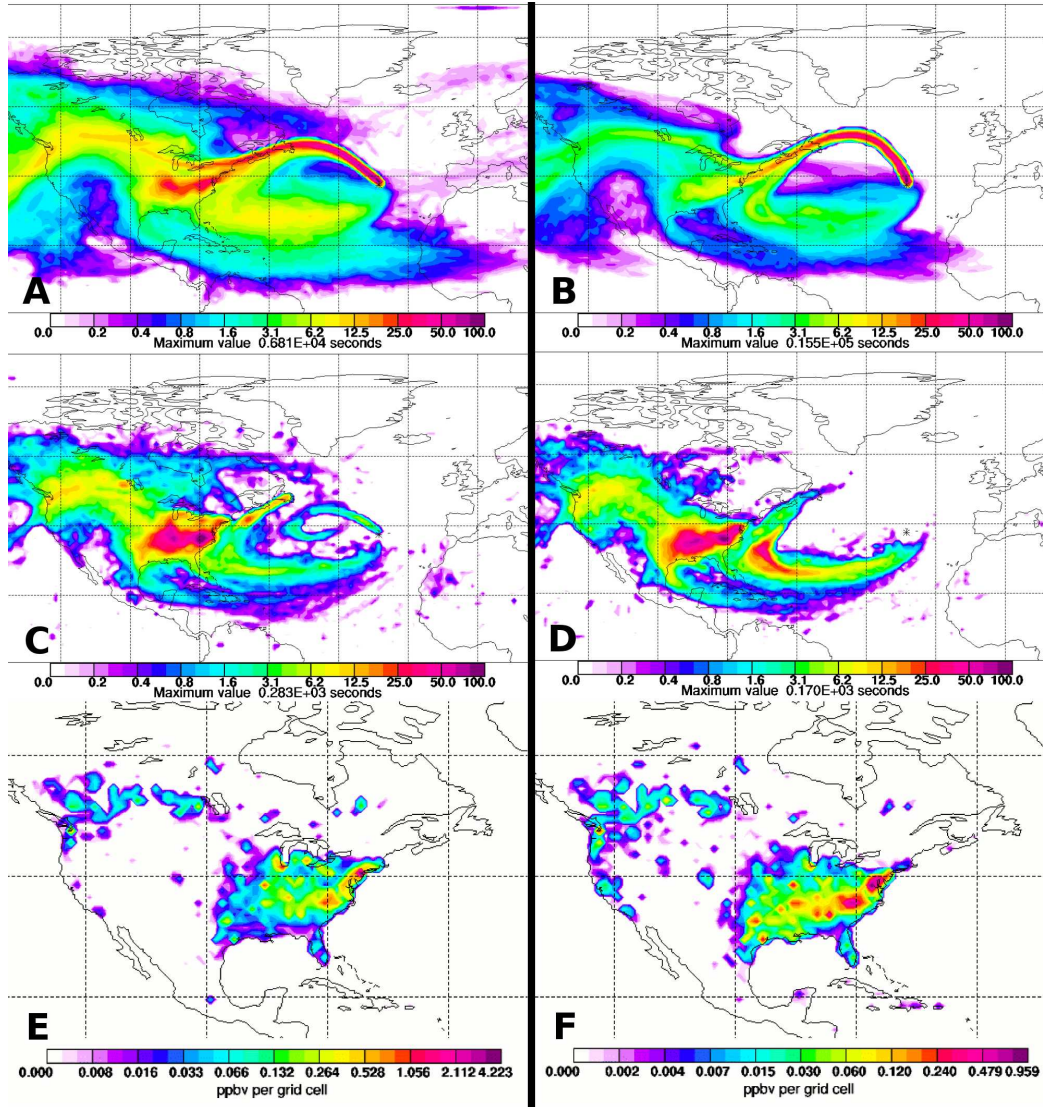


Figure 2.6 FLEXPART retroplume results for events 3 (left) and 4 (right). (a) column integrated residence time for event 3, (b) column integrated residence time for event 4, (c) residence time in the 0–300 m footprint layer for event 3, (d) residence time in the 0–300 m footprint layer for event 4, (e) source contributions to CO mixing ratio calculated during event 3, and (f) source contributions to CO mixing ratio calculated during event 4. The retroplume for event 3 was initiated 0000–0030 UTC July 20; that for event 4 was initiated 0000–0300 UTC July 22.

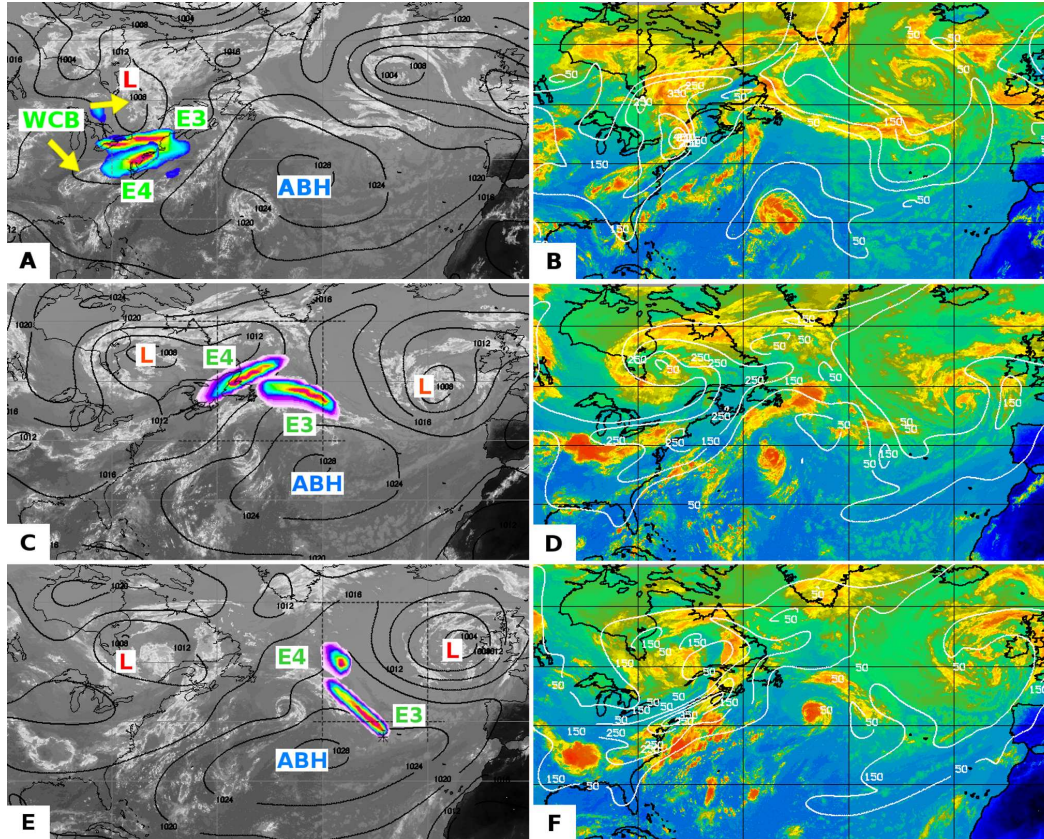


Figure 2.7 The 24-hr average locations of the retroplumes for Events 3 and 4, plotted on top of combined GOES-EAST and METEOSAT infrared image and sea level isobars (left column) and 0–3 km CO column isolines (right column) plotted on top of the same infrared image shown in false color. The retroplumes shown in the left column were initialized during event 3 (0000–0300 UTC July 20) and event 4 (0000–0300 UTC July 22). Times shown are 1200 UTC on July 16 (a and b), 1200 UTC on July 18 (c and d), and 1200 UTC on July 19 (e and f). Labels on the left-hand plots indicate the retroplume for events 3 and 4 (E3 and E4) and pertinent meteorological features (low pressure centers, L, associated warm conveyor belts, WCB, and the Azores-Bermuda High, ABH).

2.3.3.1 Export out of the U.S. boundary layer in a weak warm conveyor belt

On July 15, there were two meteorological features that contributed to the export responsible for these events: a region of high pressure over the eastern U.S. and a weak low pressure system over Canada, north of Lake Superior. The high dampened convection over the northeastern U.S. and reduced the export of pollution out of the U.S. boundary layer, as was also the case prior to event 1. The low will be responsible for the export that resulted in this event.

During July 15 the low strengthened, and a cold front formed and began to move southeast. The low tracked east-southeast, with the center of the low passing over southeastern Ontario and southern Quebec on July 16. Meanwhile, the high pressure system retreated eastward as the low pushed into the region (Figure 2.7a). The cold front continued to move southeast around the low pressure center, reaching the U.S. east coast around 1200 UTC on July 16. As the front passed over the region, the CO plume was exported out of the U.S. boundary layer in a warm conveyor belt, which was visible as a thin line of clouds that extended from the mid-western U.S. to the northeastern U.S. coast (Figure 2.7a and b). The export height, however, was limited to the lower free troposphere due to the relatively weak intensity of the low pressure system and the small temperature difference across the front. This is indicated in the FLEXPART forward results, which include the majority of the plume within the lower 3 km of the atmosphere. For this reason, 0–3 km CO columns are shown in Figures 2.7b, d, and f.

The daily-averaged retroplumes for each event (Figure 2.7a) indicate only slight differences in location during the export phase on July 16, despite their arrival time at the Azores two days apart. However, the export of retroplume

particles arriving during event 4 occurred a few hours later and farther south than that for event 3.

2.3.3.2 Transport to the station in gradient winds

By 1200 UTC on July 17, the cold front had dissipated, and the low pressure system had merged with a second low, creating a large region of low pressure over northeastern Canada. At this time the ABH was located over the central North Atlantic, centered roughly midway between the Azores and the North American east coast. The formation of the larger low pressure region caused the northeastern edge of the ABH to retreat, leaving the CO plume located between the two pressure systems, with the portion of the plume for event 3 closer to the ABH and the portion of the plume for event 4 closer to the low.

During the next day, the main CO plume tracked northeast, as it was channelled by the resulting gradient wind. By July 18, the two retroplumes had diverged significantly, as the event 3 retroplume had travelled primarily east due to its proximity to the ABH, while the event 4 retroplume had travelled northeast, due to its proximity to the Canadian low (Figures 2.7 c and d).

As the event 3 retroplume moved east of the influence of the ABH and the Canadian low, it encountered another low pressure system located northeast of the Azores. As a result, on July 18 and 19, the event 3 retroplume travelled southeast towards the Azores in the gradient wind between the ABH and the Atlantic low (Figures 2.7c and e). Meanwhile, the event 4 retroplume continued to travel northeast until approximately 0000 UTC July 19, when the Atlantic low began to influence its transport (about 1.5 days later than for event 3), and when it began to track southeast toward the Azores (Figure 2.7e).

Over the next day, both plumes continued to track southeast toward the

station, with event 3 arriving on July 20. The transport of the event 4 retro-plume was slowed, however, by the retreat of the Atlantic low to the northeast, which left the plume in a weaker gradient wind closer to the center of the ABH. By July 21, the low had advanced far enough to the northeast that the plume was no longer being channelled between the pressure features. Instead, it was on the eastern edge of the ABH where it experienced significantly reduced transport speeds during its last day of travel, before arriving at the station on July 22.

In summary, the elevated CO and O₃ levels observed over the Azores during these events were the result of the accumulation of emissions under a stagnant high-pressure system, followed by weak lofting to the lower free troposphere in a weak warm conveyor belt associated with a low passing to the north. The emissions were then transported to the Azores in gradient winds governed by the relative location of a Canadian low, an Atlantic low, and the ABH. This pair of events has the interesting attribute of being exported as essentially one plume, while arriving separated by two days. This was the result of initially small location differences that were amplified as the plumes were advected by the gradient winds between the ABH and two evolving lows to the north.

2.4 Discussion

The four events analyzed here are typical of those reaching the central North Atlantic lower FT during the summertime, as indicated by observations at the PICO-NARE station and summarized in Table 2.1. Transport in the lower troposphere was responsible for the majority of these events: the FLEXPART forward simulations indicate that transport of North American CO emissions to the Azores occurred primarily below 3 km in three of the four events

analyzed in detail (events 1, 3, and 4), and in a total of 13 of the 16 events (84% of the total event hours) in Table 2.1. Significant O₃ enhancements were also observed in most of these events (Table 2.1) and in additional similar events in 2001 and 2003 that we analyzed previously [Honrath *et al.*, 2004]. The previous analysis was based on backward trajectories, and identified North American emissions as a significant contributor to the events analyzed here; the present work confirms this. The previous analysis also identified biomass burning as a possible contributor to the CO enhancements observed in many of these events, though not in event 4 analyzed above. The analysis presented here is not affected by the likely presence of an admixture of biomass-burning emissions.

We now briefly discuss the relevance of these low-level events to North American pollution export and downwind impacts.

The altitude dependence of CO export from North America has been calculated by Li *et al.* [2005, Table 1] using GEOS-CHEM simulations for July of four years. On average, 38% of the total North American CO export (eastward across 70° N) occurs below 3 km altitude. Thus, the low-level events discussed above are not unusual. Events of this type also impact CO mixing ratios at low levels over Europe. For example, results from a 1-year FLEX-PART simulation of the transport of North American CO emissions indicate that peak impacts over Europe (at 5° E) of CO emissions emitted 10–12 days earlier occur below 3 km altitude (though significant North American CO is present from the surface to 7 km) [Stohl *et al.*, 2002a].

The relevance of low-level transport events to O₃ mixing ratios and fluxes over Europe has been addressed in several recent studies using the GEOS-CHEM model. Auvray and Bey [2005] and Li *et al.* [2002] both analyzed the impact of North American emissions on fluxes O₃ and resulting surface mixing ratios over Europe. Auvray and Bey [2005] found that horizontal advection

though the boundary layer (defined as the region below 600 hP) accounted for approximately one-third of the total summertime flux of O_3 produced from North American emissions into the European boundary layer for 1997, contributing approximately an additional 2.7 ppb of O_3 [Auvray and Bey, 2005, Figures 13 and 15, respectively]. *Li et al.* [2002] analyzed a 5-year (1993-1997) simulation and found that low-level (less than 3-km altitude) transport contributed significantly to North American impacts on European surface O_3 mixing ratios. On average, they found that during the summer, North American sources contributed 2-4 ppb of O_3 and 5-10 ppb during transport events, with low level flux into the boundary layer being as important as subsidence into the boundary layer. The events responsible for the largest enhancements of North American O_3 at European surface sites were the result of low-level westerly flow under conditions typified by a strong Icelandic Low located between Iceland and the British Isles. This situation is identical to that present in Figures 2.3g and h, which show the continued advection toward Europe of the northern portion of the large CO plume responsible for event 1. Several specific low-level events of this type are discussed by *Guerova et al.* [2006], who compare the resulting GEOS-CHEM-simulated O_3 enhancements over Europe to observations.

However, it has been noted that North American pollution is rarely seen at European surface sites [*e.g.*, *Derwent et al.*, 1998]. *Stohl et al.* [2002a] identified two possible reasons for this. Firstly, North American pollution reaches the European surface primarily south of the Pyrenees, where measurements sites are sparse, and secondly, the age of these plumes are generally older (more than 10 days old) and therefore hard to accurately trace back to sources on a case study basis. Additionally, North American emissions may also recirculate, turning anticyclonically around the ABH and head back towards the continent [Auvray and Bey, 2005]. This was fate of 14 of the 16

events in Table 2.1 based on HYSPLIT trajectories [Draxler and Rolph, 2003; Rolph, 2003] initiated at the station during each event; only event b and h head towards Europe, reaching the UK after 3 days and Spain after 3.5 days, respectively. The Azores, however, are located slightly south of the low level flow path that more often leads to Europe [e.g., Auvray and Bey, 2005, Figure 7] and so our results do not necessarily reflect the percentage of North American air that is transported in the lower troposphere, reaching Europe.

Many previous studies have also emphasized that long-range transport of O_3 exported into the MBL is unlikely and rarely observed due to the short O_3 lifetime in the MBL [e.g., Derwent *et al.*, 1998; Jaffe *et al.*, 2003; Price *et al.*, 2004; Auvray and Bey, 2005]. However, the low-level export events discussed here (events 1, 3, and 4) were exported below 3 km altitude, but significant O_3 enhancements were observed in events 3 and 4 (O_3 measurements are unavailable for event 1). Similar events in 2001 also produced characteristically significant O_3 increases [Honrath *et al.*, 2004]. However, the O_3 enhancements in both years were observed in the marine FT, not the MBL. This underscores the fact that BL structure over the oceans differs significantly from that over land, with the result that portions of plumes exported at low altitudes can be incorporated into the marine lower FT. Near the coast, exported continental plumes produce pollutant layers in a statically stable atmosphere with weak vertical mixing [Angevine *et al.*, 2004]. Over time, as the continental air-mass travels over the ocean, an internal boundary layer develops [Skylvingstad *et al.*, 2005], ultimately becoming a steady-state mixed layer [Smedman *et al.*, 1997]—*i.e.*, the MBL—and leaving a portion of the exported plume in what is now the lower FT. Alternatively, weak uplift as described for events 3 and 4 may loft the pollution directly into the lower FT. Lower FT transport of this type provides a mechanism for the long-range transport of continental pollution without the dehydration, dilution, scavenging, and expansion generally

associated with uplift to high altitudes, yet also without the greatly enhanced loss rates present in the MBL.

Finally, the importance of low-level transport to the Azores and the lower FT of the Central North Atlantic is a significant result. Midlatitude cyclones are responsible for most of the pollution outflow from North American to the North Atlantic as they track eastward across the U.S. [*Merrill and Moody*, 1996; *Li et al.*, 2005]. Specifically, WCB transport is a primary mechanism for the export of polluted air from the continental boundary layer and subsequent intercontinental transport [*Cooper and Parrish*, 2004]. However, only 2 events (event 2 of the case studies and event 1 from Table 2.1) out of 16 observed at the PICO-NARE station experienced lofting to the middle troposphere that is typical of WCB events responsible for intercontinental pollution transport. This analysis indicates that WCBs are not dominant mechanisms for transport to the Central North Atlantic lower FT during summer. Instead, advection of pollution layers at low levels, but decoupled from the MBL is responsible for the majority of transport into the region during summertime.

2.5 Summary and Conclusions

We have used the Lagrangian particle dispersion model FLEXPART to simulate the transport of North American CO emissions to the PICO-NARE station in the central North Atlantic lower free troposphere during July 2003. FLEXPART adequately captured the occurrence of CO transport events: North American anthropogenic CO enhancements of at least 10 ppbv were simulated during 15 of 16 events of enhanced CO actually observed, but only in three events not apparent in the observations. These observed events were quite frequent during July 2003, occurring on 68% of the days for which CO measurements are available. The evolving meteorological situations responsi-

ble for four of these events were analyzed in detail.

Two mechanisms were responsible for the export of pollution from the North American boundary layer that led to these events: direct advection from the continental boundary layer eastward over the Atlantic Ocean, and uplift ahead of cold fronts. In most simulated events, export occurred at low altitudes (below ~ 3 km). Three case studies of low-level events were presented, including two events exported in the same weak warm conveyor belt event. In these three cases, and in a majority of all the events simulated by FLEXPART, the subsequent transport to the Azores occurred in the lower troposphere and was the result of flow channelled between the Azores/Bermuda High and one or more lows passing to the north. The pathway followed by the pollutant plumes in these events is therefore sensitive to the location of these features, and situations only moderately different than those analyzed here can result in similar events reaching Europe.

Although the result of low-level export, these events were nevertheless observed in the free troposphere over the Azores, and O_3 enhancements were observed coincident with CO in most cases. Since low-level export of North American emissions accounts for a significant portion of the total summertime CO export [*Li et al.*, 2005], we suggest that low-level continental export leading to transport in the lower troposphere, but above the marine boundary layer, may provide an effective mechanism for long-range impacts of anthropogenic emissions on lower tropospheric O_3 in downwind regions.

A fourth event described in detail was the result of uplift to above 6 km in the warm conveyor belt associated with a cyclone passing to the north. The emissions that reached the Azores were captured in the mid-tropospheric circulation around the center of the low for two days, before becoming entrained in the same cyclone’s dry airstream. The event was observed when the dry airstream reached the Azores following the passage of the front one day later.

Chapter 3

Lagrangian tracking of pollution plumes using gridded model output[†]

3.1 Introduction

The transport experienced by a plume of emissions can have a significant influence on its chemical composition. Dry deposition, which is an important removal mechanism for many trace gases, occurs in the boundary layer (BL). Significant wet deposition is often associated with strong uplift from the BL into the free troposphere (FT) (e.g., *Stohl et al.*, 2002c). For example, soluble species, such as HNO_3 , can be removed during this uplift. Once in the

[†]This chapter is based on Owen, R. C., and R. E. Honrath, Technical note: a new method for the Lagrangian tracking of pollution plumes from source to receptor using gridded model output *Atmospheric Chemistry and Physics*, 9, 2,557-2,595, 2009. Authors retain copyright of ACP articles. Copyright permission details given in Appendix 3.

FT, however, the chemical composition of an air mass is more dependent on photochemistry and mixing (e.g., *Methven et al.*, 2003). Thus, knowing the amount of time an air mass spends in the BL, the timing and location of uplift, the time spent in the FT, and the relative amounts of mixing during these processes is essential to a complete understanding of the chemical transformations occurring in an air mass. As a result, determining these transport characteristics for a plume of emissions as it travels from its source to a downwind sample location has been an important part of many measurement efforts (e.g., *Rex et al.*, 1998; *Stohl and Trickl*, 1999; *Trickl et al.*, 2003).

The atmospheric transport pathway through which emissions traveled to a downwind receptor is often deduced with Lagrangian models, either trajectories or Lagrangian particle dispersion models (LPDMs). Trajectories remain popular because they are easy to use, but they are limited by their inability to describe the deformation of an air mass and the concentration gradients of chemical trace substances in the atmosphere [*Stohl et al.*, 2002b; *Methven et al.*, 2006]. LPDMs are superior because they address both these issues (e.g., *Han et al.*, 2005), but they also have shortcomings. Their output is more complex than that of trajectory models and much of the Lagrangian information is lost in the process of calculating concentrations on a Eulerian-type output grid. While some work has been done to simplify the LPDM output (e.g., *Stohl et al.*, 2002b), there remains a need for new products to succinctly describe LPDM output. Additionally, there are no methods available to retrieve the Lagrangian information that is lost when the output is gridded, short of saving particle trajectories. While particle trajectories provide useful information, they are typically not saved during LPDM simulations because they increase the time it takes to run and process a simulation and require large amounts of storage and memory. Particle trajectories also add a level of complexity in interpreting the output and are generally used only in advanced

LPDM studies (e.g., *Stohl et al.*, 2004a).

Traditionally, studies that use LPDMs to perform a detailed analysis of the transport of emissions to a particular receptor use both forward simulations (simulations of atmospheric concentrations resulting from an emissions field) and backward simulations (simulations of the upwind transport of air ultimately reaching a receptor), but present them as individual products. For example, *Stohl et al.* [2003] used forward simulations to understand the large scale transport from North America to aircraft-based sample locations over the North Atlantic and Europe, while backward simulations were used to determine the age distribution of the trace substance in the receptor cells, to determine the specific source regions contributing to the trace substance enhancements in the receptor cells, and to determine the transport pathway to the receptor. They assumed that the backward plume matched the pathway taken by emissions, which was reasonable only because this particular transport experienced very little deformation. *Owen et al.* [2006] analyzed the forward and backward products together, by presenting snapshots of the two simulations side-by-side, but detailed analysis of transport was limited to only those backward simulations that experienced little deformation between the source and receptor. Despite these applications, there remains a disjunction between the information provided by forward and backward LPDM simulations.

In this paper, we present and evaluate a new method, the product of which we call a folded retroplume. The folded retroplume addresses two of the shortcomings of the LPDM by simplifying the LPDM output and allowing the retrieval of some of the Lagrangian information that is lost in the process of calculating gridded (Eulerian) output fields. The purpose of the folded retroplume is to provide a way to efficiently and accurately determine the transport pathway of emissions to a receptor, highlighting only those emissions

that arrive in the receptor cell at the time of interest, using standard gridded output fields from an LPDM. As we show below, this can be accomplished by convolving the standard output from a forward model simulation with that from a backward model simulation, bringing the information from the forward and backward models together in such a way that even complex transport scenarios can be analyzed. When used in this way, the forward model describes the amount of a trace substance in the atmosphere, while the backward model describes how much of this trace substance will arrive in the receptor cell at a given time. The folded retroplume is easier to use and more accurate than using standard gridded LPDM products alone. Additionally, the method is superior to similar methods available with trajectories, as it retains the advantages of LPDMs, e.g., the ability to describe dispersion and to purvey information about the relative concentration of the trace substance along the transport pathway. Since the method requires an agreement in the transport described by the two model simulations, it also allows for an assessment of the quality and reversibility of the simulation. We also introduce similar uses of the backward model with alternate Eulerian fields that describe the state of the atmosphere (e.g., output from a chemical transport model) to determine physical and chemical properties of an air mass as it travels toward a receptor.

Below, we provide a method overview (Sect. 3.2), evaluation (Sect. 3.3), and example (Sect. 3.4) that are based on the LPDM FLEXPART [Stohl *et al.*, 2005], one of the more popular LPDMs in use today. Although the presentation is somewhat specific to this model, the method should be valid for any LPDM with an appropriately employed backward mode, because all LPDMs have the property that they are essentially self-adjoint, i.e., the backward mode only requires the reversal of the direction of advection to give the transport sensitivity for a receptor cell [Seibert and Frank, 2004].

3.2 Method overview

We begin with a brief outline of the formulation of the model output and the folded retroplume, followed by a simple case that illustrates the folded retroplume. We rely heavily on the model theory presented by *Stohl et al.* [2005] and *Seibert and Frank* [2004] and refer the reader to these sources for more detailed reviews of LPDM theory and technical descriptions of LPDM operational details. We also recommend *Flesch et al.* [1995] and *Lin et al.* [2003] for additional information on backward LPDM modeling and *Errico* [1997] and for additional background on general adjoint model theory.

3.2.1 Formulation of the model output and folded retroplume

In this section, we review the calculation of the gridded model output, starting with the forward mode, and provide several formulations involving the folded retroplume. The calculations presented in this section are limited to instantaneous model output. (The use of averaged model output can complicate the interpretation of folded retroplumes and is thus discussed in Sect. 3.3.3.)

3.2.1.1 Standard output from the forward mode

In the forward mode, particles are released at the source and then transported forward in time, according to the mean and turbulent wind components [*Stohl et al.*, 2005]. The mass of each particle at the time of emission is based on the strength of the source. Concentrations are calculated by summing the mass of all the particles that reside in each grid cell. We first consider only a puff of emissions and focus on one downwind grid cell (j) at a single model time (t). The instantaneous gridded concentration (c) from a puff of emissions released

at time t_0 is thus:

$$c_{j,t} = \frac{1}{V_j} \sum_{V_j} m_{j,t_0} \cdot p_{j,t} \cdot f_j, \quad (3.1)$$

where V_j is the volume of the cell and the summation is over all particles that reside in V_j at time t . m_{j,t_0} is the initial mass of the particle, f_j is the sampling kernel, which can be used to distribute the mass of the particle across multiple grid cells, and $p_{j,t}$ is the transmission function, which describes the percentage of the particle mass remaining from removal processes (see *Stohl et al.*, 2005 and *Seibert and Frank*, 2004 for a more complete description of these terms). In order to calculate the mass mixing ratio, the concentration is first divided by the local air density (from the meteorological data). The volume term from the concentration cancels with the volume term from the local air density, leaving the mass of air in the cell ($m_{j,\text{air}}$) and the summation of the mass of the particles in the cell. The volume mixing ratio (χ) is obtained by multiplying this value by the ratio of the mean molecular mass of air (M_{air}), to that of the trace substance being modeled (M_{tr}), giving the volume mixing ratio:

$$\chi_{j,t} = \left(\frac{M_{\text{air}}}{M_{tr}} \right) \left(\frac{1}{m_{j,\text{air}}} \right) \sum_{V_j} m_{j,t_0} \cdot p_{j,t} \cdot f_j. \quad (3.2)$$

Mixing ratios are saved at each time for each grid cell in the model domain, giving a 3-dimensional matrix (χ_t) of the volume mixing ratios for all j (i.e., each x , y , and z component).

3.2.1.2 Standard output from the backward mode

In the backward mode, particles are initiated in a single receptor volume (j') over a short interval (t' , the arrival time) and transported backward in time by reversing the direction of the mean wind. The mass of each particle is normalized by the total mass released in the receptor (m_{tot}), giving each particle units of mixing ratio, such that each particle represents one part of

the air in the receptor at the release (i.e., arrival) time and the distribution of the particles indicates the location of the receptor air at each upwind time. The backward output is gridded by summing these mixing ratios in each cell, giving the sensitivity (S) of the receptor to the mass present in the upwind cell:

$$S_{j,t,(j',t')} = \sum_{V_j} \frac{m_{j,t'}}{m_{\text{tot}}} \cdot p_{j,t} \cdot f_j, \quad (3.3)$$

Again, the output is saved at each time for each grid cell, giving a 3-dimensional matrix ($\mathbf{S}_{\mathbf{t},(j',t')}$) of the sensitivity for all j . The output of the backward mode is referred to as the sensitivity plume, or the retroplume.

3.2.1.3 The folded retroplume – combining model output to determine the source-to-receptor transport pathway

The folded retroplume at time t is the Hadamard (or entry-wise) product of the mixing ratio matrix ($\chi_{\mathbf{t}}$) from the forward mode and the sensitivity matrix ($\mathbf{S}_{\mathbf{t},(j',t')}$) from the backward mode. In terms of an individual cell, the mixing ratio (Eq. 3.2) indicates the amount of emitted trace substance in the given cell, and the sensitivity (Eq. 3.3) indicates the amount of air in the cell that will be transported to the receptor. By multiplying the two values, we can determine the amount of the trace substance in the cell that will eventually arrive at the receptor (j') at the arrival time (t'). Note the units for this operation. We begin with the volume mixing ratio, with units of parts of trace substance per parts of air in the cell. This is multiplied by the sensitivity, with the units of parts of air in the cell per parts of air in the receptor. The resulting product calculated for a specific cell j has units of volume mixing ratio, and indicates the portion of the mixing ratio in the receptor at t' (the sensitivity plume arrival time) resulting from the transport of trace substance through cell j at time t . That is, the units are parts of

trace substance in the cell per part of air at the receptor. As this mixing ratio results from only a part of the sensitivity field (the individual upwind cell considered here), we call the product the partial mixing ratio (PMR):

$$\text{PMR}_{j,t,(j',t')} = S_{j,t,(j',t')} \cdot \chi_{j,t}, \quad (3.4)$$

where t is the model time. The PMR will clearly be small or zero when there is either little of the trace substance in a cell or small sensitivity. Conversely, if there is a significant amount of trace substance in a cell and a large sensitivity, then the PMR will also be large, indicating the location of trace substance that travels from the source to the receptor. The 3-dimensional matrix $\mathbf{PMR}_{\mathbf{t},(\mathbf{j}',\mathbf{t'})}$ indicates the distribution at time t of the trace substance that will ultimately arrive at the receptor at time t' , while the matrix $\mathbf{PMR}_{(\mathbf{j}',\mathbf{t'})}$ at multiple times shows the transport pathway of the trace substance between the source and receptor.

Up to this point, we have only considered a puff of emissions in the forward model, which is not the typical model situation. Normally, emissions are continuously released into the forward simulation and each particle is carried in the model for a set number of days and then dropped. Thus, the mixing ratio from the forward model ($\chi_{j,t}$) consists of particles released over a range of times and can be divided into age classes, according to the length of time the particles have been in the model (the age of the particles). If the age of the trace substance is not taken into account when computing the PMR then the folded retroplume calculation will include particles that would be dropped from the forward simulation before they would be transported to the receptor. To avoid this, the age of forward model trace substance in Eq. (3.4) ($\chi_{j,t}$) must be less than the time difference between the release time in the backward model (that is, the arrival time) and the sample time (t). If we have a forward simulation that carries particles for A_f days and a backward

simulation with an arrival time of t' , then the PMR at some intermediate time (t) should be:

$$\text{PMR}_{j,t,(j',t')} = S_{j,t,(j',t')} \cdot \sum_{i=0}^{A_f-(t'-t)} \chi_{j,t,i}. \quad (3.5)$$

where i indicates the available age classes in days from the forward model.

The PMRs at any upwind time may be summed over the model domain to determine the mixing ratio from all (appropriately aged) emissions that are present in the model at that time. If no more emissions are added to the atmosphere between that time and arrival time, then this sum would be equal to the mixing ratio in the receptor (j') at the arrival time (t'). Thus, we call this sum the upwind mixing ratio (UMR):

$$\text{UMR}_{t,(j',t')} = \sum_j \text{PMR}_{j,t,(j',t')}. \quad (3.6)$$

The UMR is equivalent to a sensitivity-weighted average of the upwind mixing ratio field and can increase or decrease, depending on the relative rates of emission and removal processes. For example, if emissions are added to the atmosphere between time steps and no removal processes are considered, the change in the UMR from time t to $t+1$ should be

$$\text{UMR}_{t+1,(j',t')} = \text{UMR}_{t,(j',t')} + \sum_j S_{j,t,(j',t')} \cdot E_{j,t}, \quad (3.7)$$

where $E_{j,t}$ are the emissions released into the model at time t and $\sum_j S_{j,t,(j',t')} \cdot E_{j,t}$ is the so called source contribution [Stohl *et al.*, 2003]. However, if no removal processes are considered and if no emissions are added to the atmosphere after time t (or if no emissions are added in areas with sensitivity – the region where the plume is located), then the UMR should remain constant.

The UMRs therefore provide a means to evaluate the evolution of the mixing ratio of the receptor air during transport. For instance, the timing and location of wet removal could be determined by comparing the UMRs

from two folded retroplumes, one computed from forward and backward simulations with no wet removal and one computed from forward and backward simulations that include wet removal. Section 3.5 will discuss other potential applications using the UMRs derived from folding a backward simulation with mixing ratio fields from alternate sources.

3.2.2 Illustrative case

Here we present the application of our method to a simple case in order to illustrate the folded retroplume. The case is based on a puff of CO emissions released into the forward model from the Boston area, into the box bounded by 41–43° N latitude and 73–75° W longitude, from the surface up to a height of 250 m a.s.l. Emissions were released over a 1-h period, from 15:00–16:00 UTC on 14 May 2005 and were based on the EDGAR Fast Track 1999 inventory [Olivier *et al.*, 1996]. The backward simulation was initiated at the Pico Mountain observatory, located on the Azores Islands in the central north Atlantic Ocean, into the box bounded by 38.5–39.0° N latitude 28.5–28.0° W longitude, from an altitude of 2000–2250 m a.s.l. Particles for the backward simulation were also released over a 1-h period from 00:30–01:30 UTC on 19 May 2005.

FLEXPART version 6.2 was used, driven with data from the European Centre for Medium Range Weather Forecasts (ECMWF) [ECMWF, 2005] with 1°×1° horizontal resolution, 60 vertical levels and a temporal resolution of 3 h, using meteorological analyses at 00:00, 06:00, 12:00 and 18:00 UTC and ECMWF 3-h forecasts at intermediate times (03:00, 09:00, 15:00 and 21:00 UTC). The output was saved with a grid size of 0.5°×0.5° in the horizontal and 250 m in the vertical, from 0–7000 m a.s.l. The sampling kernel was turned off and instantaneous fields were saved (see Sect. 3.3.3 for a de-

scription of these model settings and more details on their impact on the folded retroplume). 500 000 particles were used for the forward simulation and 2 000 particles were used for the backward simulation, resulting in a total of 670 forward particles and 634 backward particles that successfully travel between the source and receptor cells.

Figure 3.1a and b shows the plan view and longitude-height cross section of the CO plume 1.5 days after the forward-model puff release. Figure 3.1c and d shows plan view and longitude-height cross section of the sensitivity plume 3 days upwind of the release at the receptor (and at the same time as shown in Fig. 3.1a and b). (Note that throughout the paper the terms CO plume and sensitivity plume refer to the forward and backward model simulations, respectively.) Figure 3.1e and f shows the plan view and longitude-height cross section of the folded retroplume, derived from folding the mixing ratio and sensitivity fields shown in a–d, along with the contours showing the limits of the forward (blue) and backward (magenta) plumes from panels a–d. Note that throughout the paper we color any product derived from the forward model blue, from the backward model red and magenta, and from the folded retroplume green.

The folded retroplume clearly indicates the portions of the two plumes that successfully travel between the source and receptor. The folded retroplume also indicates the relative concentrations of the receptor-bound trace substance. A comparison of the folded retroplume with the forward CO and backward sensitivity contours shows that simply superimposing the forward and backward plumes can be misleading. In this case, the overlap of the two contour lines roughly define the folded retroplume in the vertical view (f). However, this is not the case for the plan view (e), as the folded retroplume only occupies a portion of the overlapping contours. This apparent inconsistency is the result of viewing vertically integrated fields. The trace substance

plume and sensitivity field, while in the same vertical plane, are not actually colocated vertically. In Sect. 3.4, we provide an expanded comparison of the folded retroplume with the standard LPDM products in a sample analysis.

3.3 Method evaluation

3.3.1 Approach and methods for the detailed evaluation

The primary purpose of the evaluation is to determine how well the folded retroplume reconstructs the pathway of emissions from the source to the receptor. Here, we use particle trajectories from the LPDM model runs to evaluate the accuracy of the folded retroplume pathway. The secondary purpose of the evaluation is to examine the behavior of the UMRs along the transport pathway. As discussed above, the UMRs should be constant if no emissions are added to the forward model. Deviations in the UMRs, which indicate disagreement between the transport described by the forward and backward models, will also be investigated using particle trajectories. Some degree of disagreement is expected, as a result of the random components in the models (turbulence and convection). Finally, we will relate the behavior of the UMRs to the accuracy of the folded retroplume pathway.

This evaluation used the same model simulations presented in Sect. 3.2.2, which focused on the transport of anthropogenic CO emissions from a source region near Boston, MA on the US east coast to a receptor cell located around the Pico Mountain observatory in the Azores Islands in the central North Atlantic. The two are located approximately 3620 km from each other and the transport time from source to receptor was 4.4 days. This transport time and distance should be sufficient to allow for any deviations from the expected

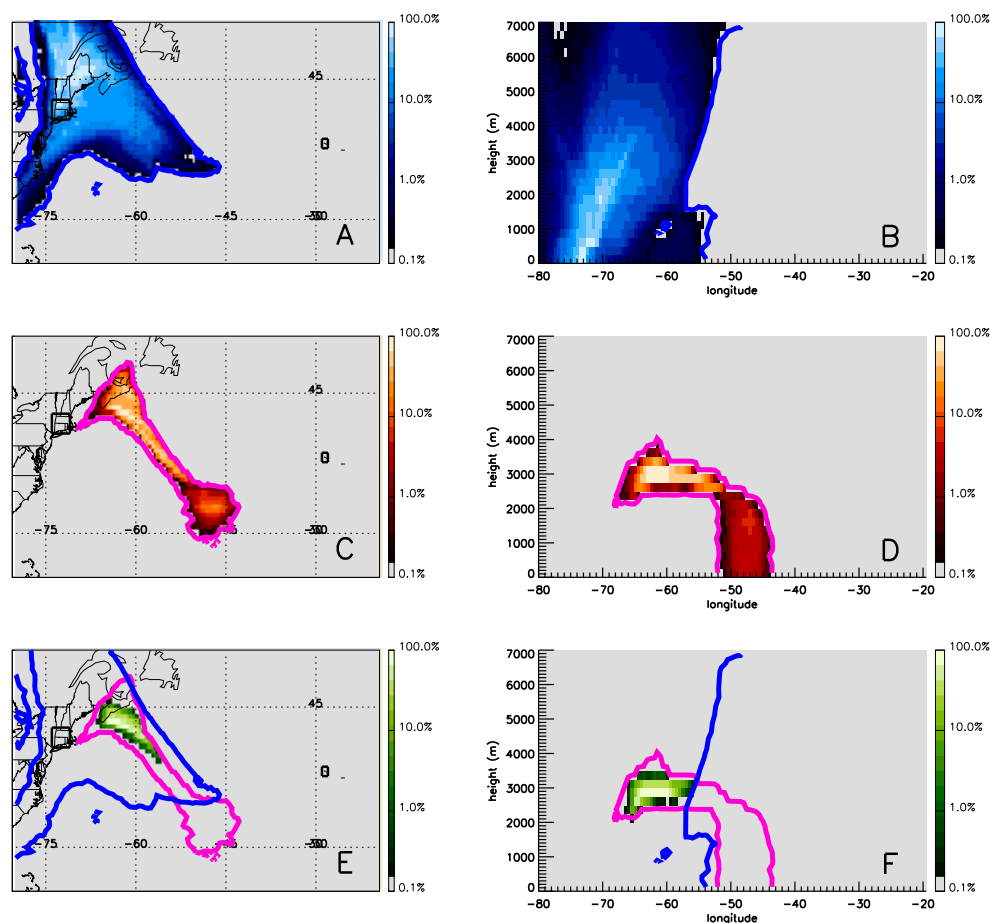


Figure 3.1 The plan views (left column) and longitude-height cross sections (right column) of snapshots of the vertically integrated (left column) and horizontally integrated (right column) CO concentrations from a forward model simulation 1.5 days after release (a and b), the sensitivity field from a backward simulation 3 days upwind of the receptor (c and d), and the folded retroplume, or results of folding the concentration and sensitivity fields from (a)–(d) (e and f). The colors for the plumes are scaled according to the maximum value in each panel. Contours indicate the limits of the forward (blue) and backward (magenta) plumes and are drawn at 1% of the maximum value for each plot. The source and receptor boxes are outlined in black in (a), (c), and (e)..

pathway and UMR to become apparent. While the source region was chosen arbitrarily, the timing of the transport scenario was selected by first running a forward simulation with continuous emissions, from April to June 2005. We then selected one of the periods with the largest CO mixing ratio in the receptor cell for further inspection, with no prior knowledge of the transport scenario.

In this evaluation, we will discuss two types of particles, termed positive particles and negative particles based on whether or not they travel from the source to receptor over the period analyzed. From the forward simulation, positive particles are the trace substance particles that arrive in the receptor cell during the release of the backward plume. From the backward simulation, positive particles are the sensitivity particles that arrive in the source cell at the release time for the forward simulation. Negative particles are all other particles, including particles that do not travel from the source to the receptor as well as particles that successfully travel between the source and receptor cells, but not within the time frame of interest.

Note that at most times and in most grid cells, there will be both positive and negative particles from both model directions. Since dispersion causes increasing separation over time between particles that are initially near one another, the amount of dispersion a plume has experienced affects the relative number of concurrent positive and negative particles. As the two plumes are tracked toward the receptor, the forward plume will disperse and the backward plume will coalesce. Thus, near the source and close to the release time, many negative forward particles should be located along with positive forward particles, as the forward trace substance plume has experienced relatively little dispersion. In contrast, near the receptor and release time, only a few negative forward particles should be colocated with the positive forward particles, as the forward trace substance plume should be highly dispersed. For

the backward simulation, there should be few concurrent positive and negative backward particles near the source and many concurrent positive and negative particles near the receptor. Theoretically, no location should ever contain only negative particles from both model directions, nor should positive particles from one model direction be located in a cell without positive particles from the other model direction. These two situations indicate differences in the transport described by the two model simulations. In practice, however, this can occur, due to the random model components, transport errors, or irreversible transport.

3.3.2 Detailed evaluation results

3.3.2.1 Detailed evaluation of the folded retroplume pathway

The time-integrated results from the forward and backward simulations used for the evaluation are shown in Fig. 3.2. Figure 3.2a, which shows the plan view of the vertically integrated CO concentration field, indicates that the bulk of the CO emissions are transported northward. These emissions move out of the plot window; later, however, some of these emissions travel southward, toward the receptor (present as the dark plume stretching south-east from the northern edge of the plot window). A significant portion of the CO plume also moves east and southeast, stretching from the US east coast, across the Atlantic, to the receptor. Figure 3.2b, which shows the time-height cross section of the horizontally integrated plots of the CO concentration field, indicates that the bulk of the CO is transported to higher altitudes during the first few days, though CO is distributed throughout all levels of the atmosphere during the last 3 days of transport. The plan view (Fig. 3.2c) and the time-height cross section (Fig. 3.2d) of the sensitivity plume indicate a number of pathways (i.e., areas of sensitivity) for air traveling to the receptor.

The regions of highest sensitivity are in a fairly compact pathway starting from just off the east coast of Nova Scotia, where the air converged, coming equally from the north and the south (from the emissions region). There is a secondary region of sensitivity that also originates near the source region and travels over the Atlantic slightly farther south than the primary sensitivity region, converging with the primary transport pathway west-southwest of the receptor.

Near the receptor, the horizontal transport pathway of the emissions can be guessed from Fig. 3.2 by comparing the sensitivity with the CO concentrations, as there is only a small region of overlap of the sensitivity (Fig. 3.2c) and CO fields (Fig. 3.2a). However, near the source and in the intermediate transport, over the Atlantic, the pathway that emissions travel to the receptor is unclear from these plots alone. Both the CO and sensitivity occupy a large area, both horizontally and vertically. Thus, even for this simplified case, with only a puff of emissions into the forward model, determining the exact pathway (or pathways) taken by the emissions as they travel to the receptor is not possible from the plan view and cross sections plots alone. One would need to view snapshots (i.e., the distribution of the plume at a single time, as opposed to the time-integrated view shown in the figure) of the two plumes in order to do that. However, even when viewing multiple snapshots of the simulations, diagnosing the correct transport pathway can be difficult or impossible, as discussed in Sect. 3.2.2.

In contrast to the gridded output from the LPDM, the particle trajectories and the folded retroplume offer a clear view of the transport pathway between source and receptor. Figure 3.3 shows the plan view (a and c) and the time-height cross section (b and d) of the positive particles from the forward (a and b) and backward (c and d) model simulations. Figure 3.3e and f shows the plan view and the time-height cross section, respectively, of the folded

retroplume pathway obtained from folding the two model simulations using Eq. (3.4).

In terms of the core transport pathway described by the three products, there was good overall agreement in both the horizontal and vertical pathways. All indicated lofting of emissions to 1–2 km in a daytime BL during the first few hours of transport. The emissions remained at this altitude, after the transition from a deeper continental daytime BL to a shallow nighttime marine BL left them located in the FT. Once in the FT, the emissions experienced slower ascent to about 3 km for approximately 1 day, where they remained for another day. Finally, during the last 2 days of transport, the emissions experienced a gradual descent from 3 km to the receptor at 2–2.25 km. The horizontal pathway shows that the emissions traveled northward along the coast to Nova Scotia, then traveled southeast before heading northeast again, toward the receptor. The common transport described by all three products indicates that the folded retroplume successfully identifies the large-scale transport pathway between the source and receptor.

A comparison of the positive particle and folded retroplume pathways reveals three interesting features outside of the core transport pathway. Two of these features are regions where, due to the random components of the model, the pathways of the forward and backward positive particles differ. One such situation occurs during the initial day of transport (15–16 May) during the ascent from 1 km to 3 km (above the core transport pathway shown in panels b and d). The forward and backward maximum particle locations indicated in panel f encompass this region, indicating it is part of the source-to-receptor transport pathway. However, the smaller number of particle trajectories indicate that the probability of transport through this region is very low. The folded retroplume correctly identifies this low-probability region with a fairly small PMR, colored with darker greens and black. The second region where

there is a difference between the forward and backward particle trajectories occurs during the last half of 16 May, when two forward positive particles stray below the core transport region (panel b). The very small number of trajectories from the forward model here indicates that this region is not part of the primary source-to-receptor transport pathway. The third feature is the presence of a few cells with a non-zero PMR that are entirely outside the limits of the positive particle pathway (i.e., a false positive PMR). These cells are all adjacent to the positive particle pathway and contain only a small PMR (generally less than 1% of the maximum PMR and roughly 1% of the UMR). We show below that a blurring of the transport pathway up to one grid cell in size is the result of the use of gridded data. In summary, this evaluation indicates that the folded retroplume correctly identified the pathway of all but a small fraction (in this case, .3%) of the source-to-receptor pathway, with errors of up to about 1 grid cell in location.

3.3.2.2 Detailed evaluation of the UMRs

The UMRs are another important component of the folded retroplume as they can be used to estimate the timing and rate of emissions into or removal of trace substance from the plume. The behavior of the UMRs can also be used to determine the degree of agreement in the transport as described by the forward and backward model. As discussed above, the UMR should be constant if no emissions are added to the model between time steps. Since we use a puff of emissions for this evaluation, the UMR should be constant at all times between release and arrival at the receptor. Thus, deviations from the expected UMR (i.e., a non-constant UMR) indicate when there are differences in the transport described by the forward and backward model simulations.

Figure 3.4 shows the UMRs at each upwind time for the folded retroplume. The left-hand axis indicates the absolute UMRs (pptv CO), and the right-

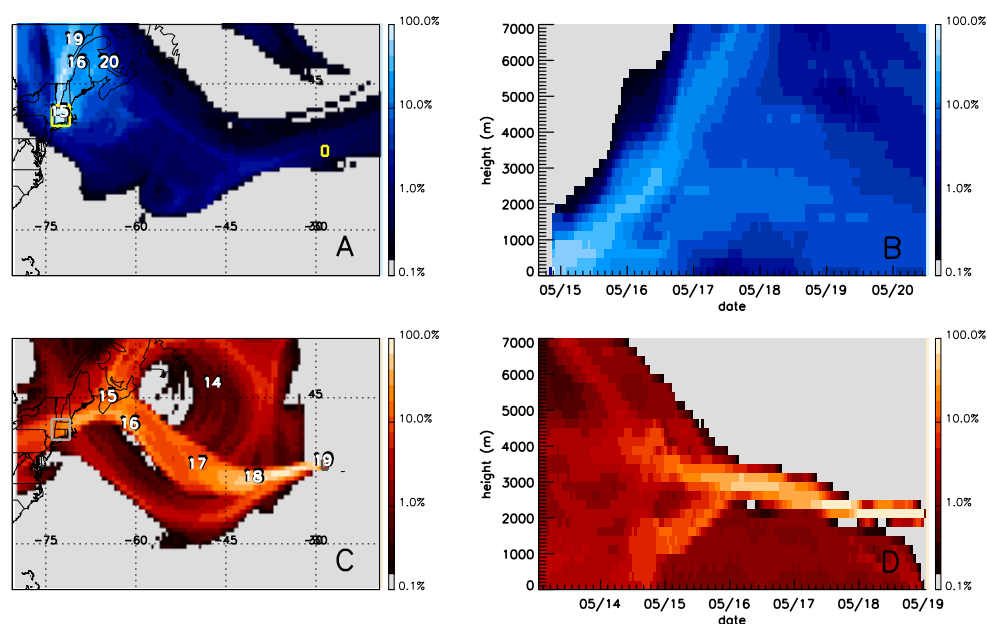


Figure 3.2 The time integrated results from the forward and backward model simulations for the evaluation case. The plan view and time-height cross section of the CO plume from the forward simulation are shown in blue in (a) and (b), respectively. The plan view and time-height cross section of the sensitivity plume are shown in red in (c) and (d), respectively. The source and receptor volumes are outlined in yellow and gray in (a) and (c), respectively. The white numerals show the average location of the CO and sensitivity plumes at 00:00 UTC on the day of month indicated by the numbers..

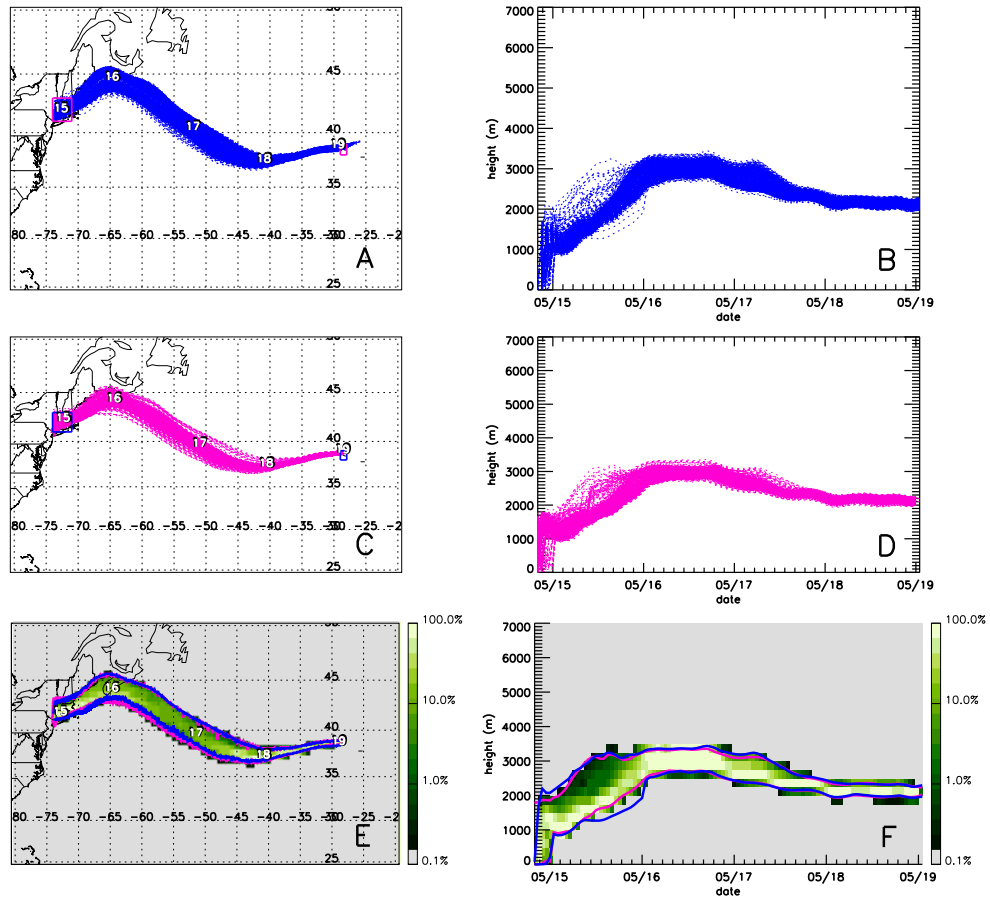


Figure 3.3 The plan view and time-height cross section of the positive particle trajectories from the forward simulation are shown in blue in (a) and (b), respectively. The plan view and time-height cross section of the positive particle trajectories from the backward simulation are shown in magenta in (c) and (d), respectively. The plan view and time-height cross section of the folded retroplume are shown in green in (e) and (f), respectively. The solid blue and magenta lines in (e) indicate the location of the maximum and minimum positive particle latitudes at each longitude from the forward and backward model simulations, respectively. Similarly, the solid blue and magenta lines in (f) indicate the maximum and minimum positive particle altitudes at each day. The source and receptor volumes are outlined in magenta and blue in (a) and (c), respectively. The white numerals show the average location of the positive particles and the folded retroplume at 00:00 UTC on the day of month indicated by the numbers..

hand axis indicates the UMRs normalized by dividing by the last UMR before arrival. (Any of the UMRs could have been used for normalization, but we chose this value because there has been little deformation of the air mass between the sample time and the arrival time in the receptor due to the calm meteorological scenario. As a result, it should closely represent the UMR that would be calculated if the forward simulation could be sampled while in the receptor cell.)

There are clearly significant variations in the UMR values, approximately 35% in the negative direction and a little more than 45% in the positive direction. We consider two possible causes for these deviations. These deviations can be the result of irreversibility, which would indicate that the forward and backward simulations are not simulating the same transport. Irreversible transport can be caused by one or more of a number of factors, including the violation of the well-mixed criterion [Thompson, 1987], not maintaining a consistent representation of the mass of particles as air density changes [Lin *et al.*, 2003], and errors induced by the interpolation of meteorology between grid points [Stohl, 1998]. The two positive particles that stray from the primary transport pathway and do not coincide with positive backward particles may indicate that a small portion of this event is irreversible. However, the overall agreement between the forward and backward positive particle trajectories indicate that the primary transport pathway described for this event is reversible. Another source of these deviations could be the presence of sub-grid gradients in the CO or sensitivity field that are lost when gridded fields are calculated. In order to investigate this potential, we conducted detailed inspections of the CO concentrations, sensitivity, and PMR fields and the distributions of the positive and negative particles in the vicinity of the positive particles at several times, marked by the dotted vertical lines in Fig. 3.4.

This investigation determined that the low UMRs resulted from minor

displacements (less than the size of a grid cell) between the groups of forward and backward positive particles, specifically in regions with a high mixing ratio or sensitivity gradient. For the transport scenario examined here, the forward CO plume near the receptor (time period 4 in Fig. 3.4) is a thin filament, on the order of 2–4 grid cells wide, and the positive forward particles are at the edge of this filament of CO (only the part of the CO plume that contains positive particles actually passes through the receptor cell). Thus, when the positive backward particles are displaced slightly from the positive forward particles, they are in a region with little CO, resulting in a very small PMR in those cells and a negative bias in the calculated UMR. A similar case can be found in period 3.

Around period 2, the roles of the sensitivity and CO plume begin to change. Around this time, the sensitivity plume splits (as it is followed backward in time), with one portion heading northeast and another portion (which contains the positive particles) heading southwest, towards the receptor. Meanwhile, the CO plume (as it is tracked forward in time) is also in the process of splitting in two. One portion is the filament that eventually travels to the receptor, and the other is the larger portion that travels northeast from the source. The positive forward and backward particles were still slightly displaced from one another. However, they were no longer located at the edge of their respective plumes, and thus no longer in a region of a high sensitivity or CO gradient. As a result of these conditions, the UMRs around this time are closer to the expected value.

Closer to the source region, however, a different situation results in UMRs with a positive bias. First, there is again a large CO gradient (as in the other periods). However, now there is a relatively large concentration of negative forward particles in these cells, because the plume has not dispersed much yet. Second, the sensitivity plume was more dispersed. The positive

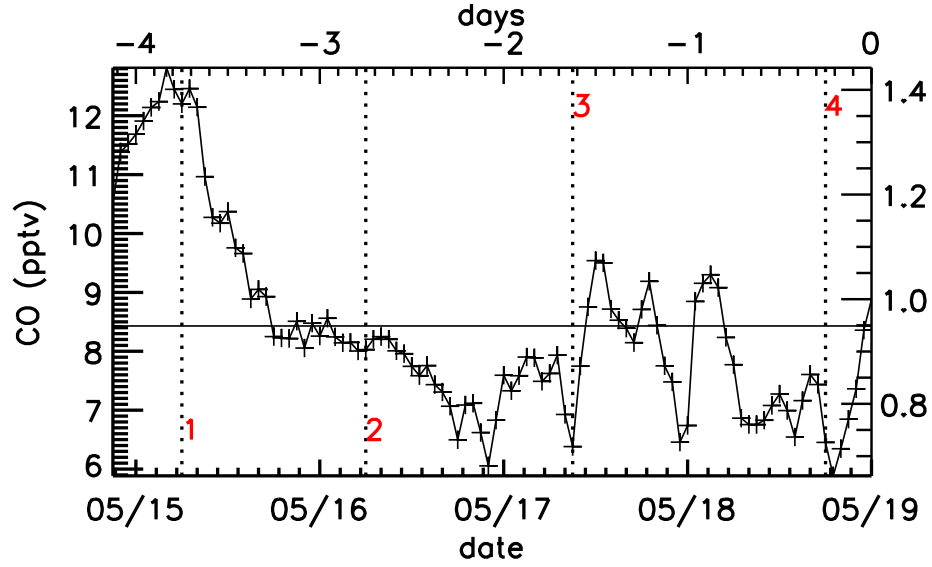


Figure 3.4 Upwind mixing ratios (UMRs) at each transport time between departure at the source and arrival at the receptor for the evaluation simulation. The bottom axis indicates the date while the top axis indicates the upwind day number. The left-hand axis indicates absolute UMRs and the right-hand axis indicates the relative UMRs, normalized by the last UMR. Numbers and vertical dotted lines indicate periods discussed in detail in the text..

backward particles reside in more cells and are distributed more uniformly than the positive forward particles. As a result of these two factors, the positive backward particles are located in cells with a large number of negative forward particles. The sensitivity plume is therefore combined with significantly higher CO, resulting in a higher UMR. As the trace substance will be ubiquitous very near source regions in most model scenarios, this situation may occur frequently when UMRs are calculated close to the source region.

3.3.3 The impact of various model settings on the folded retroplume

Many model settings can affect how well transport is described, affecting the correlation between model simulations. Additionally, the way in which the output is saved can affect the number of positive and negative particles that are identified as being colocated. As the evaluation above demonstrated, these factors can in turn significantly affect the folded retroplume UMRs. We have evaluated the impact of the model settings listed in Table 3.1 upon the folded retroplume pathway and UMRs. We used the same release scenarios used in the evaluation, with every possible combination of the settings in Table 3.1 (144 simulations in total) in order to assess the impact of each model setting on the resulting folded retroplume pathway and UMRs. By running all possible groupings for these settings, we are able to evaluate the impact of changing one setting across all other possible settings. The model settings that were identified as the best settings (i.e., produced the most accurate pathway and UMRs) were used in the evaluation presented above. Here, we discuss the degree to which use of other model settings changed the evaluation results.

3.3.3.1 Folded retroplume pathway

Across all the model settings, the folded retroplume pathway was qualitatively similar to the results presented in the evaluation above. Larger output grid sizes naturally increased the size of the folded retroplume pathway, as the cells on the edge of the pathway were larger. The use of time-averaged output produced a mild ghosting effect, which is the superposition of negative particles that are located in the same cell but at different times during the averaging period. Thus, when averaged output was used, the folded retroplume pathway tended to be larger, with the occurrence of a few false positive

Table 3.1 Model settings used for evaluation..

Parameter name	Setting options
Averaging	On and off ¹
Temporal output interval	1 ¹ , 3, and 6 h
Spatial output grid ^a	0.5° ¹ , 1.0°, and 2.0°
Kernel	On and off ¹
Internal model time step ifine and ctl	5 and 20 ^{1, b}
Number of forward particles ^c	75 000 ^d and 500 000 ^{1, e}
Number of backward particles ^c	2500 ^f and 20 000 ^{1, g}

¹ Identified as preferred model settings for folded retroplume.

^a Used for both latitude and longitude simultaneously.

^b Used for both ifine and ctl simultaneously.

^c High and low number of particle pairs only run together.

^d Resulted in 20–40 positive particles.

^e Resulted in 250–625 positive particles.

^f Resulted in 25–35 positive particles.

^g Resulted in 250–650 positive particles.

PMRs along the edge of the transport pathway. Despite these two issues, the resultant folded retroplume pathway correctly identified the core transport pathway taken by the positive particle for all settings.

3.3.3.2 Folded retroplume UMRs

The general behavior of the UMRs were similar to those shown in Fig. 3.4: lower near the receptor, highly variable from 1 to 2.5 days upwind, relatively flat at approximately 3 days upwind, and very high at 4 days upwind, near the source. The higher positive bias in the UMR near the source region was present in all scenarios, indicating that no particular setting can help resolve this issue. This is not surprising, given the cause of this issue discussed in Sect. 3.3.2.2. The absolute value of the UMRs varied significantly with changes in the spatial size of the output grid, the frequency of output, and the use of average or instantaneous output, each of which we discuss further below. However, the other three model settings (the number of particles, the sampling kernel, and the model time steps, ifine and ctl) had little impact on the UMRs, and will not be discussed in detail. The number of particles were chosen so that forward and backward simulations both had roughly 30 or 600 positive particles for the small and large number of total particle sets, respectively. The lower number of particles was sufficient to return an accurate folded retroplume, which bodes well for future use of the method, as a lower number of particles can significantly decrease the computational time necessary for the forward simulations.

3.3.3.3 Spatial grid sizes

The size of the spatial grids can affect how positive and negative particles are associated with one another. A larger grid cell can either increase or decrease the UMR, depending on the circumstances. For example, consider a

cell that extends vertically from the surface into the FT, in a case in which the pollution plume that would reach the receptor was traveling in the lower FT. This would result in all the positive forward and backward particles residing in only the top half of the cell. If this cell were over an emissions source, then the lower portion of the cell would have a large number of negative forward particles, released from the surface source. The use of this single cell would result in a significant overestimate of the PMR, since the forward particles in the BL would be included in the calculation. However, if this cell were not over an emissions source and the bottom half of the cell had no forward particles, the result would be an underestimate of the PMR in this cell, as the larger cell would dilute the mass of the positive forward particles over a larger region, giving a smaller mixing ratio, without affecting the sensitivity. In our analysis, the resulting UMRs either stayed the same or decreased by 5–10% with each increase in the grid cell size, though the decrease in the UMRs was more pronounced as the grid cell size was increased from 1° to 2° . Therefore, we recommend use of a grid size of $1^\circ \times 1^\circ$.

3.3.3.4 Averaged or instantaneous values and the length of averaging period

The folding method we propose depends on the collocation in both time and space of positive particles in order to successfully identify cells that contain the trace substance that will travel between the source and receptor. As noted in Sect. 3.3.3.1, time averaging produced a ghosting effect and produced false positive PMRs. This ghosting effect also produced a positive bias in PMRs within the folded retroplume pathway, when, over the course of the averaging period, an output grid cell spanned a region that included both the plume of interest (positive forward particles) and forward-model CO that did not ultimately reach the receptor (negative forward particles). This could occur,

for example, when a sensitivity plume was located along the edge of the CO plume. The ghosting effect could cause the sensitivity plume to sample portions of the center of the CO plume, resulting in higher UMRs. Longer averaging periods can increase the likelihood that this ghosting effect can occur. At the shortest averaging period of 1 h, the averaged and instantaneous UMRs differed only slightly. The UMRs increased significantly as the length of the averaging period increased, with the UMRs roughly doubling at all times at each time interval increase. If UMR values are to be used quantitatively, we recommend instantaneous fields over averaged fields for a folded retroplume. Instantaneous values, however, increase the stochastic uncertainties of the output field. Since the 1 hr averaged output and the instantaneous output were quite similar, a short averaging period (e.g., less than or equal to 1 h) may also be a good option, as the averaged fields will reduce the stochastic uncertainties without affecting the UMRs greatly.

3.3.4 Summary of evaluation

The transport pathway evaluation indicates that the folded retroplume does a good job of restoring the source-to-receptor transport pathway and that this pathway is quite robust over a variety of model settings. The UMR evaluation, however, indicates that large gradients in either the trace substance or sensitivity field combined with minor differences in transport can significantly impact the UMRs. We also found that the ubiquitous nature of the trace substance in the source region in combination with a well-dispersed sensitivity plume can lead to positive deviations in the UMRs. The use of larger grid sizes or an averaging period greater than 1 hour can significantly degrade the accuracy of the UMRs, resulting in bias (mainly positive).

One important result from the evaluation is that significant deviations in

the UMRs did not correlate with significant differences between the folded retroplume and positive particle transport pathways. Whether the UMRs were high or low, the correct cells were generally identified (i.e., the cells with positive particles), with the core transport and fringe cells appropriately corresponding to high- and low-probability transport regions. As a result, the folded retroplume pathway appears to be a robust product, even when differences in transport between the forward and backward model are indicated by variations in the UMRs.

3.4 Sample application

In this section, we present a sample analysis that contrasts the folded retroplume method with traditional methods using only standard gridded LPDM products. The analysis will serve to provide an example of the advantages of the folded retroplume method over traditional LPDM analysis methods. The sample analysis will again focus on the transport of US emissions to the Pico Mountain observatory, examining the transport scenario for an event observed at the Pico Mountain observatory from 21–23 April 2005. During the event, CO mixing ratios ranged from 120–180 ppbv, approximately 30–90 ppbv above the typical springtime background at the station, while FLEXPART indicated enhancements of 20–50 ppbv of CO. Ozone, nitrogen oxides, and non-methane hydrocarbons were also elevated during this period. The event is the second of two events discussed by *Honrath et al.* [2008].

As with the evaluation simulations, we use FLEXPART version 6.2, driven with ECMWF meteorological data, with North American CO emissions based on the EDGAR inventory (see Sect. 3.3.1 for more details). For the sample analysis, we chose settings that are fairly typical of FLEXPART applications, even though they deviated somewhat from the recommendations above, es-

pecially in terms of the averaging period used. CO emissions were released continuously over North America into the lowest 300 m of the atmosphere and carried in the model for 20 days, after which time they were dropped from the simulation. Particles for the backward simulation were released over a 1-h period centered on 21:00 UTC on 21 April, into a box bounded by 38–39° N latitude 29–28° W longitude, from an altitude of 2000–2500 m a.s.l. The output was saved on a $1^\circ \times 1^\circ$ grid, with the top of the output levels at 0.3, 1, 1.5, 2, 2.5, 3, 4, 5, 7.5, 10, and 15 km. 6-h averages were saved and the sampling kernel was used. Particle trajectories from the backward model simulation were also saved and were used to confirm that the folded retroplume correctly captured the transport pathway of emissions between the source and receptor during the last 8 days of transport.

3.4.1 Comparison of the folded retroplume pathway with standard LPDM products

The sensitivity plume (Fig. 3.5a and c) shows fairly well organized transport originating over the US west coast about 7 days upwind. It also shows some sensitivity over a secondary region in the central US, from Texas to the Great Lakes. Since the age of this secondary region is not apparent, snapshots of the sensitivity would be needed to determine its age. The time-height cross section shows that the sensitivity plume was in the lower levels of the atmosphere, where it would be able to pick up emissions if they were present, from about 6.5 to 4 days upwind. At 4 days upwind, the bulk of the sensitivity plume experienced relatively rapid uplift to the lower FT, where it stayed until arrival at the observatory. Thus, from these views of the sensitivity plume, one could conclude that the bulk of emissions would have been picked up during the leg of transport from the US west coast to central, northern US (i.e., between the

6 and 4 in Fig. 3.5a). A map of the source contributions (i.e., $\sum_j S_{j,t,(j',t')} \cdot E_{j,t}$ from Eq. 3.7, not shown) would show the true sources, primarily the central US and only partly from the US west coast. However, tracking the emissions once away from the primary source region would be difficult due to the low sensitivity between this region and the compact transport pathway that occurs starting from approximately 2.5 days upwind.

In contrast to the sensitivity plume, the transport leg from the west coast almost disappears in the folded retroplume (Fig. 3.5b), indicating this transport pathway actually carries only a small amount of the trace substance. The folded retroplume shows that the emissions travel slowly northward in the lower atmosphere from upwind day 7 to 2. At 2 days upwind, all of the emissions were transported out of the BL into the lower FT, significantly later than indicated by the sensitivity plume. During the last 2 days of transport, the folded retroplume and sensitivity plume indicate virtually identical transport, which is not surprising since the sensitivity indicates little dispersion in the last 2 days of transport.

In order to examine the causes for the differences between the standard and folded retroplume and to further demonstrate the utility of the folded retroplume, in Fig. 3.6 we show snapshots of the sensitivity plume (left column) and the folded retroplume (right column) plotted with contours of the total column CO from the forward simulation (blue lines). Snapshots are shown for 7 (a and b), 5 (c and d), 4 (e and f), and 3 (g and h) days upwind. For consistency among these plots, we have used the maximum sensitivity and PMR from all 4 plots for the color scale maximum.

We have selected contour levels and color scales for Fig. 3.6 that approximate typical usage for the CO and sensitivity plumes (e.g., *Trickl et al.*, 2003), as some arbitrary selection of these settings was required. In some cases, it may be possible to adjust the contour levels and color scales such that features

that were originally not apparent become visible. Often, however, this would result in confounding other areas of the figure. Additionally, adjusting these settings for each snapshot and product is not a realistic analysis approach and quite often would require the analyst to have a prior knowledge of the feature they are trying to identify (from, e.g., the folded retroplume or particle trajectories). Thus, it is reasonable to assume that the features identified by the folded retroplume but not the standard products, as presented here, would indeed be missed in most analyses of this type.

At 7 days upwind (Fig. 3.6a and b), there are significant differences between the sensitivity plume and the folded retroplume. The region of sensitivity over the western US and eastern Pacific (Fig. 3.6a) completely disappears in the folded retroplume (Fig. 3.6b), while the region of sensitivity over the central US is emphasized, with relatively large PMRs. Considering the overlap of the sensitivity plume over the central US with the CO contours, the lack of CO in the western lobe of the sensitivity field, and the altitude of the primary portion of the sensitivity plume (Fig. 3.5), this result is not surprising. However, it serves to demonstrate the cause for differences between the sensitivity plume and the folded retroplume. The emphasized area over the central US with large PMRs, however, is considerably larger than the area indicated by the sensitivity plume and extends to regions with relatively low CO levels (i.e., it extends beyond the area enclosed by the CO contours).

At 5 days upwind (Fig. 3.6c and d), there remain significant differences between the sensitivity plume and the folded retroplume. The eastern lobe of the sensitivity plume has grown larger. The western lobe of the sensitivity plume now shows up in the folded retroplume, as it has picked up some emissions over the western US. In the absence of selectively picking contours and color scales, it would be impossible to identify the western lobe as contributing to CO transported to the receptor, as the sensitivity plume and selected

CO contours do not overlap.

By 4 days upwind (Fig. 3.6e and f), the primary eastern and secondary western lobes of the sensitivity plume and PMR fields have merged. A significant portion of the folded retroplume is now outside the CO contours. (That is, this feature would be missed using the standard products shown in panel e or by using superimposed contours of the sensitivity plume and forward output alone, like those shown in Fig. 3.1. A minimum CO contour level of 1% of the maximum would be required to indicate that the sensitivity plume and CO fields overlap. However, if this level were used for all the plots, then the sensitivity plume and CO contours would completely overlap at all upwind times shown in Fig. 3.6, which would incorrectly indicate that all of the sensitivity plume carried emissions.)

By 3 days upwind (Fig. 3.6g and h), the two lobes have almost completely coalesced and the outline of the folded retroplume and sensitivity plume are fairly similar, though the colors indicate the distribution of the trace substance within the plume is still quite different than the distribution of the sensitivity field. The folded retroplume and sensitivity plume are virtually identical during the final 2 days of transport (shown in Fig. 3.5), which is expected, as the sensitivity plume was very narrow during this period.

3.4.2 Folded retroplume UMRs

The UMR distribution, shown in Fig. 3.7 (thick, green solid line marked with diamonds), can be used to identify details about the addition of emissions to the plume. Increasing UMRs indicate the time periods when emissions were being added to the plume. The slope of the UMR line during this period indicates the rate at which emissions were added. Finally, constant UMRs indicate periods when the sensitivity plume was not located over source

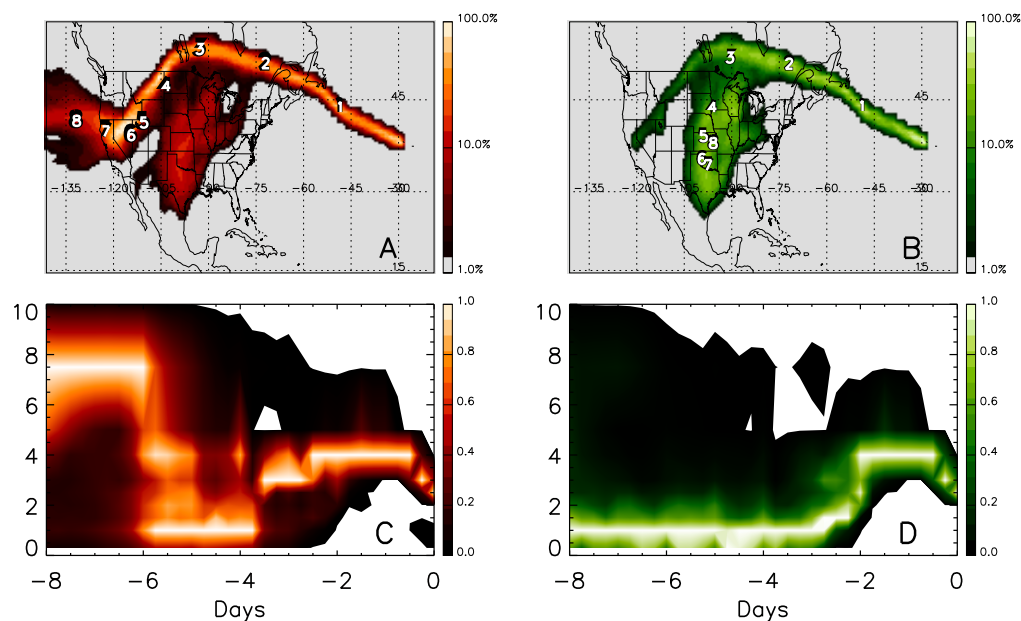


Figure 3.5 The time-height cross section (a and b) and plan views (c and d) of the sensitivity plume (red, a and c) and the folded retroplume (green, b and d) for the sample analysis. The plan-view color scales are based on the maximum specific volume weighted residence time ($3.1 \text{ s m}^3 \text{ kg}^{-1}$) and PMR (1.5 ppbv) for each plot. The white numerals show the average location of the sensitivity plume and folded retroplume at each upwind day, as indicated by the numbers. The time-height cross sections are normalized by the maximum value at each upwind time..

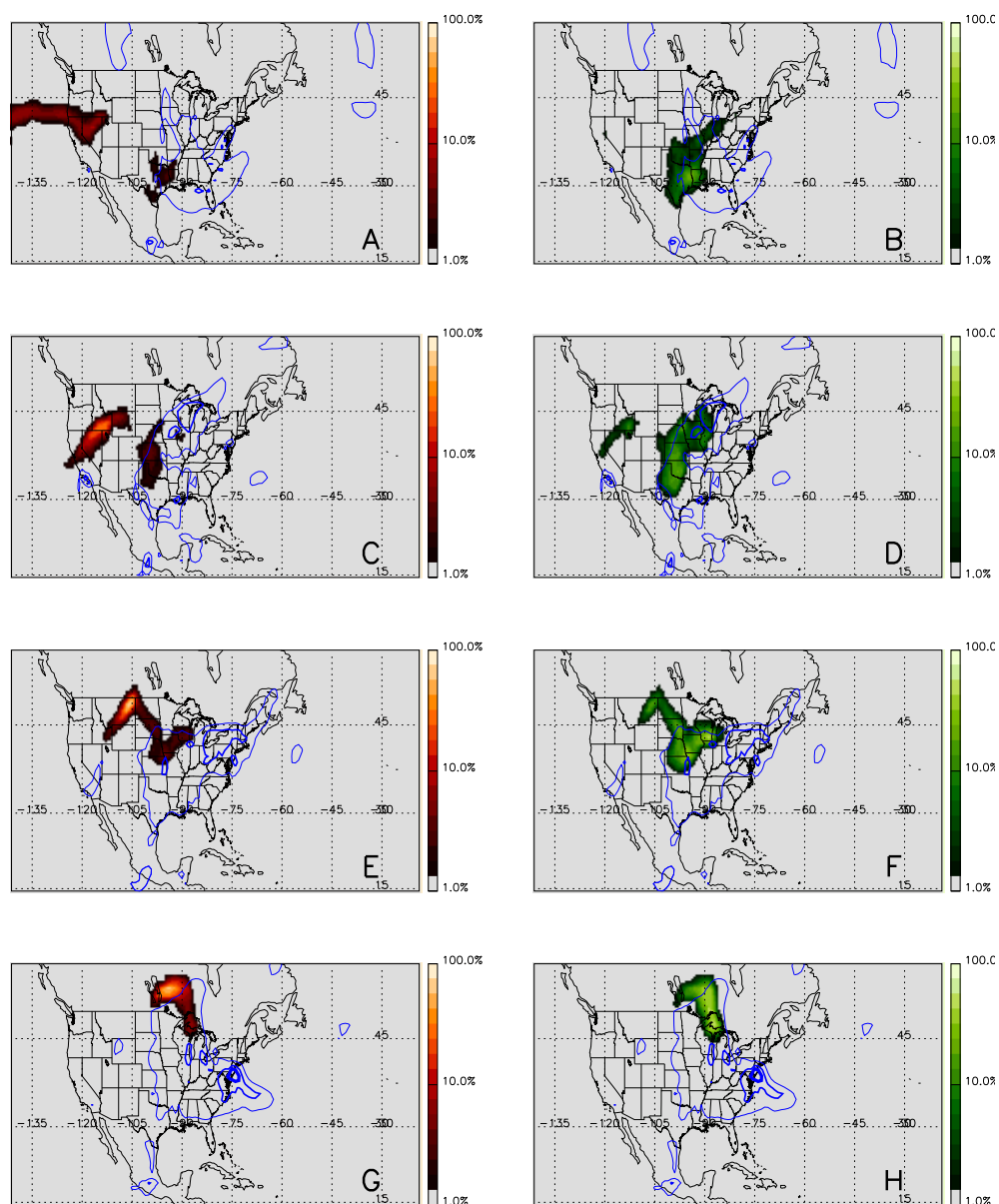


Figure 3.6 The plan view of snapshots of the horizontally integrated sensitivity plume (red, left column) and the folded retroplume (green, right column) with forward CO contours (blue lines, left and right columns) at 7 (a and b), 5 (c and d), 4 (e and f), and 3 (g and h) days upwind. Sensitivity plume color scale based on the maximum specific volume weighted residence time for all 4 plots ($3.1 \text{ s m}^3 \text{ kg}^{-1}$). Folded retroplume color scale based on the maximum PMR for all 4 plots (1.5 ppbv). CO contours are at 10, 20, 30, and 60% of the maximum column CO for all 8 plots (400 mg/m^2).

regions. (Note that if removal processes were included, decreases in the UMR would identify times during which removal occurred, though the interpretation of changes in the UMR would be complicated by the competition between emission and removal. See Sec. 3.5.3 for a discussion on this topic.)

The distribution of the CO mixing ratio age classes in the receptor, which can be derived from the forward model (blue line in Fig. 3.7) and from the backward model (by folding the sensitivity plume with the emissions inventory, magenta line in Fig. 3.7), provide another means to determine the details about the timing and rate of the addition of emissions to the plume. Therefore, a comparison of these distributions with the UMRs can help evaluate how well the UMRs accomplish this task by differentiating between changes in the UMRs that result from differences in transport and those that result from emissions (and/or removal, were it used). Since the folded retroplume samples the forward simulation, the UMR at t days upwind should equal the sum of the CO mixing ratio age classes from the forward model that are greater than t days old:

$$\text{UMR}_t \approx \sum_{i=t}^{A_f} \chi_{t,(j'),i}, \quad (3.8)$$

where A_f is the number of days the trace substance is carried in the forward model, $\chi_{t,(j'),i}$ is the CO mixing ratio of age i (in days) in the receptor cell (j'), and the sum is over all available CO mixing ratio age classes in the cell.

The relatively small change in the UMRs and CO distributions from 0 ppbv to about 5 ppbv during 20 to 10 days upwind indicates that, during this time, the majority of the plume was not over an emissions region. From 10 to 4 days upwind, however, the UMRs (green) and sensitivity-derived CO (magenta) increased by about 17 ppbv and the forward CO increased by about 20 ppbv (about 62% and 72% of the final UMR, respectively). This increase indicates that this is the time period when the plume was over the emission region and

CO was actively being added to the plume. Between 9 and 4 days upwind, the emission rate was fairly constant, at about 2 ppbv/day according to the UMRs. During the last 3 days of transport, however, CO distributions are relatively flat, indicating the plume was no longer in the emissions region, in agreement with the location of the folded retroplume shown in Figs. 3.5 and 3.6. The UMRs are also relatively flat between days 3 and 2, closely following the sensitivity-derived CO distribution. Between days 2 and 1, however, the UMRs increase significantly, coming closer to the forward CO distribution, and are flat again, following the CO from the forward simulation during the last day of transport.

In order to understand the differences between the UMRs and the forward CO mixing ratio distribution (23–24% from upwind day 7 to 2), we conducted an inspection of the CO plume, folded retroplume, sensitivity plume, and positive particles from the sensitivity plume during days 2–7 (not shown). This inspection revealed several features that, together, explain these low UMRs. First, the positive particles were not colocated with the areas with the maximum PMR, indicating minor differences in the transport between the forward and backward model simulations. Second, the gradient in the forward CO mixing ratio is fairly large over the area covered by the folded retroplume (varying by about 40% in the horizontal direction and about 85% in the vertical direction). The evaluation in Sect. 3.3 showed that in situations like this, with a large gradient in either the sensitivity or trace substance plume, the UMRs can be sensitive to minor displacements in the positive particle locations.

The magnitude of the deviations of the UMR can be put into perspective by considering the magnitude of the disagreement between the forward CO mixing ratio age class distribution (blue line in Fig. 3.7) and distribution derived from folding the sensitivity with the emissions inventory (magenta

line in Fig. 3.7). The differences between the forward (blue) and backward (magenta) derived CO distributions is partially the result of the differences in transport described by the two model simulations (and partially the result of differences in the number of particles used). *Seibert and Frank* [2004] showed that even with a short travel distance, the difference between the two can be in excess of 10%. The difference between the CO derived from the forward output and backward output convolved with the emissions inventory is not abnormal for FLEXPART, based on our own comparison of the two. (Our analysis consisted of a comparison of the total CO mixing ratio obtained from the forward model to the CO mixing ratio obtained from simulations of backward model, initiated at the Pico Mountain observatory, convolved with the emissions inventory. The comparison included approximately 6000 CO mixing ratios, covering 2 years worth of data.) The UMRs are in fact bounded by the forward and backward CO distributions, which are derived from established modeling methods. This suggests that the deviations in the UMR during these times are reasonable.

Archiving the full spectrum of forward trace substance age classes can require a significant amount of storage space. For the example presented here, we saved CO age classes at a 6-h resolution for the whole Northern Hemisphere in order to appropriately match sensitivity and CO mixing ratios. (This one-month simulation required 15.5 GB of storage). As a result, it is preferable for storage reasons to save only the total CO mixing ratio. To evaluate the effect that this would have on the UMR results, we have included the UMRs resulting from PMRs calculated by folding the sensitivity plume with all available CO mixing ratio age classes. Thus, Eq. (3.5) becomes

$$\text{PMR}_{j,t} = S_{j,t,(j',t')} \cdot \sum_{i=0}^{A_f} \chi_{j,t,i}. \quad (3.9)$$

These UMRs are shown in Fig. 3.7 (black line marked with asterisks).

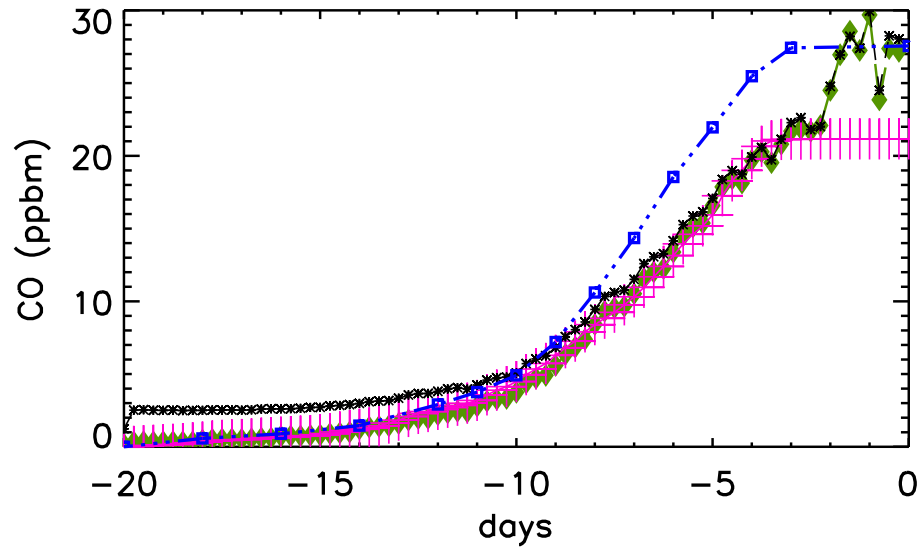


Figure 3.7 The UMRs at each upwind time for the sample evaluation. The UMRs calculated using the appropriate CO age classes are shown in green, while the alternate UMRs calculated with the total CO are shown in black. The cumulative CO distributions from the forward model (blue boxes) and the backward model folded with emissions (magenta crosses) are also shown..

They match the other UMRs quite well during the last 10 days of transport. However, from upwind days 10 to 20, there is an obvious positive bias in these UMRs, resulting from the sensitivity being folded with CO that will be dropped from the forward model before it can be transported to the receptor (i.e., the particles will reach the maximum age allowed and will thus be dropped from the forward simulation). Since we are mainly interested in the last ~ 10 days of transport, when the bulk of the emissions have been added to the plume and transported to the receptor, this would not be an issue with the example presented here. Deviations of the UMRs of this kind are most likely to occur at the greatest upwind times, when very old CO emissions can be folded with sensitivity many days upwind of the receptor.

3.5 Alternate backward LPDM combinations

The method we have presented uses the transport sensitivity from the backward formulation of a model to estimate the location and distribution of the trace substance in the forward model that will eventually arrive at the receptor. This is only one of many potential applications that combine the transport sensitivity from a backward LPDM with gridded fields from a number of sources. The simplest example, given by *Seibert and Frank* [2004], is the convolution of the sensitivity plume with an emissions inventory to determine the age class distribution of the mixing ratio of the trace substance in the receptor cell. Here, we discuss a few possibilities that focus on the calculation of UMRs and similar estimates of the physical attributes of the sensitivity plume at upwind times.

3.5.1 Chemical transport models

UMRs calculated from the folding of a chemical transport model (CTM) with the backward LPDM would allow for an evaluation of the mixing ratios in the portions of the CTM sampled by the sensitivity plume. This would give an estimate of the chemical transformations occurring in the CTM from the Lagrangian perspective provided by the sensitivity plume and would provide insight into the chemical and transport processes in the CTM. A similar application, in which the adjoint of transport was used to sample a CTM was given by *Vukićević and Hess* [2000] and *Hess and Vukićević* [2003]. *Vukićević and Hess* [2000] determined the adjoint of the CTM HANK to find the transport sensitivity of a non-reactive trace substance. These sensitivity fields were equivalent to those obtained from a backward LPDM. *Hess and Vukićević* [2003] then folded the adjoint sensitivity fields with output fields from the forward mode of the CTM to compute UMRs (or, as described by *Hess and*

Vukićević [2003], the sensitivity-weighted average of these fields).

In order to calculate UMRs from a CTM, ideally, the adjoint of transport in the CTM would be used because transport would be described the same in both model directions. However, the backward LPDM is also an attractive alternative, particularly when the adjoint of the CTM is not already available. The UMRs calculated in this manner will equal the mixing ratio in the receptor if the LPDM and CTM transport are sufficiently similar, no emissions are added, no chemistry occurs, and, if a passive trace substance is used in the LPDM, no removal has occurred within the plume. The agreement of transport between the CTM and LPDM could be tested by comparing forward model simulations using a passive trace substance and identical emissions sources. Changes in UMRs over time can thus be used to evaluate the net impact of emissions, removal and chemistry in the CTM from a Lagrangian perspective.

3.5.2 Meteorological fields, measurement and satellite data, and other applications

Other potential applications exist, since any data that are available on a 3-D grid can be combined with the backward LPDM to calculate sensitivity-weighted fields. Such 3-D data include meteorological fields, high-resolution measurements, and satellite data. Folding meteorological fields with the backward LPDM sensitivity provides the sensitivity-weighted upwind meteorological conditions for the sensitivity plume (e.g., the average temperature of the plume at each upwind time). This could be useful for determining the physical conditions of the air that will ultimately reach the receptor, which can be used to drive chemical reactions in a Lagrangian box model (e.g., *Evans et al.*, 2000).

UMRs calculated from atmospheric measurements provide an additional estimation of the chemical and physical properties of a plume upwind of a receptor. Fields of atmospheric composition derived from satellite observations or from intensive in-situ aircraft sampling could be analyzed by folding with the backward LPDM results to obtain UMRs. In these cases, the evolution of the air mass could be examined as it traveled to the receptor by comparing the UMRs over time.

3.5.3 Wet and dry removal

The discussion of the method has thus far focused only on simulations that do not include any removal processes (i.e., passive trace substances). The inclusion of removal processes, however, has important and interesting implications but requires careful consideration. Generally, removal should either be included or excluded in both of the paired model simulations. By comparing the UMRs from two sets of simulations — one from two simulations with no removal (subscript nr — UMR_{nr}) and one from two simulations with removal (subscript wr — UMR_{wr}), the timing of removal can be determined. For example, consider UMRs calculated at three time periods, one close to the source, t_s , one near the receptor, t_r , and an intermediate time, t_i . If no emissions are added to the air mass from t_s to t_r , then UMR_{nr} will be constant at all three times. However, if removal occurs between either t_s and t_i or t_i and t_r , then UMR_{wr} will decrease after the removal has occurred. Therefore, a comparison of the two UMRs will indicate the timing of removal. Once the timing has been determined from this method, the PMR_{wr} s can be compared with the PMR_{nr} s to identify where this removal occurred.

3.6 Practical considerations

There are a number of practical considerations relating to the model settings and parameters that should be taken into account when creating folded retroplumes. Two of the most important of these are discussed in this section and are expected to be of the greatest interest to readers wanting to conduct their own calculations of this type. This discussion is limited to the LPDM FLEXPART, but may be applied to other similar models by analogy.

3.6.1 FLEXPART output time stamp

In FLEXPART, the output is saved at the end of the averaging period. The forward and backward model simulations, however, are integrated in opposite directions in time. Thus, if averaged (normal) output is used, in order to appropriately match the model output, the length of the output interval must be added to the backward model output times, before identical forward and backward times are selected for folding.

3.6.2 Output unit options

The derivation presented in Sect. 3.2.1 used the volume mixing ratio and a form of the sensitivity that also has units of mixing ratio. Neither of these units are native to FLEXPART. Here, we discuss native FLEXPART output units that can be paired, along with some conversion factors, to obtain UMR units of volume mixing ratio. One such pairing is the mass missing ratio for the forward mode (with model settings `ind_source=1` and `ind_receptor=2`) and the residence time from the backward mode (`ind_source=1` and `ind_receptor=1`). In this case, the conversion factor necessary to obtain volume mixing ratios are the ratio of the mass of air to the mass of the trace substance (which serves to convert the mass mixing ratio to the volume mixing ratio) and the inverse

of output interval (which serves to convert the residence time to the mixing ratio sensitivity). (Note, the inverse of the output interval may or may not be necessary with instantaneous output, depending on how the modifications are made to the model code.) Another pair of native FLEXPART output units is concentration from the forward mode (`ind_source=1` and `ind_receptor=1`) and the specific volume weighted residence time (SVWRT, `ind_source=1` and `ind_receptor=2`). The conversion factors are identical to the first pair, the ratio of the mass of air to the mass of the trace substance and the the inverse of output interval. *Seibert and Frank* [2004] and *Stohl et al.* [2005] review other potential model outputs.

3.7 Summary and conclusions

This paper introduced a new method, the product of which we call a folded retroplume, that identifies source-to-receptor transport pathways using standard gridded products from a Lagrangian particle dispersion model (LPDM). The folded retroplume can be used to determine the transport pathway of only those emissions that arrive at the receptor at a designated time and to estimate the timing and location of emission and removal processes within the model.

An evaluation was conducted to determine the ability of the folded retroplume to identify the source-to-receptor transport pathway and to consistently calculate the expected sensitivity-weighted mixing ratios (upwind mixing ratios or UMRs) along the transport pathway. A comparison of the folded retroplume pathway with particle trajectories from both the forward and backward LPDM simulations showed that the folded retroplume was consistently able to reproduce the source-to-receptor transport pathway across a wide variety of model settings. Minor differences between the folded retroplume and particle

pathways along the edge of the core transport pathway were found, but were limited to one or two grid cells (typically 10–100 km), a fairly minor difference when compared to the length of the transport pathway analyzed (3620 km). The UMRs may be expected to be constant at all times between the source and receptor in the evaluation simulations. However, significant variations in UMRs were found to occur across all model settings. The best model settings produced UMRs that typically deviated from the expected value by 20–40%, while other model settings produced deviations exceeding 100%. These deviations resulted primarily from errors induced by using gridded data when large gradients in the trace substance and sensitivity fields were present and as well as minor differences between the source-to-receptor transport described by the forward and backward model simulations. The test simulations, however, provided a rather rigorous test of the folded retroplume properties, particularly the UMRs, as the evaluation transport scenario produced high gradients.

A sample analysis of the transport of North American CO to a monitoring station located in the Azores Islands contrasted the folded retroplume with traditional LPDM analysis tools. The folded retroplume made it possible to identify transport features that were difficult or impossible to determine with the traditional LPDM analysis techniques. The UMRs proved to be useful as a tool to determine the timing and rate at which emissions were added to the plume. Had the sample analysis included removal processes, the timing and rate of removal could also have been determined. We conclude that the folded retroplume is better than traditional LPDM analysis techniques for determining source-to-receptor pathways because it is significantly easier and more accurate than the alternative (separately viewing all the components of the LPDM). The folded retroplume also provides information unavailable from traditional LPDM products, such as the location and timing of removal processes and the spatial distribution of the trace substance between the source

and receptor.

The large deviations of the UMRs in the evaluation raised the issue of the reversibility of the transport scenario. Even though it was ultimately determined that the transport scenario was reversible, the issue of the reversibility of any particular transport scenario is important. While large variations of the UMR can occur under certain conditions even in the absence of irreversibility (as discussed in Sec. 3.3.2.2), UMRs that behave as expected indicate that the source-to-receptor transport pathway is correct and that the transport simulation was reversible. The example transport scenario presented in Sec. 3.4 serves to illustrate this idea. The UMRs in the example were within the bounds of the corresponding equivalent mixing ratios calculated from the forward and backward simulations, and the folded retroplume pathway was found to agree well with the positive particle trajectories from the backward simulation. Thus, UMRs may provide a means to determine when to accept the source-to-receptor transport pathway indicated by the folded retroplume as correct. In this respect, the folded retroplume may be superior to both a backward simulation and particle trajectories, neither of which offers a means to determine reversibility of the transport scenario.

The folded retroplume should be most useful for the analysis of measurements made in relatively remote regions of the atmosphere, where pollution impacts result from emissions that have traveled long distances. In these cases, emissions are often significantly dispersed, and the source-to-receptor pathway may be difficult to discern from standard LPDM products. This scenario is relatively typical of many monitoring stations (e.g., *Carpenter et al.*, 1997 and *Kentarchos et al.*, 2000) and aircraft-based sampling efforts (e.g., *Cooper et al.*, 2001 and *Lewis et al.*, 2007).

The LPDM folding technique can also be applied using the backward LPDM to sample fields of mixing ratios, concentrations, and meteorological

data from other sources in order to explore gridded data fields in a Lagrangian framework. When paired with the concentration output from a CTM, the method could be used to probe the chemistry occurring in the CTM within the confines of the sensitivity plume. Applied to satellite or field measurements, the method could be used to examine the chemical and physical evolution of a plume in the atmosphere. Meteorological conditions extracted with the method could be used to understand the conditions experienced by an air mass during transport and can be used in a trajectory chemical transport model or box model. These alternate applications of the technique offer new opportunities to study physical and chemical transformations that occur in the atmosphere.

Chapter 4

Summary and conclusions

This study produced two major results. First, it determined that eastward advection and transport in a weak and strong warm conveyor belts (WCB) was responsible for the export of North American emissions into the FT, while transport in the FT was governed either by easterly winds driven by the Azores/Bermuda High (ABH) and transient northerly lows or through the WCB and the associated descending dry air stream. Second, it developed and tested a new method for the Lagrangian tracking of pollution plumes using gridded model data. This section reviews the results of these analyses and discusses future work to further the understanding of the links between transport and O_3 production in the lower FT over the central North Atlantic.

4.1 Mechanisms for transport of North American pollution to the central North Atlantic lower free troposphere

In order to determine the mechanisms responsible for the transport of North American anthropogenic emissions to the central North Atlantic lower FT,

measurements of O_3 and CO were made at the Pico Mountain observatory in the Azores Islands in the central North Atlantic at an altitude of 2225 masl. These measurements were analyzed in conjunction with simulations of the Lagrangian particle dispersion model FLEXPART, satellite images, and weather maps. A total of 16 events were observed at the observatory during the month of July 2003 FLEXPART was able to reproduce 13 of these event while simulating 3 events that were not observed at the observatory. Of the 16 events, 4 were selected to be case studies for a detailed analysis and description of the responsible transport.

In three of the case studies, eastward advection over the ocean and transport in a weak warm conveyor belt (WCB) airflow was found to be the transport mechanism responsible for the export of emissions out of the North American boundary layer (BL). Once over the ocean, transport was governed by the geostrophic wind between the Azores-Bermuda High and transient northern lows. The trans-Atlantic transport altitude for these three events and 13 of the 16 events was generally less than 3 km, accounting for a total of 84% of the total event hours observed at the observatory.

Despite the relatively low-level transport for these events, O_3 enhancements were observed coincident with CO in most of the cases. Since one third of the of the CO export from North America in July occurs below 3 km [*Li et al.*, 2005], these results suggest that low-level export into the lower FT, but above the marine boundary layer (where significant O_3 destruction is known to occur), may provide an important mechanism for the long-range transport of anthropogenic O_3 . Several other studies have also noted the potential importance of this type of low-level transport, which can preserve O_3 and its precursors, allowing for continued O_3 production (e.g., *Neuman et al.* [2006]; *Guerova et al.* [2006]; *Derwent et al.* [2008]).

In the fourth case study, emissions were transported out of the US BL and

over the Atlantic Ocean by the dominant westerly wind. Once over the ocean, however, the emissions were rapidly lofted to the middle FT in a WCB. Once in the middle FT, the emissions traveled eastward for 3 days in the center of the low pressure system associated with the WCB. Finally, the emissions were transported to the Pico Mountain observatory during rapid subsidence in the cold air descending behind the cold front (or the dry air stream). One other event experienced similar lofting and subsidence in the WCB and dry air streams associated with a transient northern low. Ozone was also elevated in these two events, indicating that O_3 can effectively be exported out of the continental BL to the middle FT and be returned to the lower FT, where it is more likely to affect downwind surface regions.

4.2 New methods for tracking plumes from their source to a receptor using gridded model output

A new method was introduced which produces a new analysis tool that we call a folded retroplume. The folded retroplume can be used to identify source-to-receptor transport pathways by combining the forward and backward output from a Lagrangian particle dispersion model (LPDM). The folded retroplume can be used to determine the transport pathway of only those emissions that arrive at the receptor at a designated time and to estimate the timing and location of emission and removal processes within the model.

A comparison of the folded retroplume transport pathway with particle trajectories from both the forward and backward LPDM simulations showed that the folded retroplume was consistently able to reproduce the source-to-receptor transport pathway across a wide variety of model settings. However,

an evaluation of the ability of the folded retroplume to consistently calculate the expected sensitivity-weighted mixing ratios (upwind mixing ratios or *UMRs*) along the transport pathway showed that significant variations in *UMRs* occurred across all the model settings evaluated. The best model settings produced *UMRs* that typically deviated from the expected value by 20–40%, while other model settings produced deviations exceeding 100%. These deviations resulted from a combination of large gradients in the tracer and sensitivity fields and minor differences between the source-to-receptor transport described by the forward and backward model simulations. The test simulations, however, provided a rather rigorous test of the folded retroplume properties, particularly the *UMRs*, as the evaluation transport scenario produced high gradients.

A sample analysis of the transport of North American CO to the Pico Mountain observatory was conducted in order to contrast the folded retroplume with traditional LPDM analysis tools. The folded retroplume made it possible to identify transport features that were difficult or impossible to determine with the traditional LPDM analysis techniques. For example, the *UMRs* could be used to determine the timing and rate of wet or dry removal that occurs during transport. We conclude that the folded retroplume is better than traditional LPDM analysis techniques for determining source-to-receptor pathways because it is significantly easier and more accurate than the alternative (separately viewing all the components of the LPDM that would otherwise be required) and because it also provides information unavailable from traditional LPDM products, such as the location and timing of removal processes and the spatial distribution of the receptor-bound tracer between the source and receptor.

The LPDM folding technique can also be applied by using the backward LPDM to sample fields of mixing ratios, concentrations, and meteorological

data from other sources in order to explore gridded data fields in a Lagrangian framework. When paired with the concentration output from a CTM, the method could be used to probe the the chemistry occurring in the CTM within the confines of the sensitivity plume. When applied to satellite or field measurements, the method could be used to examine the chemical and physical evolution of a plume in the atmosphere. Meteorological conditions extracted with the method could be used to understand the physical conditions experienced by an air mass during transport and can be used in a trajectory chemical transport model or box model. These alternate applications of the technique offer new opportunities to study physical and chemical transformations that occur in the atmosphere.

4.3 Application and future analysis

The study presented here, particularly the development of the new analysis technique, lays much of the groundwork necessary for many new analyses. In this section, one such analysis focusing on determining links between transport characteristics and the O_3/CO slopes observed at the Pico Mountain observatory is presented.

As detailed in section 1.1, O_3 is formed by reactions of other pollutants that are often emitted along with CO (i.e., combustion sources such as vehicles). Therefore, the same sources responsible for increases in O_3 (above the background concentration) are likely to also be responsible for increases in CO. The ratio of O_3 to CO can thus help determine the amount of O_3 formed after the CO emission. Since CO emissions are also relatively well known, O_3/CO relationships can also be used to estimate O_3 production from the source region responsible for observed CO and O_3 enhancements [Parrish *et al.*, 1993].

Honrath et al. [2004] reported the O_3/CO slope for a 24 events observed at the Pico Mountain observatory during the summers of 2001 and 2003. Nine of these events were determined to be the result of recent anthropogenic emissions originating over North America. As outlined in section 1.1, the average slope for these events was around 1.0, significantly higher than any previously reported O_3/CO slopes. The time series of the CO and O_3 for these 9 events are shown in Figure 4.1. The cause for these higher slopes is unknown, but it was suggested that longer transport times to the mid-Atlantic allowed for continued O_3 production, thus moderately aged anthropogenic plumes could have higher O_3/CO slopes. Alternately, it was suggested that a unique transport pattern, that kept emissions in the lower FT, but above the marine BL, preserved both O_3 and O_3 precursors, allowing for continued O_3 production.

Here, we present a preliminary analysis that could be used to determine if relationships exist between the O_3/CO slope observed during events and the transport time or between the observed slope and the trans-Atlantic transport altitude. The age distribution of the FLEXPART CO tracer is used to determine the transport time from North America while folded retroplumes, calculated from retroplumes initiated at the Pico Mountain observatory and with forward model simulations of anthropogenic emissions from North America, are used to determine the average trans-Atlantic transport height. Figure 4.1 shows the total CO enhancements determined from the forward model simulation (solid blue line with diamonds) and the enhancements calculated by combining the retroplumes with the emissions inventory, which we refer to as the backward CO (solid green line with triangles). FLEXPART appears to model these events to varying degrees. For most of the events, FLEXPART at least indicates a CO enhancement (greater than 10 ppbv CO [*Owen et al.*, 2006]) during most or part of the event. During events 1 (E1), 5 (E5), and

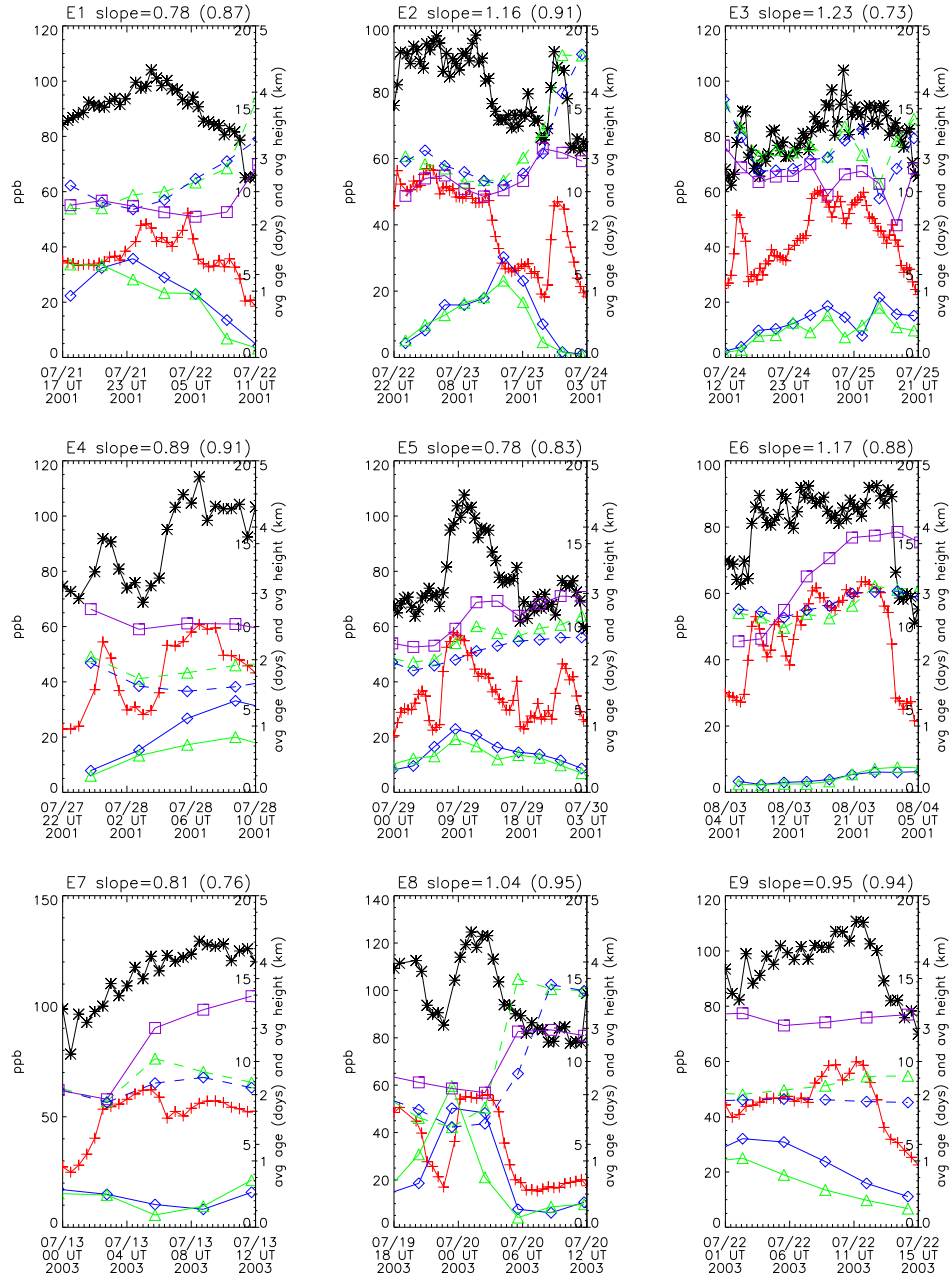


Figure 4.1 The CO (solid black line with asterisks), O₃ (solid red line with crosses), FLEXPART CO enhancement from the forward (solid blue line with diamonds) and backward (solid green line with triangles) model simulations, the average FLEXPART tracer age from the forward (dashed blue line with diamonds) and backward (dashed green line with triangles) model simulations, and the trans-Atlantic transport height determine from folded retroplumes (solid purple line with squares) for the 9 events of North American anthropogenic origin identified by *Honrath et al.* [2004]. The O₃/CO slope is given in the title of each plot window, the r^2 is given in the parentheses.

9 (E9), FLEXPART appears to capture the CO variability during the event, with the modeled CO enhancements generally increasing along with CO and O_3 . Events 2 (E2), 6 (E6), and 7 (E7), however, were not modeled well. The FLEXPART CO appears to be anti-correlated with the CO and O_3 from the observatory during event 2, while the FLEXPART CO enhancement is low and/or uncorrelated with changes in the observed CO and O_3 during events 6 and 7. However, the modeled CO enhancements during event 7, which begin to increase towards the end of the event, continue to increase for several hours after the event and may indicate that there is an offset in the timing of the modeled plume (a situation which was not dealt with in this analysis).

The average age of the modeled CO enhancements are also shown in Figure 4.1. The forward CO age is indicated by blue dashed lines with diamonds and the backward CO age is indicated by the green dashed line with triangles. There is a clear anti-correlation between the size of the modeled CO enhancement and the average age of the CO tracer - as the modeled enhancements increase, the average age decreases. This is likely because faster transport times allow for less dispersion, resulting in a more concentrated tracer plume. For the events that were modeled well by FLEXPART, this anti-correlation also appears to exist between the measured CO enhancements and the modeled average age. It is also apparent from Figure 4.1 that the average age of the modeled CO enhancements vary significantly during each event. For example, the average age of the modeled CO enhancements for event 8 varies from around 7 days at the start of the event up to 15 days towards the end of the event. (While the CO has dropped by the end of the time period shown, we have not attempted to redefine the timing or length of any events and have exclusively used the time frame determined by *Honrath et al.* [2004].) Despite the variation of the CO age during the course of the events, there appears to a reasonable correlation between the average FLEXPART tracer

age (the average of the varying average ages of the modeled CO enhancements calculated for each retroplume during an event) and the O_3/CO correlations for each event (Figure 4.2). A test for significance of the correlation indicates that the r^2 is significantly different from zero at roughly a 95% confidence level ($\alpha = 0.05$, *Ogus et al.* [2007]). If we consider only those events that were captured reasonably well by FLEXPART (Figure 4.3), the slope remains quite similar (roughly 11-12), though the r^2 is notably lower and only significantly different from zero at an 82% confidence level ($\alpha = 0.18$). (In this case, the correlation appears to be driven largely by event 3. If event 3 is excluded from the regression, the slope remains in this range, but the r^2 values are 0.05 and 0.01 for the forward and backward CO, which are not significantly different from zero). These results do not provide evidence of a strong relationship between the O_3/CO slope and the average age calculated in this fashion.

The average trans-Atlantic transport altitude was determined by calculating the average height of the folded retroplume after 75% of total model CO had been added to the folded retroplume, which was generally the last 5-6 days of transport (purple line with squares in Figure 4.1). The average transport heights also vary significantly during each event, though the resulting averages between the events vary only by 1 km, falling between 2.3 and 3.4 km (Figure 4.4). The correlation indicates that higher transport altitudes result in higher O_3/CO slopes. While the r^2 for this correlation is relatively small (0.18), the significance test indicates that the r^2 is significantly different from zero at a 75% confidence level ($\alpha = 0.25$). If we again consider only those events that were captured reasonably well by FLEXPART (Figure 4.5), the slope decreases slightly but the r^2 is so low that there is only a 40% chance that the r^2 is significantly different from zero ($\alpha = 0.60$), too low to draw any firm conclusions. (In this regression, event 9 appears to be an outlier. When removed, the r^2 increases dramatically, to 0.33, which is different from zero

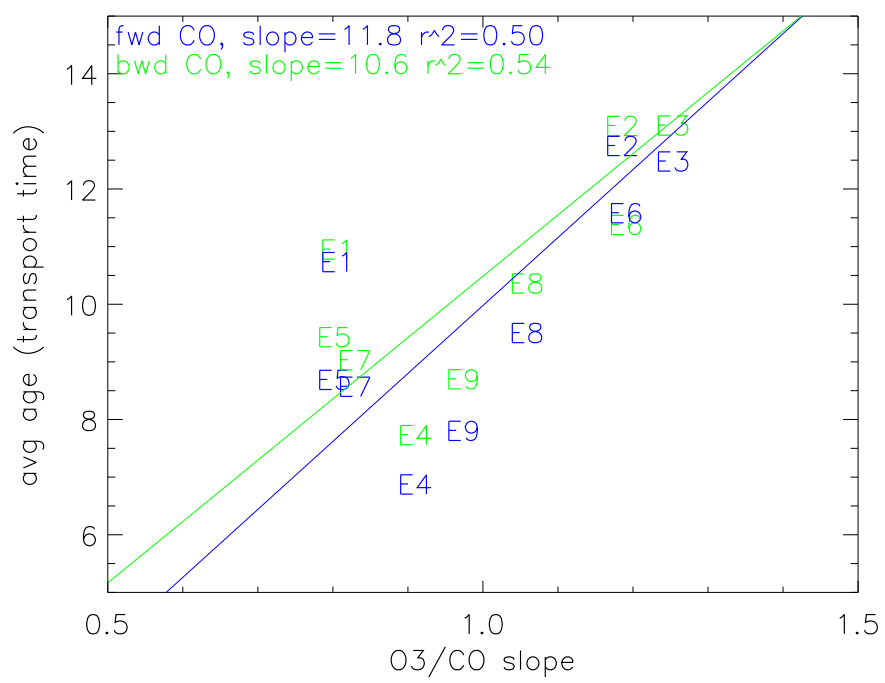


Figure 4.2 Correlations between the observed O_3/CO slopes and the average of the modeled forward (blue) and backward (green) FLEXPART CO enhancements for all events.

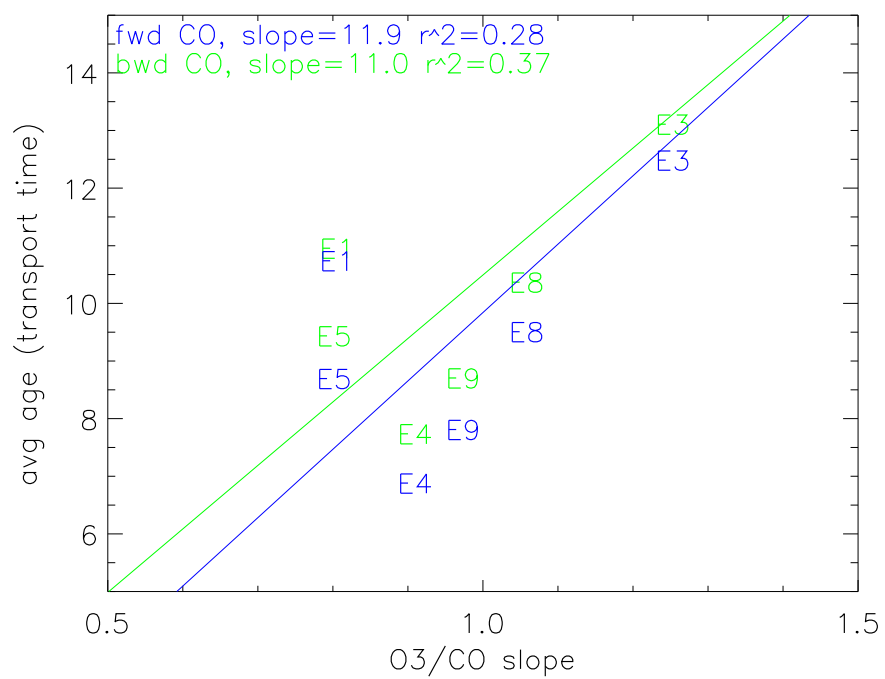


Figure 4.3 Correlations between the observed O_3/CO slopes and the average of the modeled forward (blue) and backward (green) FLEXPART CO enhancements for only those events that were sufficiently captured by FLEXPART (events 1, 3, 4, 5, 8, and 9).

at a 70% confidence level.) These results suggest that there may be a weak relationship between the O_3/CO slope and the transport height, as calculated here, but the level of significance questionable, particularly for the regression based on only the well-modeled events.

In order to determine how representative the average heights shown in Figure 4.1 are of the trans-Atlantic transport, the folded retroplumes for event 3 were examined. This event was chosen because it is one of the longer events, it displays a range of average ages and transport heights, and is of interest because it has the highest O_3/CO slope. Figures 4.6 and 4.7 show the vertical and horizontal view of the the folded retroplumes from select times during event 3. The black lines in the vertical views show the *PMR*-weighted height (or average height) calculated at each upwind time. The average of these height values give the average trans-Atlantic transport heights shown in Figure 4.1. Due to the *PMRs* in the lower probability transport regions, the average heights tend to be below the primary transport region, particularly at times during the event when there exists a higher altitude component (i.e., July 25 at 06 and 18 UT during event 3). As a result, the average transport heights for each event are also lower and forced into a smaller altitude range. Thus, it appears that the average height calculated in this fashion is not fully representative of the trans-Atlantic transport height.

Sample folded retroplumes for the other events are shown in Figures 4.8 and 4.9. These plots show that there are variety of transport pathways responsible for these events. The altitude and horizontal plots for event 7 appear to indicate WCB induced lofting and subsidence, while the horizontal and vertical plots for event 1 indicate primarily zonal flow, without significant uplift. As with the vertical plots for event 3, Figure 4.8 indicates that the *PMR*-weighted heights have a tendency to fall below the primary transport pathway once the folded retroplume becomes vertically dispersed.

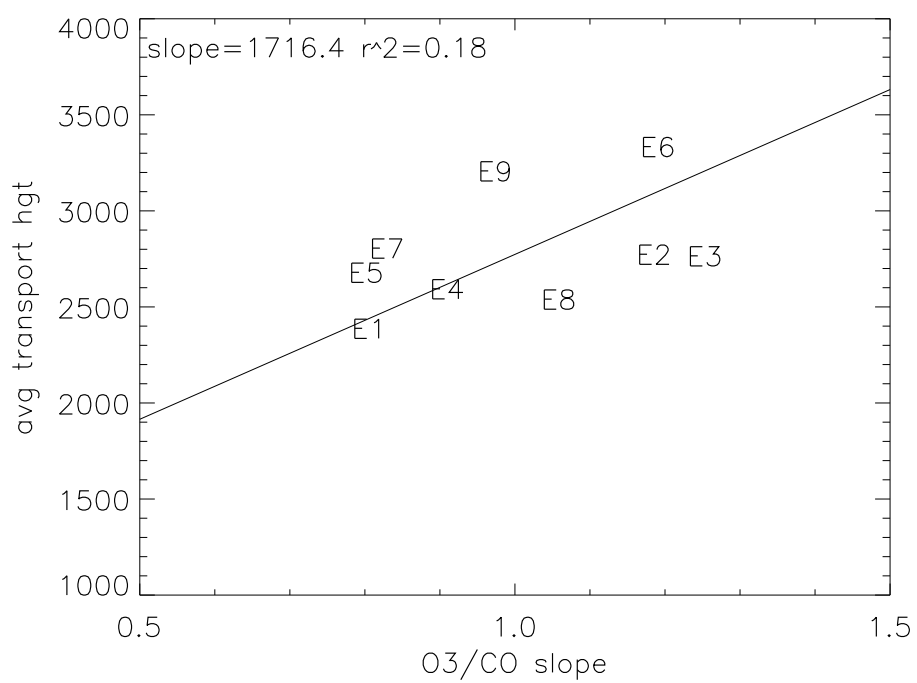


Figure 4.4 Correlations between the observed O₃/CO slopes and the average trans-Atlantic transport height of the folded retroplume for all events.

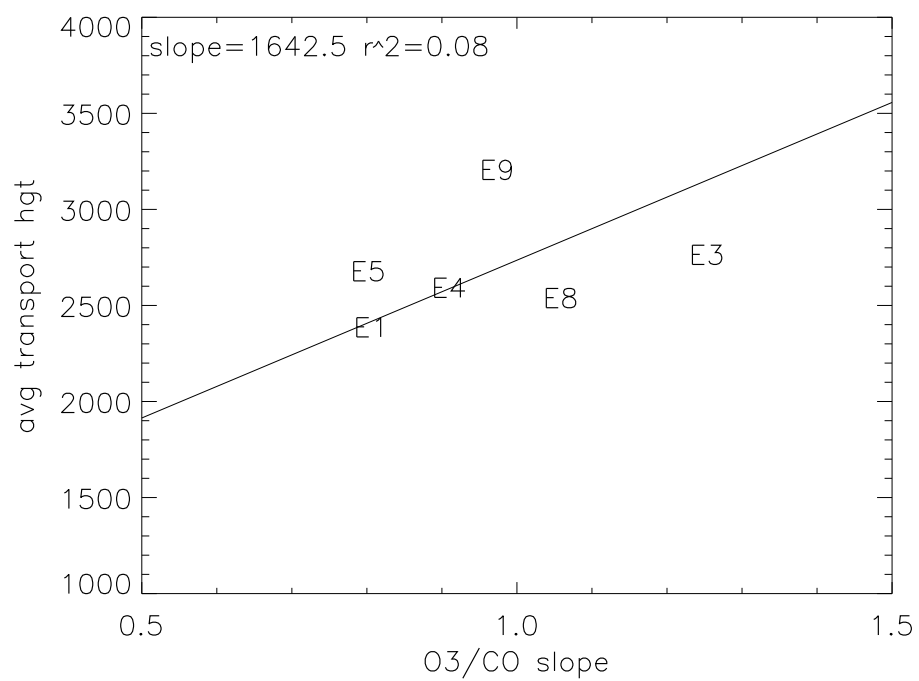


Figure 4.5 Correlations between the observed O₃/CO slopes and the average trans-Atlantic transport height of the folded retroplume for only those events that were sufficiently captured by FLEXPART (events 1, 3, 4, 5, 8, and 9).

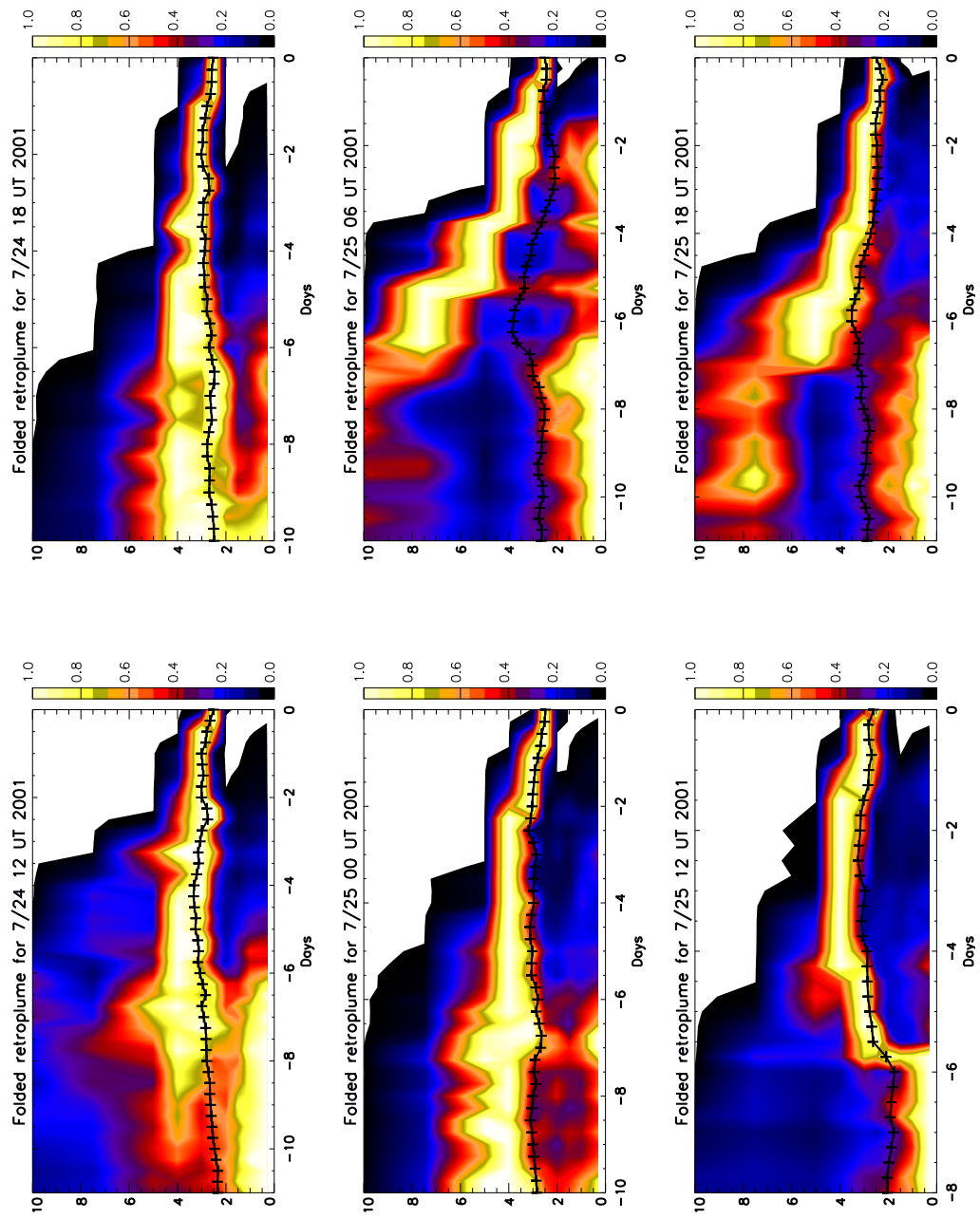


Figure 4.6 The vertical view of folded retroplumes initiated during event 3. The plots are normalized by the maximum value at each upwind times. The black lines indicate the *PMR*-weighted height, or average height, at each upwind time.

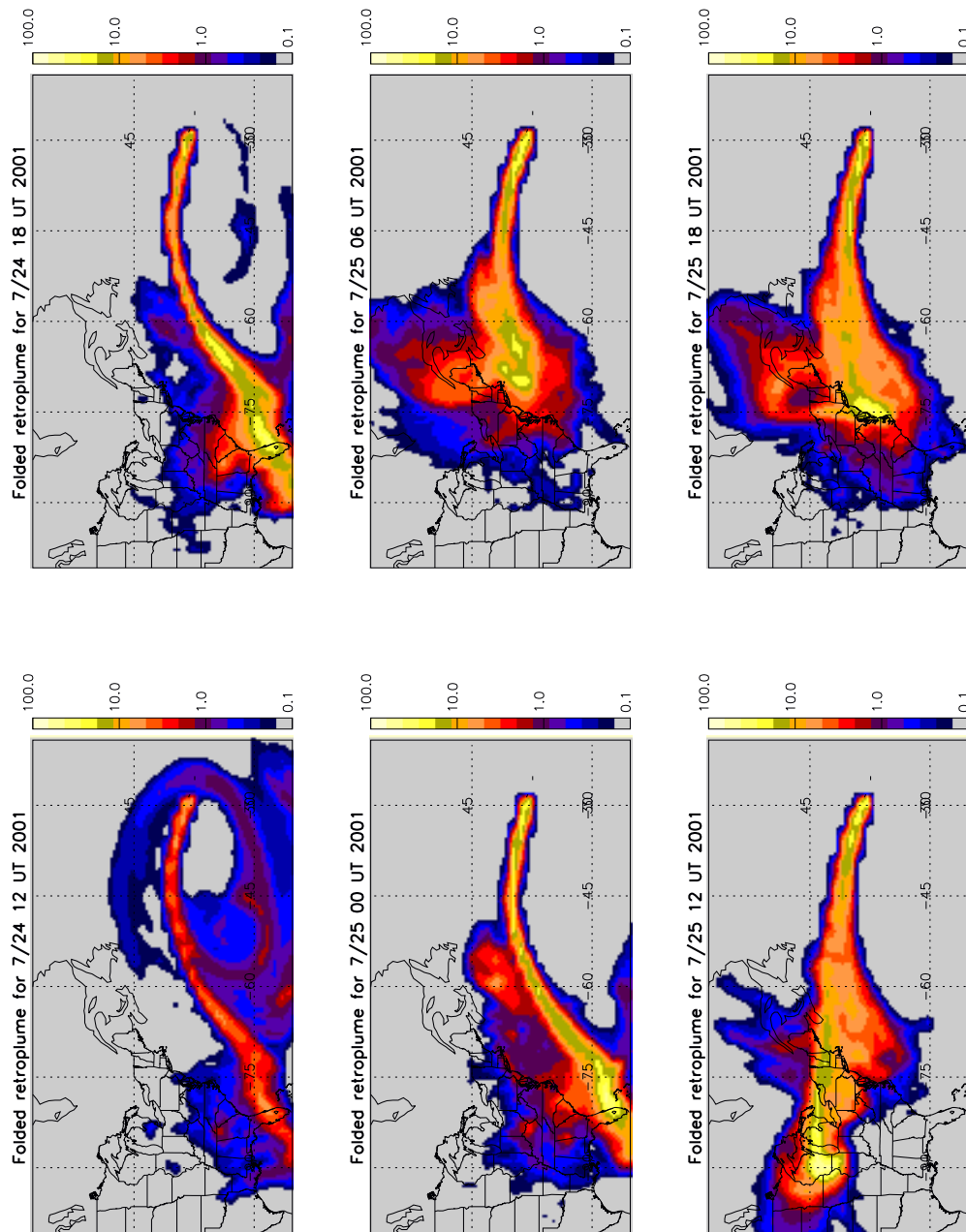


Figure 4.7 The horizontal view of folded retroplumes initiated during events 3. The plots are scaled according to the maximum mixing ratio for each plot.

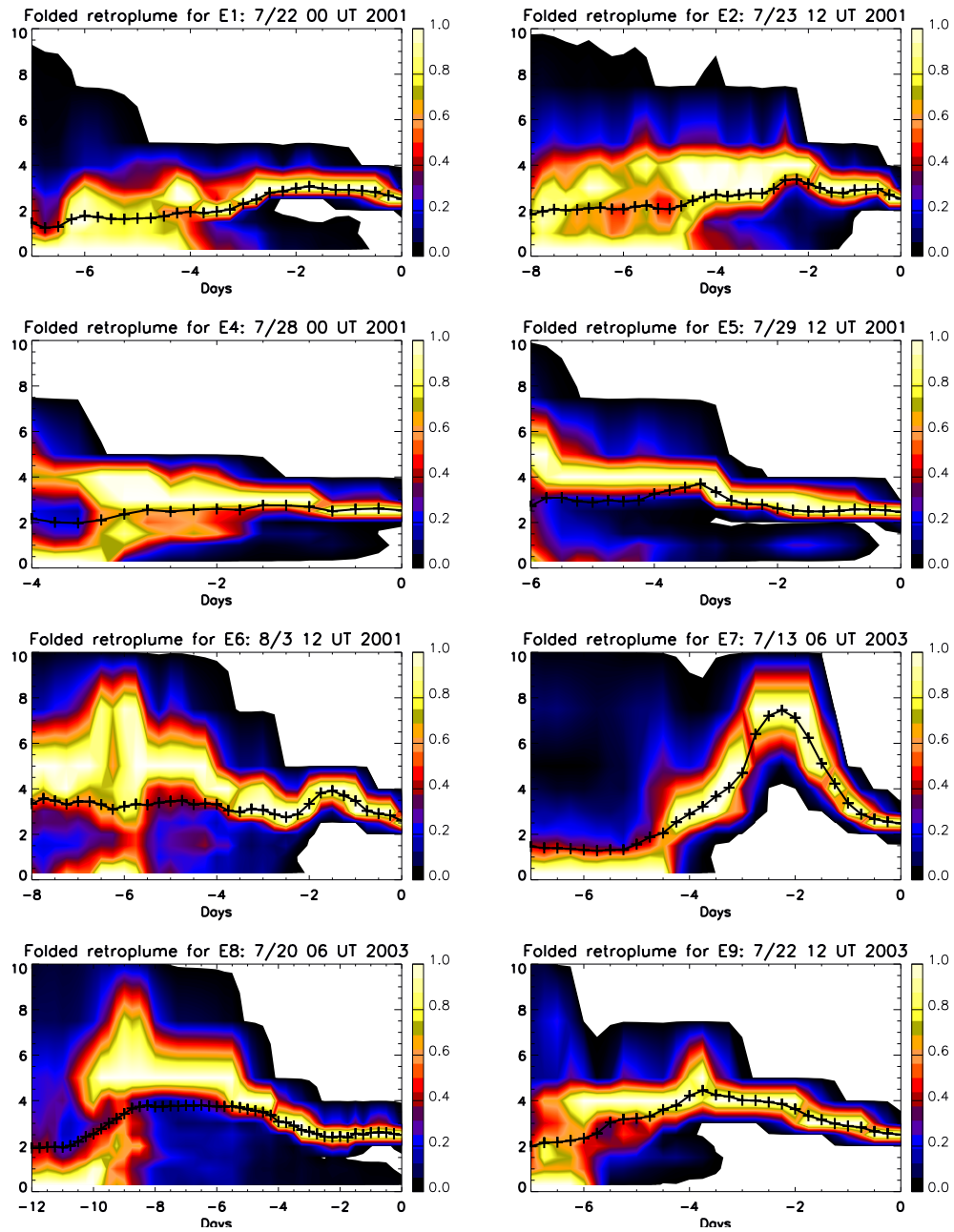


Figure 4.8 The vertical view of folded retroplumes initiated during events 1, 2, 4, 5, 6, 7, 8, and 9. The plots are normalized by the maximum value at each upwind times. The black lines indicate the *PMR*-weighted height, or average height, at each upwind time.

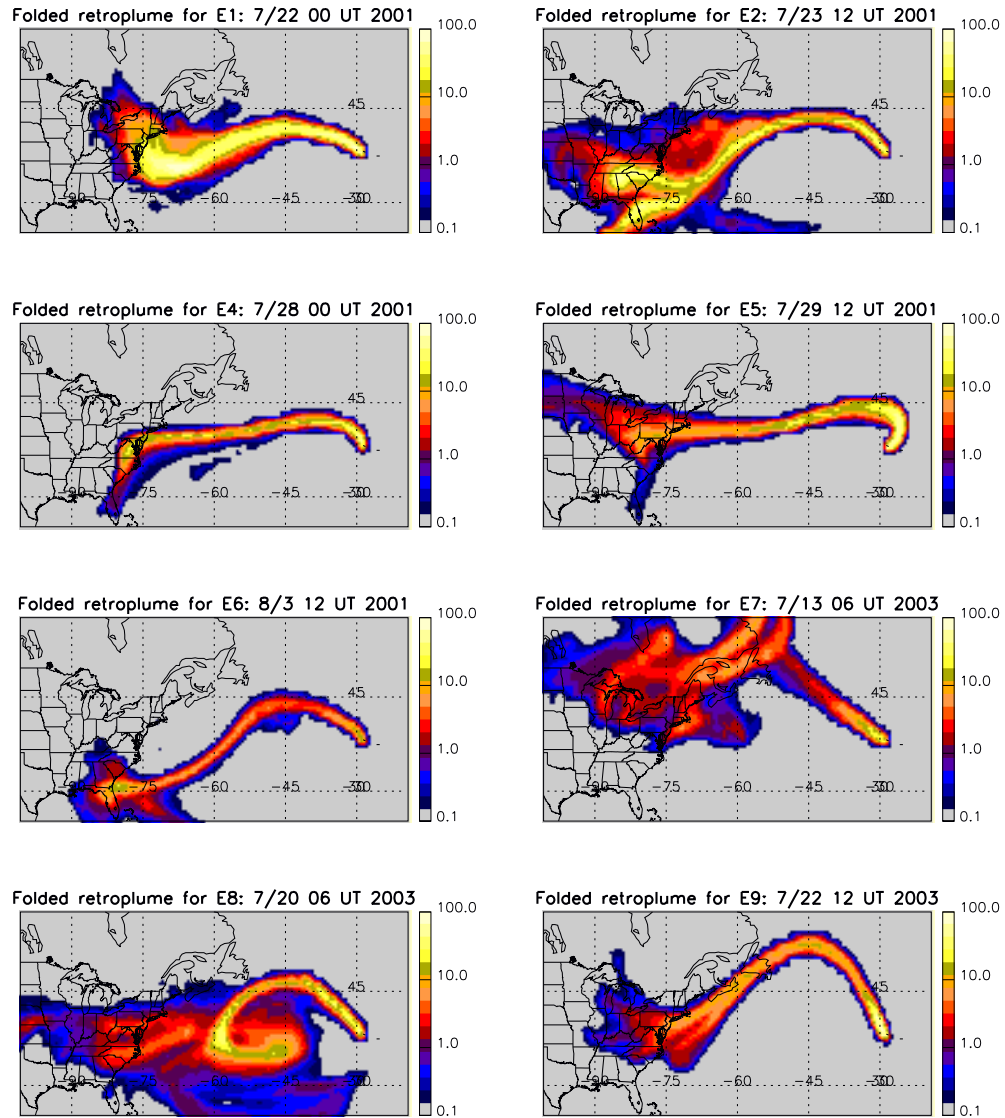


Figure 4.9 The horizontal view of folded retroplumes initiated during events 1, 2, 4, 5, 6, 7, 8, and 9. The plots are scaled according to the maximum mixing ratio for each plot.

This analysis provides interesting results. Based on the methods used here, the results suggest that there may be a relationship between both the average age of the modeled CO and the O₃/CO slopes and between the modeled average trans-Atlantic transport height and the O₃/CO slope. However, the strength and significance of the relationships vary greatly, depending on whether the poorly modeled events were included and whether or not apparently outlying data points were included in the regressions. The analysis does, however, present questions regarding both the methodology and the results, including:

- Should the transport time be determined as the average tracer age or should it be determined by some other method (i.e., the time after which 75% of total model CO had been released, as was used in the determination of the transport height).
- Was 75% of the total CO the most appropriate percentage to use when determining the transport height?
- Since the retroplumes are constrained by their departure altitude (boundary layer) and arrival altitude (the Pico Mountain observatory at 2.2 km), should the first and last days of the trans-Atlantic transport be excluded from the average transport height?
- Since the *PMR*-weighted heights tended to fall below the primary transport pathway, should the maximum or average maximum transport height be used instead average *PMR*-weighted height?
- This analysis used only 2 summers of out of a total of 5 summers and did not address O₃ and CO observed in other seasons. Could the inclusion of this larger data set improve statistical results (i.e., the r^2)?

- FLEXPART NO_x simulations that include wet removal have been made. Could these simulations provide insight into the role played by wet removal in the observed O_3/CO slopes?
- *Owen et al.* [2006] divided some of the 2003 events into smaller events, based off changes in the source regions and enhancements indicated by FLEXPART. Should a similar analysis be used to divide the events further or perhaps, could the O_3/CO ratio for each data point be considered, so as to eliminate the large variability in the FLEXPART age and height parameters that were calculated during the course of the events?

In addition to addressing these questions for the Pico Mountain observatory data, conducting a similar analysis using data from additional measurement locations would be likely to produce important results regarding the relationships between transport and O_3 production. Despite these issues, the results of such an analysis may help reveal if and how strong of a relationship exists between either modeled transport heights or modeled plume ages and observed O_3/CO slopes, which in turn could help increase the understanding of the relationships between O_3 production and pollution transport processes.

References

- Angevine, W. M., et al., Coastal boundary layer influence on pollutant transport in New England, *Journal of Applied Meteorology*, *43*, 1425–1437, 2004.
- Auvray, M., and I. Bey, Long-range transport to Europe: Seasonal variations and implications for the European ozone budget, *J. Geophys. Res.*, *110*, 2005.
- Cape, J. N., J. Methven, and L. E. Hudson, The use of trajectory cluster analysis to interpret trace gas measurements at Mace Head, Ireland, *Atmos. Environ.*, *34*, 3651–3663, 2000.
- Carpenter, L. J., P. S. Monks, B. J. Bandy, S. A. Penkett, I. E. Galbally, and C. P. Meyer, A study of peroxy radicals and ozone photochemistry at coastal sites in the northern and southern hemispheres, *J. Geophys. Res.*, *102*, 25,417–25,427, 1997.
- Chin, M., D. J. Jacob, J. W. Munger, D. Parrish, and B. Doddridge, Relationships of ozone and carbon monoxide over North America, *J. Geophys. Res.*, *99*, 14,565–14,573, 1994.
- Cooper, O., and D. Parrish, Air pollution export from and import to North America: Experimental evidence, in *The Handbook of Environmental Chemistry*, edited by A. Stohl, vol. 4G, 41–67, Springer, New York, NY, 2004.
- Cooper, O., et al., Trace gas signatures of the airstreams within North Atlantic cyclones: Case studies from the North American Regional Experiment (NARE '97) aircraft intensive, *J. Geophys. Res.*, *106*, 5437–5456, 2001.
- Cooper, O., et al., A case study of transpacific warm conveyor belt transport: Influence of merging airstreams on trace gas import to North America, *J. Geophys. Res.*, *109*, 2004.
- Cooper, O. R., J. L. Moody, D. D. Parrish, M. Trainer, T. B. Ryerson, J. S. Holloway, G. Hübner, F. C. Fehsenfeld, and M. J. Evans, Trace gas composition of midlatitude cyclones over the western North Atlantic Ocean: A conceptual model, *J. Geophys. Res.*, *107*, 2002.

- Derwent, R., P. Simmonds, S. Seuring, and C. Dimmer, Observations and interpretation of the seasonal cycles in the surface concentrations of ozone and carbon monoxide at Mace Head, Ireland from 1990 to 1994, *Atmos. Environ.*, *32*, 145–157, 1998.
- Derwent, R., D. Stevenson, R. Doherty, W. Collins, and M. Sanderson, How is surface ozone in Europe linked to Asian and North American NO_x emissions?, *Atmos. Environ.*, *42*, 7412–7422, 2008.
- Dickerson, R. R., B. G. Doddridge, and P. Kelley, Large-scale pollution of the atmosphere over the remote Atlantic Ocean: Evidence from Bermuda, *J. Geophys. Res.*, *100*, 8945–8952, 1995.
- Dickerson, R. R., et al., Thunderstorms: An important mechanism in the transport of air pollutants, *Science*, *235*, 460–465, 1987.
- Dorling, S. R., and T. D. Davies, Extending cluster analysis–synoptic meteorology links to characterise chemical climates at six northwest European monitoring stations, *Atmos. Environ.*, *29*, 145–167, 1995.
- Draxler, R. R., and G. D. Rolph, Hysplit (hybrid single-particle lagrangian integrated trajectory), Model access via NOAA ARL READY Website (<http://www.arl.noaa.gov/ready/hysplit4.html>), NOAA, Air Resources Laboratory, Silver Spring, MD., 2003.
- Eckhardt, S., A. Stohl, H. Sodemann, A. F. P. Seibert, and G. Wotawa, The Lagrangian particle dispersion model FLEXPART version 8.0, *Tech. rep.*, Norwegian Institute of Air Research, 2008.
- ECMWF, Users guide to ECMWF products 2.1, *Tech. Rep. Meteorological Bulletin M3.2*, European Center for Medium-Range Weather Forecasts (ECMWF), Reading, UK, 1995.
- ECMWF, Users guide to ECMWF products 4.0, *Tech. Rep. Meteorological Bulletin M3.2*, European Center for Medium-Range Weather Forecasts (ECMWF), Reading, UK, 2005.
- Errico, R. M., What is an adjoint model?, *Bull. Amer. Meteor. Soc.*, *78*, 2577–2591, 1997.
- Evans, M., et al., Evaluation of a Lagrangian box model using field measurements from EASE (Eastern Atlantic Summer Experiment) 1996, *Atmos. Environ.*, *34*, 3843–3863, 2000.
- Fehsenfeld, F., et al., International Consortium for Atmospheric Research on Transport and Transformation (ICARTT): North America to Europe–Overview of the 2004 summer field study, *J. Geophys. Res.*, *111*, 2006.

- Fishman, J., F. M. Vukovich, and E. V. Browell, The photochemistry of synoptic-scale ozone synthesis: Implications for the global tropospheric ozone budget, *J. of Atmos. Chem.*, *3*, 299–320, 1985.
- Flesch, T., J. Wilson, and E. Yee, Backward-Time Lagrangian Stochastic Dispersion Models and Their Application to Estimate Gaseous Emissions, *Journal of Applied Meteorology*, *34*, 1230–1332, 1995.
- Fowler, D., J. N. Cape, M. Coyle, C. Flechard, J. Kuylenstierna, K. Hicks, D. Derwent, C. Johnson, and D. Stevenson, The global exposure of forests to air pollutants, *Water, Air, and Soil Pollution*, *116*, 5–32, 1999.
- Guerova, G., I. Bey, L.-L. Attie, R. Martin, J. Cui, and M. Sprenger, Impact of transatlantic transport episodes on summertime ozone in Europe, *Atmos. Chem. and Phys.*, *6*, 2057–2072, 2006.
- Han, Y.-J., T. M. Holsen, P. K. Hopke, and S.-M. Yi, Comparison of back-trajectory based modeling and Lagrangian backward dispersion modeling for location sources of reactive gaseous mercury, *Environ. Sci. Technol.*, *23*, 1715–1723, 2005.
- Helmig, D., D. Tanner, R. Honrath, R. Owen, and D. Parrish, Non-methane hydrocarbons (NMHC) at Pico Mountain, Azores: 1. Oxidation chemistry in the North Atlantic region, *J. Geophys. Res.*, pp. submitted, January 14, 2008, 2008.
- Hess, P., and T. Vukićević, Intercontinental transport, chemical transformations, and baroclinic systems, *J. Geophys. Res.*, *108*, 4354, 2003.
- Hoell, J., D. Davis, S. Liu, R. Newell, H. Akimoto, R. McNeal, and R. Bendura, The Pacific Exploratory Mission–West Phase B: February–March, 1994, *J. Geophys. Res.*, *102*, 28,223–28,239, 1997.
- Honrath, R., R. Owen, M. Val Martín, J. Reid, K. Lapina, P. Fialho, M. Dzoibak, J. Kleissl, and D. Westphal, Regional and hemispheric impacts of anthropogenic and biomass burning emissions on summertime CO and O₃ in the North Atlantic lower free troposphere, *J. Geophys. Res.*, *109*, 2004.
- Honrath, R., D. Helmig, R. Owen, D. Parrish, and D. Tanner, Non-methane hydrocarbons (NMHC) at Pico Mountain, Azores: 2. Event-specific analysis of the impacts of mixing and photochemistry on hydrocarbon ratios, *J. Geophys. Res.*, *113*, D20S92, 2008.
- Houghton, J. T., Y. Ding, D. J. Griggs, M. Noguer, P. J. van der Linden, X. Dai, K. Maskell, and C. A. Johnson, *Climate Change 2001: The scientific basis. Contribution of Working Group I to the Third Assessment Report of the Intergovernmental Panel on Climate Change*, Cambridge University Press, 2001.

Hudman, R., et al., North American influence on tropospheric ozone and the effects of recent emission reductions: constraints from ICARTT observations, *J. Geophys. Res.*, *submitted*, 2008.

Huntrieser, H., et al., Intercontinental air pollution transport from North America to Europe: Experimental evidence from airborne measurements and surface observations, *J. Geophys. Res.*, *110*, 2005.

IPCC, Climate Change 2007: The Physical Science Basis. Contribution of Working Group I to the Fourth Assessment Report of the Intergovernmental Panel on Climate change, 2007.

Jacob, D. J., et al., Simulation of summertime ozone over North America, *J. Geophys. Res.*, *98*, 14,797–14,816, 1993.

Jaffe, D., I. McKendry, T. Anderson, and H. Price, Six "new" episodes of trans-pacific transport of air pollutants, *Atmos. Environ.*, *37*, 391–404, 2003.

Jaffe, D., et al., Transport of Asian air pollution to North America, *Geophys. Res. Lett.*, *26*, 711–714, 1999.

Kahl, J. D. W., D. A. Martinex, H. Kuhns, C. I. Davidson, J.-L. Jaffrezo, and J. M. Harris, Air mass trajectories to Summit, Greenland: A 44-year climatology and some episodic events, *J. Geophys. Res.*, *102*, 26,861–26,875, 1997.

Karnosky, D. F., K. S. Pregitzer, D. R. Zak, M. E. Kubiske, G. R. Hendrey, D. Weinstein, M. Nosal, and K. E. Percy, Scaling ozone responses of forest trees to the ecosystem level in a changing climate, *Plant, Cell and Environment*, *28*, 965–981, 2005.

Kentarchos, A. S., G. J. Roelofs, J. Lelieveld, and E. Cuevas, On the origin of elevated surface ozone concentrations at Izana Observatory, Tenerife during late March 1996, *Geophys. Res. Lett.*, *27*, 3699–3702, 2000.

Kleissl, J., R. Honrath, M. Dziobak, D. Tanner, M. Val-Martín, R. Owen, and D. Helmig, The occurrence of upslope flows at the Pico mountaintop atmospheric observatory: a case study of orographic flows on small, volcanic islands, *J. Geophys. Res.*, *112*, D10S35, 2007.

Lapina, K., R. E. Honrath, R. C. Owen, M. V. Martín, E. Hyer, and P. Fialho, Late-summer changes in burning conditions in the boreal regions and their implications for NO_x and CO emissions from boreal fires, *J. Geophys. Res.*, *113*, 2008.

Lelieveld, J., J. van Aardenne, H. Fischer, M. de Reus, J. Williams, and P. Winkler, Increasing ozone over the Atlantic Ocean, *Science*, *304*, 1483–1487, 2004.

Levy, H., II, Normal Atmosphere: Large Radical and Formaldehyde Concentrations Predicted, *Science*, *173*, 141–143, 1971.

Lewis, A. C., et al., Chemical composition observed over the mid-atlantic and the detection of pollution signatures far from source regions, *J. Geophys. Res.*, *112*, D10S39, 2007.

Li, Q., D. Jacob, R. Park, Y. Wang, C. L. Heald, R. H. R. Martin, and M. Evans, North American pollution outflow and the trapping of connectively lifted pollution by upper-level anticyclone, *J. Geophys. Res.*, *110*, 2005.

Li, Q., et al., Transatlantic transport of pollution and its effects on surface ozone in Europe and North America, *J. Geophys. Res.*, *107*, ACH 4–1–ACH 4–21, 2002.

Lin, C.-Y. C., D. J. Jacob, J. W. Munger, and A. M. Fiore, Increasing background ozone in surface air over the United States, *Geophys. Res. Lett.*, *27*, 3465–3468, 2000.

Lin, J., C. Gerbig, S. Wofsy, A. Andrews, B. Daube, K. David, and C. Grainger, A near-field tool for simulating the upstream influence of atmospheric observations: The Stochastic Time-Inverted Lagrangian Transport (STILT) model, *J. Geophys. Res.*, *108*, 4493, 2003.

Liu, S. C., M. Trainer, F. C. Fehsenfeld, D. D. Parrish, E. J. Williams, D. W. Fahey, G. Hübner, and P. C. Murphy, Ozone production in the rural troposphere and the implications for regional and global ozone distributions, *J. Geophys. Res.*, *92*, 4191–4207, 1987.

Mauzerall, D., and X. Wang, Protecting agricultural crops from the effects of tropospheric ozone exposure: Reconciling science and standard setting in the United States, Europe, and Asia, *Annual Review of Energy and the Environment*, *26*, 237–268, 2001.

McKeen, S. A., E. Y. Hsu, M. Trainer, R. Tallamraju, and S. C. Liu, A regional model study of the ozone budget in the Eastern United States, *J. Geophys. Res.*, *96*, 10,809–10,845, 1991.

Merrill, J. T., and J. L. Moody, Synoptic meteorology and transport during the North Atlantic Regional Experiment (NARE) intensive: Overview, *J. Geophys. Res.*, *101*, 28,903–28,921, 1996.

Methven, J., S. Arnold, F. O'Connor, H. Barjat, K. Dewey, J. Kent, and N. Brough, Estimating photochemically produced ozone throughout a domain using flight data and a Lagrangian model, *J. Geophys. Res.*, *108*, 2003.

Methven, J., et al., Establishing Lagrangian connections between observations within air masses crossing the Atlantic during the international Consortium for

- Atmospheric Research on Transport and Transformation experiment, *J. Geophys. Res.*, *111*, D23S62, 2006.
- Mickley, L. J., D. J. Jacob, B. D. Field, and D. Rind, Climate response to the increase in tropospheric ozone since preindustrial times: A comparison between ozone and equivalent co₂ forcings, *J. Geophys. Res.*, *109*, 2004.
- Monks, P. S., Gas-phase radical chemistry in the troposphere, *Chemical Society Reviews*, *34*, 376–395, 2005.
- Moody, J. L., S. J. Oltmans, H. L. II, and J. T. Merrill, Transport climatology of tropospheric ozone: Bermuda, 1988–1991, *J. Geophys. Res.*, *100*, 7179–7194, 1995.
- Moy, L. A., R. R. Dickerson, and W. F. Ryan, Relationship between back trajectories and tropospheric trace gas concentrations in rural Virginia, *Atmos. Environ.*, *28*, 2789–2800, 1994.
- Neuman, A., et al., Reactive nitrogen transport and photochemistry in urban plumes over the North Atlantic Ocean, *J. Geophys. Res.*, *111*, D23S54, 2006.
- Ogus, E., A. C. Yazici, and F. Gurbuz, Evaluating the significance test when the correlation coefficient is different from zero in the test of hypothesis, *Communications in Statistics Simulation and Computation*, *36*, 847–854, 2007.
- Olivier, J., and J. Berdowski, Global emissions sources and sinks, in *The Climate System*, edited by J. Berdowski, R. R. Guicherit, and B. Heij, vol. 33–78, A.A. Balkema Publishers / Swets and Zeitlinger Publishers, Lisse, The Netherlands, 2001.
- Olivier, J., A. Bouwman, C. V. der Maas, J. Berdowski, C. Veldt, J. Bloos, A. Visschedijk, P. Zandveld, and J. Haverlag, Description of EDGAR Version 2.0: A set of global emission inventories of greenhouse gases and ozone-depleting substances for all anthropogenic and most natural sources on a per country basis and on 1°x1° grid, *Tech. Rep. 771060 002 and R96/119*, National Institute of Public Health and the Environment (RIVM) and Netherlands Organization for Applied Scientific Research (TNO), 1996.
- Olivier, J., A. Bouwman, J. Berdowski, C. Veldt, J. Bloos, A. Visschedijk, and C. V. der Maas P.Y.J. Zandveld, Sectoral emission inventories of greenhouse gases for 1990 on a per country basis as well as on 1°x1° degree, *Environ. Sci. Technol.*, *2*, 241–264, 1999.
- Owen, R. C., and R. E. Honrath, Technical note: A new method for the Lagrangian tracking of pollution plumes from source to receptor using gridded model output, *Atmospheric Chemistry and Physics*, *9*, 2577–2595, 2009.

- Owen, R. C., O. R. Cooper, A. Stohl, and R. E. Honrath, An analysis of the mechanisms of North American pollutant transport to the central North Atlantic lower free troposphere, *J. Geophys. Res.*, *111*, D23S58, 2006.
- Park, R. J., D. J. Jacob, B. D. Field, R. M. Yantosca, and M. Chin, Natural and transboundary pollution influences on sulfate-nitrate-ammonium aerosols in the United States: Implications for policy, *J. Geophys. Res.*, *109*, 2004.
- Parrish, D., M. Trainer, J. Holloway, J. Yee, M. Warshawsky, and F. Fehsenfeld, Relationships between ozone and carbon monoxide at surface sites in the North Atlantic region, *J. Geophys. Res.*, *103*, 13,357–13,376, 1998.
- Parrish, D. D., J. S. Holloway, M. Trainer, P. C. Murphy, G. L. Forbes, and F. C. Fehsenfeld, Export of North American ozone pollution to the North Atlantic Ocean, *Science*, *259*, 1436–1439, 1993.
- Parrish, D. D., J. S. Holloway, and F. C. Fehsenfeld, Routine, continuous measurement of carbon monoxide with parts per billion precision, *Environ. Sci. Technol.*, *28*, 1615–1618, 1994.
- Parrish, D. D., M. Trainer, E. J. Williams, K. J. Olszyna, R. A. Harley, J. F. Meagher, and F. C. Fehsenfeld, Decadal change in carbon monoxide to nitrogen oxide ratio in U.S. vehicular emissions, *J. Geophys. Res.*, *107*, 2002.
- Parrish, D. D., et al., Changes in the photochemical environment of the temperate North Pacific troposphere in response to increased Asian emissions, *J. Geophys. Res.*, *109*, 2004.
- Pickering, K. E., A. M. Thompson, R. R. Dickerson, W. T. Luke, D. P. McNamara, J. P. Greenberg, and P. R. Zimmerman, Model calculations of tropospheric ozone production potential following observed convective events, *J. Geophys. Res.*, *95*, 14,049–14,062, 1990.
- Pochanart, P., H. Akimoto, Y. Kajii, V. M. Potemkin, and T. V. Khodzher, Regional background ozone and carbon monoxide variations in remote Siberia/East Asia, *J. Geophys. Res.*, *108*, 2003.
- Price, H. U., D. A. Jaffe, O. R. Cooper, and P. V. Doskey, Photochemistry, ozone production, and dilution during long-range transport episodes from Eurasia to the northwest United States, *J. Geophys. Res.*, *109*, 2004.
- Reeves, C. E., et al., Potential for photochemical ozone formation in the troposphere over the North Atlantic as derived from aircraft observations during ACSOE, *J. Geophys. Res.*, *107*, 2002.
- Rex, M., et al., In situ measurements of stratospheric ozone depletion rates in the Arctic winter 1991/1992: A Lagrangian approach, *J. Geophys. Res.*, *103*, 5843–5853, 1998.

Rodríguez, S., C. Torres, J.-C. Guerra, and E. Cuevas, Transport pathways of ozone to marine and free-troposphere sites in Tenerife, Canary Islands, *Atmos. Environ.*, *38*, 4733–4747, 2004.

Rolph, G. D., Real-time Environmental Applications and Display sYstem (READY) Website (<http://www.arl.noaa.gov/ready/hysplit4.html>), NOAA, Air Resources Laboratory, Silver Spring, MD., 2003.

Ryall, D. B., R. H. Maryon, R. G. Derwent, and P. G. Simmonds, Modelling long-range transport of CFCs to Mace Head, Ireland, *Q. J. R. Met. Soc.*, *124*, 417–446, 1998.

Seibert, P., and A. Frank, Source-receptor matrix calculation with a Lagrangian particle dispersion model in backward mode, *Atmos. Chem. and Phys.*, *4*, 51–63, 2004.

Sillman, S., The relation between ozone, NO_x and hydrocarbons in urban and polluted rural environments, *Atmos. Environ.*, *33*, 1821–1845, 1999.

Sirois, A., and J. W. Bottenheim, Use of backward trajectories to interpret the 5-year record of PAN and O₃ ambient air concentrations at Kejimikujik National Park, Nova Scotia, *J. Geophys. Res.*, *100*, 2867–2881, 1995.

Skyllingstad, E. D., R. M. Samelson, L. Mahrt, and P. Barbour, A numerical modeling study of warm offshore flow over cool water, *Mon. Weather Rev.*, *133*, 345–361, 2005.

Smedman, A.-S., H. Bergström, and B. Grisogono, Evolution of stable internal boundary layers over a cold sea, *J. Geophys. Res.*, *102*, 1091–1099, 1997.

Stevenson, D. S., et al., Multimodel ensemble simulations of present-day and near-future tropospheric ozone, *J. Geophys. Res.*, *111*, 2006.

Stohl, A., Computation, accuracy and applications of trajectories—a review and bibliography, *Atmos. Environ.*, *32*, 947–966, 1998.

Stohl, A., A one-year lagrangian "climatology" of airstreams in the northern hemisphere troposphere and lowermost stratosphere, *J. Geophys. Res.*, *106*, 2001.

Stohl, A., and D. J. Thomson, A density correction for Lagrangian particle dispersion models, *Boundary Layer Met.*, *90*, 155–167, 1999.

Stohl, A., and T. Trickl, A textbook example of long-range transport: simultaneous observation of ozone maxima of stratospheric and North American origin in the free troposphere over Europe, *J. Geophys. Res.*, *104*, 30,445–30,462, 1999.

Stohl, A., and T. Trickl, Long-range transport of ozone from the North American boundary layer to Europe: Observations and model results, in *Air Pollution Modelling and its Application XIV*, edited by S.-E. Gryning and F. Schiermeier, vol. 257-266, Kluwer Academic/Plenum Publishers, New York, 2001a.

Stohl, A., and T. Trickl, Experimental evidence for trans-Atlantic transport of air pollution, *IGACt. Newsl.*, *24*, 10–12, 2001b.

Stohl, A., M. Hittenberger, and G. Wotawa, Validation of the Lagrangian particle dispersion model FLEXPART against large scale tracer experiment data, *Atmos. Environ.*, *24*, 4245–4264, 1998.

Stohl, A., S. Eckhardt, C. Foster, P. James, and N. Spichtinger, On the pathways and timescales of intercontinental air pollution transport, *J. Geophys. Res.*, *107*, 2002a.

Stohl, A., S. Eckhardt, C. Foster, P. James, N. Spichtinger, and P. Seibert, A replacement for simple back trajectory calculations in the interpretation of atmospheric trace substance measurements, *Atmos. Environ.*, *36*, 4635–4648, 2002b.

Stohl, A., M. Trainer, T. B. Ryerson, J. S. Holloway, and D. D. Parrish, Export of NO_y from the North American boundary layer during 1996 and 1997 North Atlantic Regional Experiments, *J. Geophys. Res.*, *107*, 4132, 2002c.

Stohl, A., O. Cooper, R. Domah, F. Fehsenfeld, C. Forster, E.-Y. Hsie, G. Hübler, D. Parrish, and M. Trainer, Forcecasting for a Lagrangian aircraft campaign, *Atmos. Chem. and Phys.*, *4*, 1113–1124, 2004a.

Stohl, A., O. Cooper, and P. James, A cautionary note on the use of meteorological analysis fields for quantifying atmospheric mixing, *Journal of Atmospheric Sciences*, *61*, 1446–1453, 2004b.

Stohl, A., C. Forster, A. Frank, P. Seibert, and G. Wotawa, Technical note: The Lagrangian particle dispersion model FLEXPART version 6.2, *Atmos. Chem. and Phys.*, *5*, 2461–2474, 2005.

Stohl, A., et al., A backward modeling study of intercontinental pollution transport using aircraft measurements, *J. Geophys. Res.*, *108*, 4370, 2003.

Tanner, D., D. Helmig, J. Heuber, and P. Goldan, Gas chromatography system for the automated, unattended, and cryogen-free monitoring of C2 to C6 non-methane hydrocarbons in the remote troposphere, *J. of Chromatography A*, *1111*, 76–88, 2006.

Thompson, A., Criteria for the selection of stochastic models of particle trajectories in turbulent flows, *J. Fluid Mech.*, *180*, 529–556, 1987.

- Thompson, A., K. Pickering, R. Dickerson, W. E. Jr., D. Jacob, J. Scala, W. Tao, D. McNamara, and J. Simpson, Convective transport over the central United States and its role in regional CO and ozone budgets, *J. Geophys. Res.*, *99*, 18,703–18,711, 1994.
- Traub, M., et al., Chemical characteristics assigned to trajectory clusters during the MINOS campaign, *Atmos. Chem. and Phys.*, *3*, 459–468, 2003.
- Trickl, T., O. R. Cooper, H. Eisele, P. James, R. Mücke, and A. Stohl, Intercontinental transport and its influence on the ozone concentrations over Europe: Three case studies, *J. Geophys. Res.*, *108*, 2003.
- Val Martín M., R. E. Honrath, R. C. Owen, and K. Lapina, Large-scale impacts of anthropogenic pollution and boreal wildfires on the nitrogen oxides over the central North Atlantic region, *J. Geophys. Res.*, *113*, 2008a.
- Val Martín M., R. E. Honrath, R. C. Owen, and Q. B. Li, Seasonal variation of nitrogen oxides in the central North Atlantic lower free troposphere, *J. Geophys. Res.*, *113*, 2008b.
- Vukićević, T., and P. Hess, Analysis of tropospheric transport in the Pacific Basin using the adjoint technique, *J. Geophys. Res.*, *105*, 7213–7230, 2000.
- Zhang, L., et al., Ozone-CO correlations determined by the TES satellite instrument in continental outflow regions, *J. Geophys. Res.*, *33*, 2006.

Appendix A

A users guide for FLEXPART as implemented at MTU

A.1 Introduction

This document details the implementation of the FLEXPART transport model at Michigan Technological University (MTU). The details include:

- The code modifications necessary to run FLEXPART version 6.2 and 8.0 on the redhat@Linux environment at MTU using the g77 and GFortran compilers (including some debugging information)
- The code modifications made to obtain special output fields that are useful for folding (e.g., *Owen and Honrath [2009]*)
- The IDL programs created to run FLEXPART and process and plot the output, including a general description of the approach used by these program to handle the often large output file
- A brief discussion of the meteorological data and the libraries required by FLEXPART to read them
- The IDL program created to load, plot, and process the FLEXPART output

The information provided here should be used in conjunction with the FLEXPART documentation, *Stohl et al. [2005]* and *Eckhardt et al. [2008]*, which provides details

on many aspects of the technical and theoretical operation of the model. Chapter 2 [Owen and Honrath, 2009] and Seibert and Frank [2004] are also useful for understanding the theoretical operation of the model.

A.2 Running FLEXPART at MTU

FLEXPART is a Fortran program. However, IDL is used as the control program. FLEXPART settings are specified in IDL programs and IDL is used to write all the required input and Fortran control programs, compile the Fortran code, execute the resulting Fortran program, and process all the output files. This section details the IDL programs used to run FLEXPART and process its output.

The basic operation of FLEXPART (i.e., aside from any controlling programs like IDL) is composed of three separate components that are largely independent of one another: The Fortran FLEXPART code, the meteorological files, and the GRIB decoder library, which is used as by FLEXPART to read the meteorological fields. This section details the primary IDL programs written to compile and run the FLEXPART Fortran code and process the output on the Linux systems at MTU. This section also includes a general description of the approach for organizing the processing and final data products. The meteorological files and GRIB decoder library are also briefly discussed.

The operation of the forward and backward modes has been separated, such that one set of programs handles the forward mode and another set of programs handles the backward mode. The description given here of the IDL programs are thus divided into two sections, one for the forward and one for the backward modes.

Two separate IDL programs are provided for running the forward and backward modes of FLEXPART and a set of directories have been created for the Fortran FLEXPART files and the meteorological files (linked to the original meteorological files). The Fortran directories are

/local/reh/rcowen/group_flex_idl/REH_group_FLEXPART#/,

and the meteorological directories are

/local/reh/rcowen/group_flex_idl/ECMWF#/,

The directory number (#) is manually selected for the forward runs with directories 10-15 being reserved for the forward runs, while the directory number is automatically selected for backward runs with directories 1-9 reserved for the backward runs (see below for more information). This is the variable “index” in the IDL programs that control the forward and backward FLEXPART simulations.

A.2.1 Forward simulations

The control and processing of the forward simulations have been split into 3 parts. The first part compiles and writes all the files needed to run FLEXPART and executes the resulting Fortran executable file. The second part processes the resulting output files (formatted ASCII text) into intermediate binary files (created by the IDL 'SAVE' command). The third part combines the intermediate IDL binary files into the final IDL binary files and puts them into the appropriate directory. This three-part process was chosen for two reasons. First, the output files are excessively large. This could be remedied by altering the output options to give sparse (as opposed to regular, gridded) data. However, sparse output is not always the most compact form. In fact, the native FLEXPART 6.2 code will switch between sparse and gridded binary files. The gridded ASCII files were chosen to simplify processing the output. The second reason for the two step process is that it allows for the output files to be processed while the run is proceeding, thus reducing the overall time it takes to obtain the final output files. Under the present system, processing the output takes 40-60% of the time it takes to complete the model run. Since typical forward runs take days to weeks to complete, parallel processing of the output is a significant time advantage. Alternate settings (e.g., saving more age classes or saving output more frequently) could significantly increase the time required to process the output, enhancing the advantage of this three part process. The three part process, however, has two significant disadvantages. First, the process requires more oversight from the user. This leads to the second disadvantage, which is a greater opportunity for user error. The processing programs have been designed to reduce the opportunity for error, but they have not been entirely eliminated.

Forward FLEXPART runs should span a minimum of a month, though a run spanning several months at a time is ideal. There are two reasons for these requirements and recommendations. First, the final IDL binary files contain a full month's worth of data. The processing programs are not designed to append the existing final files with the data currently being processed and will instead prompt the user to over write the existing files. Second, the forward mode requires model spin-up time, to allow the full range of the selected age classes to be released into the model. This spin-up time is equal to the maximum age of the tracer in the model, typically 20 days. Thus, to obtain the required one month of final data, the model will need to run for an extra 20 days. However, to obtain 6 months of consecutive months of final data, the model will still need to only be run for 20 extra days, decreasing the relative amount of model spin-up time required.

A.2.1.1 Part 1: Running FLEXPART

The IDL program *run_forward_flexpart* is the primary FLEXPART control program and handles the first stage of the FLEXPART run. This program compiles and

writes all the files necessary to run FLEXPART and executes the resulting Fortran executable. The required input parameters for this program specifies the source (e.g., North American anthropogenic) and species (e.g., CO) to be modeled (with emissions based on the EDGAR Fast Track inventory [*Olivier et al.*, 1996, 1999]) and the time period for the simulation. Optional inputs allow for a custom emissions file (as opposed to the default EDGAR-based emissions), selection of the number of particles, the ability to use instantaneous or averaged output, and the option to use the averaging kernel. The actions taken by this program include (Note: items marked with an asterisk (*) are described in more detail in the FLEXPART documentation *Stohl et al.* [2005]):

- Write pathnames file.*
- Move wind files, write AVAILABLE file.*
- Write COMMAND file.*
- Write OUTGRID file.*
- Write RELEASES.*
- Write AGECLASSES file.*
- Write makefile
- Compile FLEXPART Fortran code and execute resulting Fortran executable file (thus starting the FLEXPART run)

Full documentation on the IDL program can be found using *doc_library*. The following example call of the program runs an anthropogenic CO simulation with emissions from North America during the whole 2003 measurement period.

```

starttr = julday(12, 10, 2002, 00) ;Start with 20-day spin-up
endr = julday(1, 1, 2004, 00) ;End 1 output time after last output period
region = 1 ;1=US
index = '2' ;Directory number where the Fortran programs will be compiled, the
met files stored, and the output files written to
spec = ['25'] ;Anthropogenic CO
maxpt = 4.75e6 ;Number of particles. See section 3.3.3 for a discussion on the
appropriate number of particles to use in model simulations.
instant = 0 ;use standard average output
no_kernel = 0 ;use the standard averaging kernel

```

```

run_forward_flexpart, starttr, endr, region, index, spec, $
    emiss_source = source_file, $
    use_pre_release = rel_file, $
    maxpt = maxpt, $
    DO_NOT_run_flex = DO_NOT_run_flex, $
    instant = instant, $
    no_kernel = no_kernel

```

A.2.1.2 Part 2: Initial ASCII output to intermediate files

Using the default settings, output will be saved to `/local/reh/rcowen/flex_forward_running_output/processing#/,` where `#` is given in the index keyword, 2 in the case above. Standard output will include concentration (e.g., `grid_conc_20030827000000.txt`), mixing ratio (e.g., `grid_pptv_20030827030000.txt`), and flux files (e.g., `grid_flux_20030827120000.txt`). Before processing the output files, the spin-up files should first be removed. In the example above, this would mean deleting all December 2002 files as these do not contain a full 20 days of emissions (the default maximum age class). Output files are processed by the program *read_forward_flex_output*, which reads in the ASCII files and creates temporary IDL binary files. The keywords `go_conc`, `go_pptv`, and `go_flux` determine which type of output files are to be processed. All the file types could be processed in parallel on different machines to speed the processing time (see section A.2.5). The intermediate IDL binary files are saved to the same directory containing the original ASCII FLEXPART output files. Intermediate concentration files are full grid (360 by 90), with one file saved for each output level and each output time (e.g., `2003083100_temp_gridded_conc_00300.idlsav` for the 0-300 m output level). Mixing ratios files contain only a small grid around Pico and thus have only a single file for each output time, containing a 5 by 7 by (number of output levels) array (e.g., `2006083112_temp_ppb_near_pico.grid.idlsav`). Two flux files are created for each output time, one for the standard fluxes (e.g., `2003083115_temp_fluxes.idlsav`) and one for the more detailed, special fluxes (`2003083115_temp_special_fluxes.idlsav`). The standard fluxes contain 'curtains' of the flux, covering large areas (e.g. one curtain contains the flux through the curtain along the -70 degree meridian from 20 deg N to 60 deg N in the EASTWARD direction). The special fluxes contain the eastward fluxes for a number of smaller curtains spanning from the US to Europe (5 degree in longitude, one for each output level). A sample call of the program is listed below.

Once the final, month-long files have been created by *convert_forward_temps_to_gridded_final*, the intermediate files will need to be manually removed from the `processing#` directory. Due to the large number of files, they will typically need to be deleted in subsets (e.g., the flux files for one month, followed by the remaining files that month).

A.2.2 Backward simulations

The basic control, compilation, and processing of the backward mode of FLEXPART is quite similar to the forward mode. One program is used to compile and run FLEXPART (*run_retro_flex*), another program is used to read the ASCII output and convert it into intermediate files (*read_retro_flex_output*) and another converts the intermediate files into the final IDL binary files (*convert_retro_temps_to_gridded_final*). However, since only a limited number of backward releases (i.e., retroplumes) can be initiated within a single FLEXPART simulation (due to memory limitations) and backward simulations run for a limited number of days (typically 20), the size of the FLEXPART ASCII output files are more manageable than those for the forward simulations. As a result, a single program is used to call the three control and processing programs listed above, which only requires the user to select the date range and type of removal to be used (this avoids requiring the user to manually process the output, as was required with the forward output).

Two programs exist to run the suite of programs listed above, one for 'dry' retroplumes (with no removal, *doit_dry_auto_retro*) and one for 'wet' retroplumes (with the removal rate of NO₂, *doit_wet_auto_retro*). These two programs will loop over successive calls of each of the three programs, running and processing several days of retroplumes with each loop. The variable `delta_days` determines the number of days included in each FLEXPART run; a higher number increases the overall speed, though due to memory limitations, this is limited to a maximum of about 5 days. These programs use default settings for the standard Pico group retroplumes (1 hour releases every 3rd hour) and require the user to enter the date ranges. These programs automatically select a temporary output directory, process the ASCII and intermediate output files, create the final output files and move them to the appropriate directory, and delete all the ASCII and intermediate files.

No sample calls of the program are given here, as they are commented fairly well and *doit_dry_auto_retro* and *doit_wet_auto_retro* do not have any input of keyword parameters.

A.2.3 Meteorological data

Most versions of FLEXPART are designed to use ECMWF meteorological data. Version 3.2 and 6.4 support GFS data, version 8.0 supports both GFS and ECMWF, and versions 3.1 and 6.4 support MM5 data. We exclusively use ECMWF data, which has been provided for use with the Pico Mountain station by Diamantino Henriques of the Portuguese Meteorological Institute. Scripts to download the ECMWF data are provided on the FLEXPART website (http://zardoz.nilu.no/%7Eandreas/flextra/ecmwf_routines_V2.tar.gz) and (http://zardoz.nilu.no/%7Eandreas/flextra/flex_extract_ecgate.tar).

GFS, or Global Forecast System, data are, as the name applies, forecast data, and as such, is not ideal for running historic simulations. Instead, analyzed meteorological fields derived from processing observations) are preferred. The analyzed version of GFS is FNL, or Final Global Analysis. Initially, I attempted to use FNL data to run FLEXPART, as it was available for download for free on-line (at least, the data from the previous year, older data had to be purchased). However, I had problems with the files I downloaded being incomplete (levels of data missing). At the time, I believe I was using what is known as ds083.2, which is global met data with 1 by 1 degree resolution, 26 vertical levels, and output every 6 hours, which can be found at <http://dss.ucar.edu/datasets/ds083.2/>.

A.2.4 GRIB decoder

GRIBdd Binary (GRIB) is a data format that has been standardized by the World Meteorological Organization. It is commonly used as the format for meteorological data, including those used by FLEXPART. GRIB decoders are available from ECMWF and UCAR, though I have only had success with those available from ECMWF. The ECMWF GRIB decoder library is named GRIBEX, though there is also GRIB API and EMOSLIB, two groups of libraries which include the GRIBEX decoder.

All three have automated installers that detect the operating system and prompt for additional options (e.g., 32 or 64-bit installations). Theoretically, any version of GRIBEX (or GRIB API or EMOSLIB) that supports the meteorological files (there are GRIB versions 1 and 2, though I believe the meteorological data is version 1) should work with either installation of FLEXPART (6.2 or 8.0). However, I have found that the libraries need to be recompiled for each new version of FLEXPART and for each operating system. I have frequently had difficult compiling the libraries and have only been successful with GRIBEX, though I was able to compile EMOSLIB well enough that the GRIBEX portion worked so that FLEXPART was able to read the meteorological files. I have not been able to determine the cause of the compilation problems and have generally taken a shotgun approach,

trying different versions of the three libraries until one compiled well enough for FLEXPART to use it. The only thing I have learned is that, despite running on a 64-bit machine, the 64-bit compilation options do not work. It's possible this is actually a Fortran compilation problem on the FLEXPART side (e.g., missing a compilation flag needed for 64-bit applications) or that machines are only running the 32-bit version of redhat. A novice Fortran user (like myself) should be able to get a working set of libraries, an experienced Fortran user should have little difficulty.

Two versions of GRIBEX are currently being used, 310 (for FLEXPART 6.2) and 350 (for FLEXPART 8.0). The existing installations are located in /local/reh/rcowen/libs/gribex_000310_RHEL5_64B/, and /local/reh/rcowen/libs/gribex_000350/, respectively. The locations of these libraries need to be written in the FLEXPART makefile, which is handled by the IDL programs.

A.2.5 Computational considerations

This section briefly outlines several computational considerations relevant to running FLEXPART and processing the output, including CPU and memory usage, network limitations, and approximate running and processing times.

FLEXPART is not a threadable program, meaning it can only utilize a single CPU at one time. Thus, each FLEXPART simulation will utilize 100% of each CPU available on the host machine. The memory requirements for each forward FLEXPART simulation is roughly 1 GB and roughly 10 MB for each release in a backward simulations. (Note that the 10 MB in backward simulations are for particle tracking, an additional 400 MB of memory is required for each backward simulation regardless of the number of releases and particles, for storage of the meteorological data, output arrays, etc.). Based on these two computational limitations, the computer called *flexruns*, which has 4 CPUs and 8 GB of ram, can easily accommodate 4 simultaneous FLEXPART simulations. The greatest limitation in this situation would be the speed of the network connection (i.e., reading and writing to the network storage). The meteorological data files are roughly 62 MB each and the output file sizes can range from 100 to 600 MB, depending on the settings. The network connection for *flexruns* is 100 Mbps and can generally handle 2 forward runs without overtaxing the connection, some I/O wait will be experienced with 3 forward runs, and moderate I/O wait will be experienced with 4 forward runs. Using IDL to process the forward output can only utilize a single CPU and requires only minimal memory, however, since the data processing is all reading and writing to the network storage, it can also tax the network connection. *oz* has 2 CPUs, 3.3 GB of memory, and is on a 100 Mbps and can easily handle 2

forward or backward simulations, though this will slow any other users or processes being run on the machine. The *rehgroup* computers also have 2 CPUs, but have only 2 GB of memory and are on 10 Mbps connections. These can do 1 forward or backward simulation with some I/O wait, but 2 simulations tend to overtax the network connection. A good approach for maximizing the number of forward simulations is to run 3 FLEXPART simulations on *flexruns* and use 1 CPU each on *oz* and the *rehgroup* computers to process the output. The demands for backward simulations are generally less than the forward runs (the processing of the output is done after the run and not parallel as in the forward runs) and it's not unreasonable to run 4 backward simulations on *flexruns* one or two on *oz* (depending on the demand from other users and processes, FLEXPART could be reniced to lower the priority) and possibly one on each *rehgroup* computer.

Aside from I/O wait considerations, the total running time for a FLEXPART simulation is determined by the CPU speed. The four computers mentioned above have 3.0 Ghz processors, *flexruns* has a Xeon class processor (typically used for servers), while the other three have Pentium D processors (typically used for personal desktop computers). Identical FLEXPART simulations will likely run slightly faster on *flexruns*, then equally on the other three (if no other processes limiting the memory on the *rehgroup* computers).

A number of the FLEXPART settings can affect the total run time. Larger *ifine* and *ctl* values decrease the internal model time step, linearly increasing the processing time (if *ctl* and *ifine* are doubled, the simulation will take twice as long). A larger number of particles and more frequent sampling of the concentration fields require additional processing on the CPU and thus will also increase the time to complete a simulation. The frequency of output, the size of the output grid, the number of age classes, and the number of output fields chosen will increase the time it takes to write the output files, increasing the time spent writing to the disks. Using the standard settings for the Pico simulations (see section A.5), a forward simulation will take 4-5 days of processing time for a 1-month model simulation (not including processing the output, which can be done as the simulation proceeds). For backward simulations, 3 days of retrorplumes (8 retrorplumes/day) run in a single simulation will require 6 hours, including time to process the output files.

A.3 Custom modifications to the FLEXPART Fortran code

A number of modifications were made to the FLEXPART Fortran code, both to correct bugs that prevented the program from running on the MTU computer systems and for custom modifications to augment the output for use at MTU. This

section details these modifications.

A.3.1 FLEXPART version 6.2 modified files

Each of the files modified to run FLEXPART 6.2 are stored in /home/darcy/rcowen/rco_research/flex_dirs/modified/, while the original, unmodified source code is stored in /home/darcy/rcowen/rco_research/flex_dirs/FLEXPART_source/. Each time FLEXPART is run with the above programs, the unmodified program files are copied to the running directory (see above), then the modified programs are copied over the unmodified ones. All modifications should be accompanied with a comment within the modified file, the comment string should begin with 'RCO' and include a brief description of the modification.

A.3.1.1 FLEXPART bug fixes

calcmatrix.f

Line 50, replaced “nuvz” with nconvlev+2”

Line 73, changed “nconvlev” to “nconvlev+1”

These modifications were made in conjunction with/by approval from Andreas Stohl and Caroline Forester to fix some bugs in the code that had apparently only been apparent on our system. It was not determined why these cause a crash at MTU and had not been caught earlier, perhaps compiler options had suppressed this error with other users.

A.3.1.2 FLEXPART modifications for folding

timemanager_instant.f

Lines 182-187 modified to only calculate concentrations (conccalc) at the output time in order to give instantaneous concentrations (and mixing ratios). New code reads:

```
if (itime.eq.loutend) then
  weight=1.0
  outnum=outnum+weight
  call conccalc(itime,weight)
endif
```

conccalc_no_kernel.f

Line 139, changed “nspecpointer(1)=npoint(i)” to “nspecpointer(1)=1”, which for an unknown reason, was causing a crash with the no-kernel option.

Line 172-176 and 181-201 commented out in order to prevent the kernel from distributing each particle’s mass across multiple grid cells.

A.3.1.3 FLEXPART modifications for easier processing at MTUconcoutput.f

Lines 175-192 and 205-209 commented out to force gridded (as opposed to sparse) output.

Line 283, 286, 393, “form=’formatted’” added for formatted output.

Lines 304-341 and 398-434 commented out to prevent the wet and dry deposition fields from being saved.

Lines 345-367 and 437-465 edited for formatted output and commented out uncertainty output (lines 366-367 and 462-465)

fluxoutput.f

Lines 45-46 and 157-237 edited for formatted output

Lines 54-117 commented output to force gridded output

makefile.L_64-REHL5

The FC, FFLAGS and LDFLAGS have been modified for the g77 compiler and the location of the local GRIBEX decoder.

includepar* Several includepar files are present in the modified directory. Generally, a few lines in includepar must be modified for each FLEXPART run, while the remainder will remain the same. *run_retro_flex* and *run_forward_flexpart* both write a few sections of the includepar file (to the temp.txt and temp1.txt files location in the modified directory) and use the Linux “cat” command to piece together a complete includepar file. The following lines of the original includpar file (in the FLEXPART_source) directory often need to be modified.

Lines 99-100 will need to be changed if the dimensions of the meteorological data change.

Lines 149-150 need to be change to accommodate the chosen output gird dimensions.

Lines 169-170 need to be changed to accommodate the the number of release points.

readwind.f

Line 77 changed to print the name of the meteorological file currently being read.

writeheader.f

All output lines modified for formatted output.

A.3.2 FLEXPART version 8.0 modified files

A.3.2.1 FLEXPART bug fixes

FLEXPART.f

Line 145, changed 'eq' to 'eqv'.

includecom

Line 443, added '*1' after integer.

makefile

Removed compile of 'readspecies', as it's no longer called in any program and there was a persistent crash I couldn't fix during compilation.

concoutput.f

Lines 281, 316, 354, 400, 438, 476, changed 'eq' to 'eqv'.

timemanager.f

Line 454, changed 'eq' to 'eqv'.

outgrid_init.f

Line 178, changed 'eq' to 'eqv'.

outgrid_init_nest

Line 182, added 'then' statement after 'if (ldirect.gt.0)'.

concoutput_nest.f

Line 258, 293, 331, 337, 415, 453, changed 'eq' to 'eqv'.

readwind.f, gridcheck.f, gridcheck_nests.f, readwind_nests.f

Changed grib_api_fortran.h to grib_api_f77.h.

TEMPORARY:readland.f

Replaced new version of readland.f with old version of readland.f, copied old landuse.asc file to local options dir, replaced surfdata.t with version from 6.2.

TEMPORARY:includepar.f

Line 195, changed numclass from 13 to 9 to reflect temporary change above.

readspecies.f

Line 120, Changed 'err' to 'end'.

A.4 IDL programs for processing and plotting FLEXPART

This section details the IDL programs that have been made for loading, processing, and plotting the standard FLEXPART products. The name and general purpose of each program is given here, however, the full documentation on each program is provided within the program and can be accessed with *doc_library*. These programs are stored in

/home/darcy/rcowen/rco_research/flex_dirs/pros/group_share/.

Additional programs can be found in

/home/darcy/rcowen/rco_research/flex_dirs/pros/,

however, these have not all been developed for general use and may not have sufficient documentation and some of the keywords may not work.

A.4.1 Programs for loading data

load_flex_gridded_data - This program can be used to load the gridded concentration fields from the forward FLEXPART runs.

load_flex_pico_ppb_data - This program can be used to load the small grid (centered on the Pico Mountain observatory) of mixing ratios from the forward FLEXPART runs.

load_flex_retro_data - This program can be used to load the retroplume data.

A.4.2 Programs for processing data

get_avg_flex_height - This program will return the average height of an input grid along either the t, x (lon), or y(lat) dimension.

get_flex_density - This program will return either the average height or average age at each horizontal location (latitude and longitude) of an input grid.

get_flex_height_density - This program sums the input grid along two dimensions, one horizontal direction (either latitude or longitude) and either the other horizontal direction or in time, returning the a 2-D grid (height and either latitude, longitude, or time) of the input density. (The standard forward output is density with ng/m^2 . The standard retroplume output, however, is specific volume weighted residence time, or SVWRT, with units of s kg m^{-3} , so not actually a density).

get_retro_folded_conc - This program combines the SVWRT from the retroplumes with either a CO or NO₂ emissions inventory to determine the concentration in the receptor cell from emissions from various regions.

A.4.3 Programs for plotting data

make_std_4_panel_retro_plot - This program can be used to load and plot standard retroplume plots which include the height distribution, total column horizontal distribution, the average age at each horizontal location, and a source contribution map. The program asks for a start and end date and creates plots from all available retroplumes in between the start and end dates.

oplot_density_field2 - This program will plot gridded data scaled according to a linear or log scale onto a plot window. This program will work with multiple windows in the plot page (i.e., !p.multi=[0,2,3]). It uses the IDL procedure BYTSCL and TV to scale and plot the colored grid.

oplot_density_field - This program will plot gridded data scaled according to a linear or log scale onto a plot window. This program will work with multiple windows in the plot page (i.e., !p.multi=[0,2,3]). It uses the IDL procedure CONTOUR to draw the color contours. *oplot_density_field2* is the preferred plotting procedure, as it does not use any interpolation (resultant file sizes are also typically smaller in version 2).

oplot_distf - This program will plot gridded data onto a map, with each grid being colored according to a log or linear color scale. This program will work with multiple windows in the plot page (i.e., !p.multi=[0,2,3]). It uses the BYTSCL procedure to select the colors for the grid cells and the map_image procedure to draw the grids on a map.

color_key_for_dist - This program draws the color scale for the oplot programs above.

Since it is not intended to be used alone, it is not commented as thoroughly as the other programs in the `group_share` directory.

A.5 FLEXPART settings for standard MTU products

This section details the settings used for the standard products archived at MTU.

A.5.1 Retroplumes

Retroplumes were initiated once every 3 hours (i.e., 00 UT, 03 UT, 06 UT...) with a release duration of 1 hour, centered on the hour (from 11:30 to 12:30 UT). For runs with wet removal, the species was 32 (Air tracer with wet removal rate equal to NO₂, see section A.5.4 for a more detailed explanation) and 24 (Air tracer) for runs without wet removal (see the SPECIES file provided with the FLEXPART code for more information on the species). These settings were used for retroplumes used in *Lapina et al.* [2008], *Val Martín et al.* [2008b], and *Val Martín et al.* [2008a].

Command file options:

- `t_output` = 6 hours (output frequency. COMMAND file uses seconds, which would be 21600)
- `t_average` = `t_output` (time average of output. COMMAND file uses seconds, which would be 21600)
- `sample_rate` = 0.25 hours (sample rate of output. COMMAND file uses seconds, which would be 900)
- `part_split` = '9999999'
- `sync_time` = 0.25 hours (synchronization interval of FLEXPART. COMMAND file uses seconds, which would be 900)
- `ctl` = '10'
- `ifine` = '10'
- `ipout` = '0'
- `iout` = '1'
- `LSUBGRID` = '1' ;(1 = on, 0 = off)

- LCONVECTION = '1' ;(1 = on, 0 = off)
- LAGESPECTRA = '1' ;(1 = on, 0 = off)
- IPIN = '0' ;(1 = on, 0 = off)
- IFLUX = '0' ;(1 = on, 0 = off)
- MDOMAINFILL = '0' ;(1 = on, 0 = off)
- IND_SOURCE = '1'
- IND_RECEPTOR = '2'
- MQUASILAG = '0' ;(1 = on, 0 = off)
- NESTED_OUTPUT = '0' ;(1 = on, 0 = off)

OUTGRID options:

- OUTLONLEFT = (-179)
- OUTLATLOWER = (0)
- NUMXGRID = (360)
- NUMYGRID = (90)
- DXOUTLON = 1.0
- DYOUTLAT = 1.0
- vlevels = [300, 1000, 1500, 2000, 2500, 3000, 4000, 5000, 7500, 10000, 15000]

RELEASES file options:

- back_box = [-29., -28., 38., 39., 2000, 2500]
- back_num_particles = 4000 for each release

A.5.2 Retroplume exceptions

The retroplumes used in *Honrath et al.* [2008] had an temporal output resolution of 3 hours (as opposed to the standard 6 hours). A sub-set of these retroplumes were run for 40 days backward in time.

Retroplumes were created for folding with GEOS-CHEM simulations. These retroplumes used instantaneous output, the vertical resolution was 250 m from the surface up to 15 km, the vertical and horizontal resolution was 2.5° by 2.5° , and output was saved at a 1-hour resolution.

A.5.3 Forward runs

Forward simulations included anthropogenic CO (species 25) and NO_x, with and without removal (species 23 and 29, respectively), biomass burning CO emissions (species 22), and a stratospheric O₃ tracer (species 2). The anthropogenic runs used 4.75e6 particles, the O₃ runs used 6e6 particles, the below ground smoldering (BGS) biomass burning runs used 4.5e6 particles, and the above ground flaming (AGF) biomass burning runs used 3e6 particles. The number of particles for the anthropogenic, O₃ and BGS runs were selected to roughly maximize the number of particles that could be used on our computer systems and the number of particles for the AGF runs were selected to such that the mass per particle was roughly equal between the BGS and AGF runs. These settings were used for forward model simulations used in *Lapina et al.* [2008], *Val Martín et al.* [2008b], and *Val Martín et al.* [2008a].

Command file options:

- `t_output = 3` hours (output frequency. COMMAND file uses seconds, which would be 21600)
- `t_average = t_output` (time average of output. COMMAND file uses seconds, which would be 21600)
- `sample_rate = 0.25` hours (sample rate of output. COMMAND file uses seconds, which would be 900)
- `part_split = '9999999'`
- `sync.time = 0.25` hours (synchronization interval of FLEXPART. COMMAND file uses seconds, which would be 900)
- `ctl = '2'`
- `ifine = '6'`
- `iout = '3'`
- `ipout = '0'`
- `LSUBGRID = '1'` ;(1 = on, 0 = off)
- `LCONVECTION = '1'` ;(1 = on, 0 = off)
- `LAGESPECTRA = '1'` ;(1 = on, 0 = off)
- `IPIN = '0'` ;(1 = on, 0 = off)
- `IFLUX = '1'` ;(1 = on, 0 = off)

- MDOMAINFILL = '0' ;(1 = on, 0 = off)
- IND_SOURCE = '1'
- IND_RECEPTOR = '1'
- MQUASILAG = '0' ;(1 = on, 0 = off)
- NESTED_OUTPUT = '0' ;(1 = on, 0 = off)

OUTGRID options:

- OUTLONLEFT = (-179)
- OUTLATLOWER = (0)
- NUMXGRID = (360)
- NUMYGRID = (90)
- DXOUTLON = 1.0
- DYOUTLAT = 1.0
- vlevels = [300, 1000, 1500, 2000, 2500, 3000, 4000, 5000, 7500, 10000, 15000]

A.5.4 Recommended settings

These settings should be suitable for (a) retroplumes to be combined with emissions then aged with OH for NMHC calculations (e.g., *Helmig et al.* [2008]), (b) transport analysis using folded retroplume derived from folding flex+flex (see *Owen and Honrath* [2009]), and (c) forward simulations to be used to analyze the large scale transport (see *Owen et al.* [2006]). Note, however, that these settings may require large amounts of storage space and/or significantly long running and processing times and may thus only be suitable for shorter simulations.

A number of FLEXPART species should be used for both forward and backward runs. Backward runs would use species 24 (air tracer) for “dry” runs (runs without any removal). “Wet” runs would need a new species which uses the same settings for air tracer, with the wet removal parameters (wet deposition parameters “A” and “B” in the SPECIES file) for NO₂ (A = 1.0E-05 and B = 0.62). Species settings for forward runs would vary based on the species being considered and the version of FLEXPART (if available, anthropogenic emissions should use the SPECIES with daily and weekly emissions variations). CO with emissions variation (species 25) should be used for anthropogenic CO. “Dry” anthropogenic NO_x should also use NO₂ emissions with variation (species 23). For the “wet” NO_x runs, a new

species should be made, using the emissions variation for NO₂ (species 23) and with the wet deposition for NO₂ (species 4). This requires making a new chemical in the SPECIES file and adding a new EMISSION_VARIATION_###.dat file (copy EMISSION_VARIATION_023.dat with the new species number).

COMMAND file options (listed in order used in the COMMAND file). Settings match for both forward and backward simulations with the exception of IOUT, IND_SOURCE, and IND_RECEPTOR. Both settings for these parameters are listed, with the backward setting listed first.

- LDIRECT = -1/1 (simulation direction, depends on model run)
- start of simulation (depends on model run)
- end of simulation (depends on model run)
- output frequency = 10800 or 21600 seconds (3 or 6 hours)
- time average of output = 10800 or 21600 seconds (match output frequency)
- sampling rate of output = 900 seconds
- time constant for particle splitting = '9999999' (do not split)
- synchronization interval of FLEXPART = 900 seconds
- CTL = 10 (controls the internal model time step)
- IFINE = 10 (controls the internal model time step for vertical motion)
- IOUT = 1/3 (output selection. 1 for bwd runs for SVWRT, 3 for fwd runs for both concentration and mixing ratios)*
- IPOUT = 0 (do not dump particles)
- LSUBGRID = 1 (use subgrid terrain effect parameterization)
- LCONVECTION = 1 (use convection parameterization)
- LAGESPECTRA = 1 (save age spectra)
- IPIN = 0 (do not continue run using particle dump)
- IFLUX = 0/1 (output fluxes - 0 for bwd, 1 for fwd)**
- MDOMAINFILL = 0 (do not use the domain filling trajectory option)
- IND_SOURCE = 1/1 (1 for bwd, 1 for fwd. 1 indicates source is mass units)* * *

- IND_RECEPTOR = 1/2 (1 for bwd, 2 for fwd. 1 indicates receptor is mass units the. 2 indicates receptor is mixing ratios).***
- MQASILAG = 0 (do not use the “QUASILAGRANGIAN” mode to track individual particles)
- NESTED_OUTPUT = (do not use nested output)

* Note that for folding, the forward concentration fields should be combined with the retroplumes. Thus, concentrations for the whole northern hemisphere (NH) should be saved, but individual age classes may not be necessary, as the older tracer tends to be insignificant when compared to the primary transport pathway (see *Owen and Honrath* [2009]). Since the mixing ratio for the whole NH are not needed for folding, the whole NH grid need not be archived permanently. It may be preferable to archive the mixing ratio on a small field near Pico (or Summit), but keeping the whole age spectrum.

** Note that the whole NH fluxes may not be needed. In the past, a subset of flux curtains were saved as opposed to the flux for the whole NH. It is questionable whether or not even this subset of fluxes will be used, so forward fluxes could potentially be eliminated entirely.

*** Note that the settings for IND_SOURCE and IND_RECEPTOR for the backward runs give output units of $\text{s m}^3 \text{ kg}^{-1}$ (SVWRT) and ng m^{-1} for the forward concentration output files and pptm for mixing ratio files.

OUTGRID options. Settings would match for both forward and backward runs.

- OUTLONLEFT = -179
- OUTLATLOWER = 0
- NUMXGRID = 360
- NUMYGRID = 90
- DXOUTLON = 1.0
- DYOUTLAT = 1.0
- vertical levels = 300, 500, 1000, 1500, 2000, 2500, 3000, 3500, 4000, 4500, 5000, 6000, 7000, 8000, 9000, 10000, 11000, 12000, 13000, 14000, 15000

RELEASES file options for backward runs initiated at Pico:

- back_box = [-29., -28., 38., 39., 2000, 2500]
- back_num_particles = 4000 for each release

The RELEASES file options for forward run all used a set number of particles, which were selected to maximize the number of particles the machines could handle. For anthropogenic runs (or any forward run with constant emissions), this was 4.75e6 particles in the model at any given time. This resulted in roughly 280 kg/particle for NO_x runs and 1100 kg/particle for the CO runs. Stratospheric O₃ simulations use 6e6 particles, which resulted in a varying amount of particles, determined by the location of the tropopause.

RELEASES file for backward runs initiated at Pico:

- back_box = [-29., -28., 38., 39., 2000, 2500]
- back_num_particles = 4000 for each release

The recommended age classes for use in the AGECLASSES file options for anthropogenic (and biomass burning, if applicable) forward runs are (3, 4, 5, 6, 7, 8, 9, 10, 11, 12, 14, 16, 18, 20) days and (1, 2, 3, 4, 5, 6, 7, 8, 9, 10, 12, 14, 16, 18, 20) days for the stratospheric O₃ tracer.

Appendix B

IDL programs for running, processing, and plotting FLEXPART and products

This section contains the IDL programs described in Appendix A.1 that were written to control, process, and plot FLEXPART and products. These programs are included as attachments, similar to an email. A right-click on the paper-clip icon should open a new menu with options to open or save the attached program locally.

B.1 Plotting programs

This section contains all the programs created for plotting FLEXPART output.

B.1.1 `color_key_for_dist.pro`

The program *color_key_for_dist.pro*, which can be used to plot a color scale on many color-contoured or color-shaded plots, is required by all of the plotting programs I have developed. The program, however, was initially authored by Paul Ricchiazzi of the Earth Space Research Group, UCSB, and has been modified and edited to the current form. The original base program is named *color_key.pro* and is part of the *idlmteo* program. In order to avoid any potential copyright issues, it is not

included as part of this dissertation.

B.1.2 `make_std_4_panel_retro_plot.pro`



```

03/19/09                                make_std_4_panel_retro_plot.pr
PRO make_std_4_panel_retro_plot, st_time, en_time, fname, source, lev, $
ev, $                                     lim = lim, encaps = encaps, wet_retros = we
t_retros, $                               custom_dir = custom_dir

;+
; NAME:
;
;     make_std_4_panel_retro_plot
;
; PURPOSE:
;
;     This program will make a multi-page ps file of retroplumes,
;     starting from st_time, ending at en_time. Each page will have
;     the following plots from the SVWRT: latitude vs longitude,
;     height vs time, average age (lat vs lon), footprint (SVWRT
;     folded w/ emissions, lat vs lon).
;
; CATEGORY:
;
;     FLEXPART plotting.
;
; CALLING SEQUENCE:
;
;     make_std_4_panel_retro_plot, st_time, en_time, fname, source, $
;                                     species, lev, lim = lim
;
; INPUTS:
;
;     st_time - the time for the first retroplume in julian days.
;
;     en_time - the time for the last retroplume in julian days.
;
;     fname - file name to be used for the ps file.
;
;     species - spec values are: 1 - Anth CO, 2 - Anth NOx, 3 - NA
;                                     Boreal CO, 4 - Siberian CO, 5 - Air
;                                     tracer (MVM calculated)
;
;     source - source values are: for 1 & 2 above:
;                                     1- North America, 2 - Europe, 3 -
;                                     Asia (Siberia for BB), 4 - South
;                                     America, 5 - Africa, 6 -
;                                     Australia, 7 - World
;
;                                     for 3 & 4 above:
;                                     1 - AG F, 2 - AG S, 3 - BG F, 4 -
;                                     BG S, 5 - all sources
;
;                                     for 5 above:
;                                     NA-1, EU-2, AS-3, SA-4, AF-5, AU-6,
;                                     AR-7, TR-8, ST-9
;
;     lev - levels to use for folding emissions. levels should be
;     a 2 elements array, [st_level, en_level], specifying
;     the number of the corresponding level (e.g., 0 =
;     lowest level).
;
; OPTIONAL INPUTS:
;
;     lim - limits for the plot map. default is [15, -140, 55, -15]
;           [lat1,lon1,lat2,lon2].
;

```

```

;      wet_retros - when set, the standard pico wet retroplumes will be load
ed as
;                  opposed to the standard pico dry retroplumes
;
;      custom_dir - used to specify retroplumes other than the
;                  standard pico wet or dry retroplumes
;
; KEYWORD PARAMETERS:
;
;
;
; OUTPUTS:
;
;
;
; EXAMPLE:
;
;
; MODIFICATION HISTORY:
;
;-

dummy = LABEL_DATE( DATE_FORMAT = '%N %D!C%Z')
mydevice = !D.NAME
SET_PLOT, 'PS'
DEVICE, FILENAME = fname+'.ps', /color, $
        xsize = 10, ysize = 8, /inches, encapsulated = encaps
!P.MULTI = [0, 2, 2, 0, 0]
@set_tek_color
IF (keyword_set(lim)) THEN limits = lim ELSE limits = [15, -140, 55, -15]
min_dis = 0.001
barwd = 0.1

get_time = st_time
WHILE (get_time LE en_time) DO BEGIN

    caldat, get_time, mo, da, yr, hr
    IF (mo LE 9) THEN mo = '0'+strtrim(string(mo), 2) ELSE mo = strtrim(strin
ng(mo), 2)
    IF (da LE 9) THEN da = '0'+strtrim(string(da), 2) ELSE da = strtrim(strin
ng(da), 2)
    IF (hr LE 9) THEN hr = '0'+strtrim(string(hr), 2) ELSE hr = strtrim(strin
ng(hr), 2)
    yr = strtrim(string(yr), 2)
    print, mo, ' ', da, ' ', yr, ' ', hr

    load_flex_retro_data, get_time, $
                        rp_grid, jdr_grid, xr_grid, yr_grid, zr_grid, $
                        levels = zg_grid, $
                        time_first = 0, $
                        wet_remov = wet_retros, $
                        custom_dir = custom_dir

;make plot 1 - h vs t, non-normalized
get_flex_density, rp_grid, rp_h_dens, $
                lons = lons, lats = lats, $
                normalize = normalize, $
                height_grid = height_grid, $
                age_grid = age_grid

tttitle = 'rp for '+yr+' '+mo+' '+da+' '+hr

```

```

oplot_density_field2, transpose(rp_h_dens), $
    jdr_grid, zr_grid/1000., $
    pct_ct = pct_ct, $
    Absolute = absolute, $
    min_display = min_dis, $
    max_val = max(rp_h_dens, /nan), $
    thetitle = tttitle, $
    add_legend = 1, $
    use_label_date = '%N!C%D', $
    xunit = 'Days', xtickint = 2, $
    chsz = chsz, chtk = chtk, bchsz = bchsz, $
    xadj = xadj, yadj = yadj, $
    ybarsz = ybarsz, barwd = barwd, $
    xscale = 0.9, yscale = 0.9

sz = size(rp_grid)
rp_grid2 = make_array(sz(4), sz(1), sz(2), sz(3))
FOR loop_re = 0, sz(4)-1 DO $
    rp_grid2(loop_re, *, *, *) = rp_grid(*, *, *, loop_re)

get_avg_flex_height, rp_grid2, zr_grid, rp_h_avgs, $
    lons = lons, lats = lats, gridded = gridded

@set_tek_color
oplot, jdr_grid, rp_h_avgs(0, *)/1000., psym = -1, thick = 2, color = black
ack
oplot, jdr_grid, (rp_h_avgs(0, *)-rp_h_avgs(1, *)^(0.5))/1000., $
    psym = -1, linestyle = 2, symsize = .5, color = black
oplot, jdr_grid, (rp_h_avgs(0, *)+rp_h_avgs(1, *)^(0.5))/1000., $
    psym = -1, linestyle = 2, symsize = .5, color = black

;plot xy total column
x = (indgen(xr_grid(2))+xr_grid(1))*xr_grid(3)
y = (indgen(yr_grid(2))+yr_grid(1))*yr_grid(3)
xmin = limits(1)
ymin = limits(0)
xmax = limits(3)
ymax = limits(2)

xy_rp_grid = total(total(rp_grid, 4), 3)
ttitel = 'max val '+strtrim(String(max(xy_rp_grid)), 2)
OPLOT_DISTf, xy_rp_grid, x, y, xmin, ymin, xmax, ymax, $
    pct_ct = pct_ct, $
    Absolute = 0, $
    min_display = min_dis, $
    max_val = max(xy_rp_grid, /nan), $
    thetitle = tttitle, $
    chtk = chtk, chsz = chsz, $
    add_legend = 1, bchsz = bchsz, $
    force_min = force_min, old_colors = old_colors, $
    bilin = bilin, map_cent = map_cent, $
    lat_del = lat_del, lon_del = lon_del, $
    add_loc = add_loc, xadj = xadj, yadj = yadj, $
    ybarsz = ybarsz, barwd = barwd, barfmt = barfmt, $
    bar_unit_mult = bar_unit_mult, $
    xscale = 0.9, yscale = 0.9

;plot avg age
n_days = 20
avg_age = total(rp_grid, 3)

```

03/19/09

make_std_4_panel_retro_plot.pr

```
st_pos = n_elements(avg_age(0, 0, *))
IF ((st_pos-n_days*4-2) LT 0) THEN spos = 0 ELSE spos = (st_pos-n_days*4-2)
-2)
stop_t = st_pos-(n_days*4)-1
temp_avg_age = make_array(360, 90)
FOR loopn = stop_t, (st_pos-1) DO BEGIN
    temp_avg_age = temp_avg_age + (avg_age(*, *, loopn) * (((st_pos-loopn-1)*6.)/24.))
ENDFOR

avg_age = temp_avg_age/total(avg_age, 3, /nan)

OPLOT_DISTf, avg_age, x, y, xmin, ymin, xmax, ymax, $
    pct_ct = pct_ct, $
    Absolute = 1, $
    min_display = 1.0, $
    max_val = max(avg_age, /nan), $
    thetitle = thetitle, $
    chtk = chtk, chsz = chsz, $
    add_legend = 1, bchsz = bchsz, $
    force_min = force_min, old_colors = old_colors, $
    bilin = bilin, map_cent = map_cent, $
    lat_del = lat_del, lon_del = lon_del, $
    add_loc = add_loc, xadj = xadj, yadj = yadj, $
    ybarsz = ybarsz, barwd = barwd, barfmt = barfmt, $
    bar_unit_mult = bar_unit_mult, $
    xscale = 0.9, yscale = 0.9

;plot footprint contribution
get_retro_folded_conc, rp_grid, jdr_grid, xr_grid, yr_grid, $
    zr_grid, species, source, concentration, $
    levels = lev, gridded = 1
footprint_grid = total(total(concentration, 1), 1)
tttitle = 'total conc = '+strtrim(string(total(concentration)), 2)
OPLOT_DISTf, footprint_grid, x, y, xmin, ymin, xmax, ymax, $
    pct_ct = pct_ct, $
    Absolute = 0, $
    min_display = min_dis, $
    max_val = max(footprint_grid, /nan), $
    thetitle = tttitle, $
    chtk = chtk, chsz = chsz, $
    add_legend = 1, bchsz = bchsz, $
    force_min = force_min, old_colors = old_colors, $
    bilin = bilin, map_cent = map_cent, $
    lat_del = lat_del, lon_del = lon_del, $
    add_loc = add_loc, xadj = xadj, yadj = yadj, $
    ybarsz = ybarsz, barwd = barwd, barfmt = barfmt, $
    bar_unit_mult = bar_unit_mult, $
    xscale = 0.9, yscale = 0.9

    get_time = get_time+double(0.125)
enDWHILE

device, /close

return
end
```

B.1.3 oplot_density_field2.pro



```

PRO oplot_density_field2, count, x, y, $
  pct_ct = pct_ct, $
  Absolute = absolute, $
  min_display = min_display, $
  max_val = max_val, $
  thetitle = thetitle, $
  add_legend = add_legend, $
  use_label_date = use_label_date, $
  xunit = xunit, xtickint = xtickint, $
  chsz = chsz, chtk = chtk, bchsz = bchsz, $
  xadj = xadj, yadj = yadj, $
  ybarsz = ybarsz, barwd = barwd, $
  xscale = xscale, yscale = yscale, $
  xrnge = xrnge, yrnge = yrnge

;+
; NAME:
;
;       oplot_density_field2
;
; PURPOSE:
;
;       This program will plot gridded data scaled according to a
;       linear or log scale onto a plot window. This program
;       will work with multiple winds in the plot page (i.e.,
;       !p.multi=[0,2,3]). It uses the idl procedure bytscl and tv to
;       scale and plot the colored grid.
;
; CATEGORY:
;
;       Plotting.
;
; CALLING SEQUENCE:
;
; INPUTS:
; count - an x by y array containing the values of the input
;        data. the output will be color scaled (either linear
;        or log) according to these values.
;
;        x - the values along the x axis corresponding to the first
;        dimension of count. the dimension of x must be
;        identical to the first dimension of count
;
;        y- the values along the y axis corresponding to the second
;        dimension of count. the dimension of y must be
;        identical to the second dimension of count
;
; KEYWORD PARAMETERS:
;
;        pct_ct - custom labels for color scale. default is [0,
;        0.2,..., 1.0] for absolute and every power of 10 for
;        log scale.
;
;        absolute - turns on absolute scaling. the default is to use
;        log scaling.
;
;        min_display - the lowest percentage be displayed. everything
;        less than max_val*min_display will be colored
;        white. the default is 0.01 (1%).
;
;        max_val - the maximum value to be displayed. everything larger
;        than the maximum value will be colored the same as

```



```

;           the maximum value. units at the same as count. the
;           default is max(count).
;
;   thetitle - title for the plot window. the default is no title.
;
;   add_legend - when set, the plot will include the color scale
;               (on the right hand side of the plot). the default
;               is no color scale.
;
;   use_label_date - if set, this keyword will use the date
;                   formatting for the x axis. the format is
;                   specified by this keyword (i.e.,
;                   use_label_date = '(%M/%d)'). If set, xunit
;                   and xtickint must also be set.
;
;   xunit - unit for xtickunits. typically 'Days'
;
;   xtickint - interval for xticks. If xunit= 'Days' and xtickint
;             = 2, then an x tick will be drawn every 2 days.
;
;   chsz - charsize for plot (as opposed to color scale)
;
;   chtk - charthick for plot (as opposed to color scale)
;
;   bchsz - charsize for color scale.
;
;   xadj - stretch or shrink the default x plot window. units are
;         decimal fraction of plot window, format is [left
;         stretch, right stretch]. by default, the x plot region is
;         [0.12, 0.9].
;
;   yadj - stretch or shrink the default y plot window. units are
;         decimal fraction of plot window, format is [bottom
;         stretch, top stretch]. by default, the y plot region is
;         [0.125, 0.925]
;
;   ybarsz - y size of the color scale. units are in plot units
;           (i.e., pixels?). the default will make the color bar
;           the same size of the plot.
;
;   barwd - with of the color bar. units = ?. default = 0.25.
;
;   xscale - factor to shrink or stretch the x plot window. units
;           are decimal fraction, default = 1.0 (100%). this is a
;           quicker way to make the plot bigger or smaller than xadj.
;
;   yscale - factor to shrink or stretch the y plot window. units
;           are decimal fraction, default = 1.0 (100%). this is a
;           quicker way to make the plot bigger or smaller than yadj.
;
;   xrnge = custom x range for the plot. the default is [min(x), max(x)]
;
;   yrnge = custom y range for the plot. the default is [min(y), max(y)]
;
;
; OUTPUTS:
;
;
;
; EXAMPLES:
;
;

```

```

;-

;determine the position for a multi page plot
t_window = !p.multi(0)
n_col = !p.multi(1)
n_row = !p.multi(2)
n_windows = !p.multi(1)*!p.multi(2)
;determine which row and column we are to plot in
;determine which row and column we are to plot in
IF (!p.multi(4) EQ 0) THEN BEGIN
    rows = make_Array(n_col, n_row)
    cols = rows
    rowf = ((-1)*indgen(n_row))+(n_row-1)
    colf = indgen(n_col)

    FOR loopr = 0, n_row-1 DO cols(*, loopr) = colf
    FOR loopc = 0, n_col-1 DO rows(loopc, *) = rowf
    IF (t_window EQ 0) THEN pos = 0 ELSE pos = n_windows-t_window
ENDIF ELSE BEGIN
    rows = make_Array(n_row, n_col)
    cols = rows
    rowf = ((-1)*indgen(n_row))+(n_row-1)
    colf = indgen(n_col)
    FOR loopr = 0, n_row-1 DO cols(loopr, *) = colf
    FOR loopc = 0, n_col-1 DO rows(*, loopc) = rowf
    IF (t_window EQ 0) THEN pos = 0 ELSE pos = n_windows-t_window
ENDELSE

x_start = cols(pos)*(1./n_col)
y_start = rows(pos)*(1./n_row)

IF (keyword_set(xscale)) THEN xscal = xscale ELSE xscal = 1.0
IF (keyword_set(yscale)) THEN yscal = yscale ELSE yscal = 1.0

del_x = (float(fix((1./n_col)*50.))/50.)*xscal
del_y = (float(fix((1./n_row)*50.))/50.)*yscal
IF (keyword_Set(xadj)) THEN xshift = xadj ELSE xshift = [0, 0]
IF (keyword_Set(yadj)) THEN yshift = yadj ELSE yshift = [0, 0]
xregionp = ([0.12, 0.9]+xshift)*del_x+x_start
yregionp = ([0.125, 0.925]+yshift)*del_y+y_start

maxcolor = !d.n_colors
loadct, 5
common colors, r_orig, g_orig, b_orig, r_curr, g_curr, b_curr

IF (keyword_set(old_colors)) THEN BEGIN
    g_curr(0) = 200
    b_curr(0) = 200
    r_curr(0) = 200
ENDIF ELSE BEGIN
    g_curr(0) = 255
    b_curr(0) = 255
    r_curr(0) = 255
ENDELSE
tv1ct, r_curr, g_curr, b_curr
error_code = !VALUES.F_NAN

IF (NOT keyword_set(max_val)) THEN maximum_value = max(count) ELSE $
    maximum_value = max_val
IF (keyword_set(use_label_date)) THEN BEGIN
    dummy = LABEL_DATE(DATE_FORMAT = [use_label_date])

```

```

        xdate = 'LABEL_DATE'
        unit = xunit
        xtint = xtickint
    ENDIF ELSE BEGIN
        xdate = ''
        unit = ''
        xtint = ''
    ENDELSE

    IF keyword_set(xrng) THEN x_range = xrng ELSE x_range = [min(x), max(x)]
    IF keyword_set(yrng) THEN y_range = yrng ELSE y_range = [min(y), max(y)]

    plot, x, y, title = thetitle, charthick = chtk, $
        charsize = chsz, XTICKFORMAT = xdate, xtickunits = [unit], $
        /nodata, color = 1, ystyle = 1+4+8, xstyle = 1+4+8, $
        xrange = x_range, yrange = y_range, $
        position = [xregionp(0), yregionp(0), xregionp(1), yregionp(1)]

;*****

    IF (keyword_set(max_val)) THEN result = max_val ELSE $
        result = max(count, /nan)

;find min dx and dy in input array bc we need to them to be equal
;between data points in the output image
    delx = make_array(n_elements(x)-1, /double)
    FOR lx = 1, n_elements(x)-1 DO delx(lx-1) = x(lx)-x(lx-1)

    dely = make_array(n_elements(y)-1, /double)
    FOR ly = 1, n_elements(y)-1 DO dely(ly-1) = y(ly)-y(ly-1)

    dx = min(abs(delx)) & dy = min(abs(dely))
    nx = fix((max(x)-min(x))/dx)+1 & ny = ((max(y)-min(y))/dy)+1

    img = fltarr(nx, ny) ;define array: img
    new_count = fltarr(nx, ny)
;fill new_count with old count values, floor function
    FOR loopx = double(0), nx-1 DO BEGIN
        FOR loopy = double(0), ny-1 DO BEGIN
            iox = where(x GE (min(x)+(dx*loopx)))
            ioy = where(y GE (min(y)+(dy*loopy)))
            iox = min(iox) & ioy = min(ioy)
            new_count(loopx, loopy) = count(iox, ioy)
        endFOR
    endFOR

    count = new_count
; Scale count

    IF (keyword_set(min_display)) THEN $
        minimum_displayed = min_display $
    ELSE $
        minimum_displayed = 0.01

    IF (NOT KEYWORD_SET(ABSOLUTE)) THEN BEGIN ; logarithmic scaling
        img = alog10(count > minimum_displayed(0)*result)
        img = bytscl(img, top = maxcolor, max = alog10(result))
        ltm = WHERE((count LE minimum_displayed*result) OR $
            (count EQ error_code))
        IF (ltm(0) NE (-1)) THEN img(ltm) = maxcolor
    ENDIF ELSE BEGIN ; Then it's ABSOLUTE SCALING
        IF (keyword_set(force_min)) THEN BEGIN

```

```

        miok = where(count LE minimum_displayed*max(count))
        count(miok) = 0
    ENDIF
    max_GT_max_val = where(count GT max_val)
    IF (max_GT_max_val(0) NE (-1)) THEN count(max_GT_max_val) = 0
    min_n_zero = where((count GT 0) AND (count NE error_code))
    IF ((n_elements(min_n_zero) LT n_elements(count)) AND $
        (min_n_zero(0) NE (-1))) THEN $
        min_non_zero = min(count(min_n_zero)) ELSE $
        min_non_zero = min(count)
    img = (count > min_non_zero)
    img = bytscl(img, top = maxcolor, /NAN)

    ENDELSE ; ABSOLUTE SCALING

; Now show the image
px = !x.window*!d.x_vsize
py = !y.window*!d.y_vsize
xl = px(0)
dx = px(1)-px(0);+!d.x_ch_size*4
yl = py(0)
dy = py(1)-py(0);-!d.y_ch_size*4
tv, img, xl, yl, xsize = dx, ysize = dy, /device

;redraw the axis, since they will probably have been overplotted by
;the image

axis, xl, yl, xaxis = 0, $
    charthick = chtk, charsize = chsz, $
    XTICKFORMAT = xdate, color = 1, xstyle = 1, $
    /save, /device, xrange = x_range, $
    xtickunits = [unit], xtickinterval = xtint
axis, xl, yl, yaxis = 0, $
    charthick = chtk, $
    charsize = chsz, color = 1, ystyle = 1, $
    /save, /device, yrange = y_range

;*****

    IF (keyword_set(add_legend)) THEN begin
; labeled intervals on legend (1.0=100%)
        labels = double([minimum_displayed])
        loop_lab = 0.
        loop_cnt = 1.
        WHILE (loop_lab LT 1) DO BEGIN
            labels = [labels, minimum_displayed*10^loop_cnt]
            loop_lab = minimum_displayed*10^loop_cnt
            loop_cnt = loop_cnt+1.
        ENDWHILE
        linear_labels = [0, .2, .4, .6, .8, 1.0] ; for linear
; MUST keep the initial value = 0 and final = 1.0 in labels and no other 0's
!
; temp = count array values at 0%, 5%, ... 100%
        IF (NOT KEYWORD_SET(absolute)) THEN BEGIN ; logarithmic scaling
            IF (KEYWORD_SET(pct_ct)) THEN BEGIN
                labels = pct_ct
            ENDIF
            temp = labels * maximum_value
            temp(0) = minimum_displayed * maximum_value
; spread to 255 as max

```

```

temp3 = bytscl(aalog10(temp), top = maxcolor)
; temp4 = fraction of height of color key for each label
; = logarithmic % of count arrays.
temp4 = float(temp3)/max(temp3, /nan)
temp4(0) = 0 ; because we define all values below min=col
or 0
color_key_positions = temp4
ENDIF ELSE BEGIN ; absolute scaling
IF (KEYWORD_SET(pct_ct)) THEN BEGIN
labels = pct_ct
color_key_positions = pct_ct
ENDIF ELSE BEGIN
color_key_positions = linear_labels
labels = (maximum_value*linear_labels)/100.
ENDELSE
ENDELSE

IF (NOT keyword_set(max_val)) THEN max_val = max(count, /nan)
IF ((minimum_displayed LE (10e-4)) OR (max_val GE 10e4)) then $
formatt = '(E7.1)' ELSE formatt = '(f7.1)'

IF (keyword_Set(barwd)) THEN barwidth = barwd ELSE barwidth = 0.25
COLOR_KEY_FOR_DIST, maxcolor, color_key_positions, labels, $
maximum_value, $
range = [0., 1.], $
barwidth = barwidth, ysize = ybarsz, $
frmt = formatt, charsize = bchsz

ENDIF

RETURN
END

```

B.1.4 oplot_density_field.pro



```

PRO oplot_density_field, count, x, y, $
    pct_ct = pct_ct, $
    Absolute = absolute, $
    min_display = min_display, $
    max_val = max_val, $
    thetitle = thetitle, $
    add_legend = add_legend, $
    xtitl = xtitl, ytitl = ytitl, $
    use_label_date = use_label_date, $
    xunit = xunit, xtickint = xtickint, $
    chsz = chsz, chtk = chtk, $
    bchsz = bchsz, bchtk = bchtk, $
    c_fill = c_fill, f_fill = f_fill, $
    xadj = xadj, yadj = yadj, $
    ybarsz = ybarsz, barwd = barwd, barfmt = barfmt, $
    xscale = xscale, yscale = yscale, $
    old_colors = old_colors, custom_ct = custom_ct, $
    bw_colors = bw_colors, $
    bartitle1 = bartitle1, bartitle2 = bartitle2

;+
; NAME:
;
;       oplot_density_field
;
; PURPOSE:
;
;       This program will plot gridded data scaled according to a
;       linear or log scale onto a plot window. This program
;       will work with multiple winds in the plot page (i.e.,
;       !p.multi=[0,2,3]). It uses the idl procedure contour to draw
;       the color contours. oplot_density_field2 is the preferred
;       plotting procedure, as it does not use any interpolation
;       (resultant file sizes are also typically smaller in version
;       2).
;
; CATEGORY:
;
;       Plotting.
;
; CALLING SEQUENCE:
;
; INPUTS:
; count - an x by y array containing the values of the input
;         data. the output will be color scaled (either linear
;         or log) according to these values.
;
;       x - the values along the x axis corresponding to the first
;         dimension of count. the dimension of x must be
;         identical to the first dimension of count
;
;       y - the values along the y axis corresponding to the second
;         dimension of count. the dimension of y must be
;         identical to the second dimension of count
;
; KEYWORD PARAMETERS:
;
;       pct_ct - custom labels for color scale. default is [0,
;         0.2,..., 1.0] for absolute and every power of 10 for
;         log scale.
;
;       absolute - turns on absolute scaling. the default is to use
;         log scaling.

```

```

;
;   min_display - the lowest percentage be displayed. everything
;                   less than max_val*min_display will be colored
;                   white. the default is 0.01 (1%).
;
;   max_val - the maximum value to be displayed. everything larger
;               than the maximum value will be colored the same as
;               the maximum value. units at the same as count. the
;               default is max(count).
;
;   thetitle - title for the plot window. the default is no title.
;
;   add_legend - when set, the plot will include the color scale
;                 (on the right hand side of the plot). the default
;                 is no color scale.
;
;   use_label_date - if set, this keyword will use the date
;                     formatting for the x axis. the format is
;                     specified by this keyword (i.e.,
;                     use_label_date = '%M/%d'). If set, xunit
;                     and xtickint must also be set.
;
;   xunit - unit for xtickunits. typically 'Days'. only works
;           with use_label_date keyword.
;
;   xtickint - interval for xticks. If xunit= 'Days' and xtickint
;               = 2, then an x tick will be drawn every 2
;               days. only workd with use_label_date keyword.
;
;   chsz - charsize for plot (as opposed to color scale)
;
;   chtk - charthick for plot (as opposed to color scale)
;
;   bchsz - charsize for color scale.
;
;   bchtk - charthick for color scale
;
;   c_fill - turns on the cell_fill keyword in contour (see idl
;             help for contour.
;
;   f_fill - turns on the fill keyword in contour (see idl
;             help for contour.
;
;   xadj - stretch or shrink the default x plot window. units are
;           decimal fraction of plot window, format is [left
;           stretch, right stretch]. by default, the x plot region is
;           [0.12, 0.9].
;
;   yadj - stretch or shrink the default y plot window. units are
;           decimal fraction of plot window, format is [bottom
;           stretch, top stretch]. by default, the y plot region is
;           [0.125, 0.925]
;
;   ybarsz - y size of the color scale. units are in Device units
;             (i.e., pixels?). the default will make the color bar
;             the same size of the plot.
;
;   barwd - width of the color bar. units = ?. default = 0.25.
;
;   barfmt - format code for color scale. default is either
;             '(E7.1)' or '(f5.1)' depending on the number of
;             decimal places in the max and min values on the color

```



```

;           scale.
;
;       bartitle1 - title for the color bar, written along the axis
;
;       bartitle2 - title for the color bar, written next to each value
;
;       xscale - factor to shrink or stretch the x plot window. units
;               are decimal fraction, default = 1.0 (100%). this is a
;               quicker way to make the plot bigger or smaller than xadj.
;
;       yscale - factor to shrink or stretch the y plot window. units
;               are decimal fraction, default = 1.0 (100%). this is a
;               quicker way to make the plot bigger or smaller than yadj.
;
;
; OUTPUTS:
;
;
;
; EXAMPLES:
;
;
;-

;determine the position for a multi page plot
t_window = !p.multi(0)
n_col = !p.multi(1)
n_row = !p.multi(2)
n_windows = !p.multi(1)*!p.multi(2)
;determine which row and column we are to plot in
IF (!p.multi(4) EQ 0) THEN BEGIN
    rows = make_Array(n_col, n_row)
    cols = rows
    rowf = ((-1)*indgen(n_row))+(n_row-1)
    colf = indgen(n_col)
    FOR loopr = 0, n_row-1 DO cols(*, loopr) = colf
    FOR loopc = 0, n_col-1 DO rows(loopc, *) = rowf
    IF (t_window EQ 0) THEN pos = 0 ELSE pos = n_windows-t_window
ENDIF ELSE BEGIN
    rows = make_Array(n_row, n_col)
    cols = rows
    rowf = ((-1)*indgen(n_row))+(n_row-1)
    colf = indgen(n_col)
    FOR loopr = 0, n_row-1 DO cols(loopr, *) = colf
    FOR loopc = 0, n_col-1 DO rows(*, loopc) = rowf
    IF (t_window EQ 0) THEN pos = 0 ELSE pos = n_windows-t_window
ENDELSE

x_start = cols(pos)*(1./n_col)
y_start = rows(pos)*(1./n_row)

IF (keyword_set(xscale)) THEN xscal = xscale ELSE xscal = 1.0
IF (keyword_set(yscale)) THEN yscal = yscale ELSE yscal = 1.0

del_x = (float(fix((1./n_col)*50.))/50.)*xscal
del_y = (float(fix((1./n_row)*50.))/50.)*yscal
IF (keyword_Set(xadj)) THEN xshift = xadj ELSE xshift = [0, 0]
IF (keyword_Set(yadj)) THEN yshift = yadj ELSE yshift = [0, 0]
xregionp = ([0.12, 0.9]+xshift)*del_x+x_start
yregionp = ([0.125, 0.925]+yshift)*del_y+y_start

maxcolor = !d.n_colors

```

```

IF (NOT keyword_set(bw_colors)) THEN BEGIN
  IF keyword_set(custom_ct) THEN loadct, custom_ct ELSE loadct, 5
ENDIF ELSE loadct, 0
common colors, r_orig, g_orig, b_orig, r_curr, g_curr, b_curr

IF (keyword_set(old_colors)) THEN BEGIN
  g_curr(0) = 200
  b_curr(0) = 200
  r_curr(0) = 200
ENDIF ELSE BEGIN
  g_curr(0) = 255
  b_curr(0) = 255
  r_curr(0) = 255
ENDELSE

IF (NOT keyword_set(absolute)) THEN count = ALOG10(count)
IF (NOT keyword_set(min_display)) THEN minimum_displayed = 0 ELSE $
  minimum_displayed = min_display
IF (NOT keyword_set(max_val)) THEN maximum_value = max(count) ELSE $
  maximum_value = max_val
IF (keyword_set(use_label_date)) THEN BEGIN
  dummy = LABEL_DATE(DATE_FORMAT = [use_label_date])
  xdate = 'LABEL_DATE'
  unit = xunit
  xtint = xtckint
ENDIF ELSE BEGIN
  xdate = ''
  unit = ''
  xtint = xtckint
ENDELSE

contour, (count), x, y, nlevels = 255, c_colors = indgen(255), $
  closed = 1, cell_fill = c_fill, fill = f_fill, $
  min_value = minimum_displayed, $
  max_value = maximum_value, $
  XTICKFORMAT = xdate, color = 1, $
  yrange = [0, max(y)], xrange = [min(x), max(x)], $
  xstyle = 1, ystyle = 1, $
  ytickformat = '(I6)', title = thetitle, $
  xtitle = xtitl, ytitle = ytitl, $
  charsize = chsz, charthick = chtk, $
  xticks = floor((max(x)-min(x))/2), $
  position = [xregionp(0), yregionp(0), xregionp(1), yregionp(1)], ↵
$
  xtckunits = [unit], xtckinterval = xtint

IF (keyword_set(add_legend)) THEN begin
; labeled intervals on legend (1.0=100%)
  labels = double([minimum_displayed])
  loop_lab = 0.
  loop_cnt = 1.
  WHILE (loop_lab LT 1) DO BEGIN
    labels = [labels, minimum_displayed*10^loop_cnt]
    loop_lab = minimum_displayed*10^loop_cnt
    loop_cnt = loop_cnt+1.
  ENDWHILE
  linear_labels = [0, .2, .4, .6, .8, 1.0] ; for linear
; MUST keep the initial value = 0 and final = 1.0 in labels and no other 0's↵
!
; temp = count array values at 0%, 5%, ... 100%
  IF (NOT KEYWORD_SET(absolute)) THEN BEGIN ; logarithmic scaling

```

```

        IF (KEYWORD_SET(pct_ct)) THEN BEGIN
            labels = pct_ct
        ENDIF
        temp = labels * maximum_value
        temp(0) = minimum_displayed * maximum_value
; spread to 255 as max
        temp3 = bytscl(alog10(temp), top = maxcolor)
; temp4 = fraction of height of color key for each label
; = logarithmic % of count arrays.
        temp4 = float(temp3)/max(temp3, /nan)
        temp4(0) = 0 ; because we define all values below min=col
or 0
        color_key_positions = temp4
    ENDIF ELSE BEGIN ; absolute scaling
        IF (KEYWORD_SET(pct_ct)) THEN BEGIN
            labels = pct_ct
            color_key_positions = pct_ct
        ENDIF ELSE BEGIN
            color_key_positions = linear_labels
            labels = (maximum_value*linear_labels)/100.
        ENDELSE
    ENDELSE

    IF (NOT keyword_set(max_val)) THEN max_val = max(count, /nan)
    IF (NOT keyword_Set(barfmt)) THEN BEGIN
        IF ((minimum_displayed LE (10e-4)) OR (max_val GE 10e4)) then $
            formatt = '(E7.1)' ELSE formatt = '(f5.1)'
        ENDIF ELSE formatt = barfmt

    IF (keyword_Set(barwd)) THEN barwidth = barwd ELSE barwidth = 0.25
    IF (keyword_set(bartitle1)) THEN bartitle11 = bartitle1 ELSE bartitle11 = ''
    IF (keyword_set(bartitle2)) THEN bartitle22 = bartitle2 ELSE bartitle22 = ''

    COLOR_KEY_FOR_DIST, maxcolor, color_key_positions, labels, maximum_value
, $
        range = [0., 1.], $
        barwidth = barwidth, ysize = ybarsz, $
        frmt = formatt, charsize = bchsz, $
        charthick = bchtk, $
        title = bartitle11, add_title_to_value= bartitle22

    ENDIF

    RETURN
END

```

B.1.5 oplot_distf.pro



```

PRO OPLOT_DISTF, count, x, y, xmin, ymin, xmax, ymax, $
  pct_ct = pct_ct, $
  Absolute = absolute, $
  min_display = min_display, $
  max_val = max_val, $
  thetitle = thetitle, $
  chtk = chtk, chsz = chsz, $
  add_legend = add_legend, bchsz = bchsz, bchtk = bchtk, $
  force_min = force_min, old_colors = old_colors, $
  bilin = bilin, map_cent = map_cent, $
  lat_del = lat_del, lon_del = lon_del, $
  add_loc = add_loc, xadj = xadj, yadj = yadj, $
  ybarsz = ybarsz, barwd = barwd, barfmt = barfmt, $
  bar_unit_mult = bar_unit_mult, $
  xscale = xscale, yscale = yscale, $
  map_scale = map_scale, bw_colors = bw_colors, $
  azmut = azmut, lamb = lamb, ortho = ortho, sat = sat

;+
; NAME:
;
;       oplot_distf
;
; PURPOSE:
;
;       This program will plot gridded data onto a map, with each grid being
;       colored according to a log or linear color scale. This program
;       will work with multiple winds in the plot page (i.e.,
;       !p.multi=[0,2,3]). It uses the bytscl procedure to select the
;       colors for the grid cells and the map_image procedure to draw
;       the grids on a map.
;
; CATEGORY:
;
;       Plotting.
;
; CALLING SEQUENCE:
;
; INPUTS:
; count - an x by y array containing the values of the input
;         data. the output will be color scaled (either linear
;         or log) according to these values.
;
;       x - the latitude values corresponding to the first dimension
;         of count. the dimension of x must be
;         identical to the first dimension of count
;
;       x - the longitude values corresponding to the first dimension
;         of count. the dimension of y must be identical to the
;         second dimension of count
;
; KEYWORD PARAMETERS:
;
;       pct_ct - custom labels for color scale. default is [0,
;         0.2,..., 1.0] for absolute and every power of 10 for
;         log scale.
;
;       absolute - turns on absolute scaling. the default is to use
;         log scaling.
;
;       min_display - the lowest percentage be displayed. everything

```

```

;               less than max_val*min_display will be colored
;               white. the default is 0.01 (1%).
;
;   max_val - the maximum value to be displayed. everything larger
;             than the maximum value will be colored the same as
;             the maximum value. units are the same as count. the
;             default is max(count).
;
;   thetitle - title for the plot window. the default is no title.
;
;   add_legend - when set, the plot will include the color scale
;               (on the right hand side of the plot). the default
;               is no color scale.
;
;   use_label_date - if set, this keyword will use the 'date'
;                   formatting for the x axis. the format is
;                   specified by this keyword (i.e.,
;                   use_label_date = '(%M/%d)'). If set, xunit
;                   and xtickint must also be set.
;
;   xunit - unit for xtickunits. typically 'Days'. only works with
;          use_label_date keyword.
;
;   xtickint - interval for xticks. If xunit= 'Days' and xtickint
;             = 2, then an x tick will be drawn every 2
;             days. only works with use_label_date keyword.
;
;   chsz - charsize for plot (as opposed to color scale)
;
;   chtk - charthick for plot (as opposed to color scale)
;
;   bchsz - charsize for color scale.
;
;   bchtk - charthick for color scale.
;
;   force_min - force the cells with less than min_display*max_val
;              to be set to zero. i think this is an artifact of
;              old problems and no longer necessary. I don't want
;              to "break" anything, so I am leaving it for now.
;
;   old_colors - use the old color scheme, which filled the "blank"
;               cells with gray. otherwise, "blank cells" will be
;               white.
;
;   bilin - turn on bilinear interpolation (smoothing) between
;           data points (called in map_image).
;
;   map_cent - custom center location for the map. default is
;             [0,0], format is [lat, lon]
;
;   lat_del - delta latitude between lines of latitude drawn on
;             the map. default is 15.
;
;   lon_del - delta longitude between lines of longitude drawn on
;             the map. default is 15.
;
;   add_loc - add an asterix (*) at a select location on the
;            map. format is [lat, lon]
;
;   xadj - stretch or shrink the default x plot window. units are
;          decimal fraction of plot window, format is [left
;          stretch, right stretch]. by default, the x plot region is

```

```

;           [0.12, 0.9].
;
;   yadj - stretch or shrink the default y plot window. units are
;          decimal fraction of plot window, format is [bottom
;          stretch, top stretch]. by default, the y plot region is
;          [0.125, 0.925]
;
;   ybarsz - y size of the color scale. units are in Device units
;            (i.e., pixels?). the default will make the color bar
;            the same size of the plot.
;
;   barwd - width of the color bar. units = ?. default = 0.25
;           (inches?).
;
;   barfmt - format code for color scale. default is either
;            '(E7.1)' or '(f5.1)' depending on the number of
;            decimal places in the max and min values on the color
;            scale.
;
;   bar_unit_mult - shift the color scale up or down? units ?
;
;   xscale - factor to shrink or stretch the x plot window. units
;            are decimal fraction, default = 1.0 (100%). this is a
;            quicker way to make the plot bigger or smaller than
;            xadj.
;
;   yscale - factor to shrink or stretch the y plot window. units
;            are decimal fraction, default = 1.0 (100%). this is a
;            quicker way to make the plot bigger or smaller than
;            yadj.
;
;   xrnge = custom x range for the plot. the default is [min(x), max(x)]
;
;   yrnge = custom y range for the plot. the default is [min(y), max(y)]
;
;   map_scale - scale for map (see idl documentation for map_set,
;               keyword scale).
;
;   bw_colors - make plot in black and white as opposed to color.
;
;   azmut, lamb, ortho, sat - use azimuthal, lambert,
;                             orthographic, or satellite
;                             projection for the map.
;
; OUTPUTS:
;
;
; EXAMPLES:
;
;-

; If called with /absolute, will plot absolute values on a linear scale.
; min_display is minimum fraction of the max value in the array to scale
; color table to (if logarithmic scaling)
; max_val is the maximum color for the scale, useful if you want to
; compare two plots with the same scale

; determine the position for a multi page plot

```

```

t_window = !p.multi(0)
n_col = !p.multi(1)
n_row = !p.multi(2)
n_windows = !p.multi(1)*!p.multi(2)
;determine which row and column we are to plot in
  IF (!p.multi(4) EQ 0) THEN BEGIN
    rows = make_Array(n_col, n_row)
    cols = rows
    rowf = ((-1)*indgen(n_row))+(n_row-1)
    colf = indgen(n_col)
    FOR loopr = 0, n_row-1 DO cols(*, loopr) = colf
    FOR loopc = 0, n_col-1 DO rows(loopc, *) = rowf
    IF (t_window EQ 0) THEN pos = 0 ELSE pos = n_windows-t_window
  ENDF ELSE BEGIN
    rows = make_Array(n_row, n_col)
    cols = rows
    rowf = ((-1)*indgen(n_row))+(n_row-1)
    colf = indgen(n_col)
    FOR loopr = 0, n_row-1 DO cols(loopr, *) = colf
    FOR loopc = 0, n_col-1 DO rows(*, loopc) = rowf
    IF (t_window EQ 0) THEN pos = 0 ELSE pos = n_windows-t_window
  ENDELSE

x_start = cols(pos)*(1./n_col)
y_start = rows(pos)*(1./n_row)

IF (keyword_set(xscale)) THEN xscal = xscale ELSE xscal = 1.0
IF (keyword_set(yscale)) THEN yscal = yscale ELSE yscal = 1.0

del_x = (float(fix((1./n_col)*50.))/50.)*xscal
del_y = (float(fix((1./n_row)*50.))/50.)*yscal
IF (keyword_Set(xadj)) THEN xshift = xadj ELSE xshift = [0, 0]
IF (keyword_Set(yadj)) THEN yshift = yadj ELSE yshift = [0, 0]
xregion = ([0.12, 0.9]+xshift)*del_x+x_start
yregion = ([0.125, 0.925]+yshift)*del_y+y_start

maxcolor = !d.n_colors
IF (NOT keyword_set(bw_colors)) THEN loadct, 5 ELSE loadct, 0
common colors, r_orig, g_orig, b_orig, r_curr, g_curr, b_curr

IF (keyword_set(old_colors)) THEN BEGIN
  g_curr(0) = 200
  b_curr(0) = 200
  r_curr(0) = 200
ENDIF ELSE BEGIN
  g_curr(0) = 255
  b_curr(0) = 255
  r_curr(0) = 255
ENDELSE
tvlct, r_curr, g_curr, b_curr
error_code = !VALUES.F_NAN
error_code2 = !values.f_infinity
;*****
IF (NOT keyword_set(chtk)) THEN chtk = 1
IF (NOT keyword_set(chsz)) THEN chsz = 1

plot, [0, 1], [0, 1], title = thetitle, charthick = chtk, $
  charsize = chsz, $
  /nodata, ystyle = 4+8, xstyle = 4+8, color = 1, $
  position = [xregion(0), yregion(0), xregion(1), yregion(1)], $
  xticks = 1, yticks = 1

```



```

IF (keyword_Set(map_cent)) THEN cent_map = map_cent ELSE $
  cent_map = [0, 0]

IF (NOT keyword_set(map_Scale)) THEN BEGIN
  map_set, cent_map(0), cent_map(1), /mercator, $
  limit = [ymin, xmin, ymax, xmax], $
  color = 1, charsize = 1, /noerase, $
  position = [xregion(0), yregion(0), xregion(1), yregion(1)]
ENDIF ELSE BEGIN
  map_set, cent_map(0), cent_map(1), scale = map_scale, $
  color = 1, charsize = 1, /noerase, $
  position = [xregion(0), yregion(0), xregion(1), yregion(1)], $
  azimuthal = azmut, lambert = lamb, orthographic = ortho, satell
ite = sat
ENDELSE

;*****
IF (keyword_set(max_val)) THEN result = max_val ELSE $
  result = max(count, /nan)

; Scale count
img = fltarr(n_elements(x), n_elements(y)) ;define array: img

IF (keyword_set(min_display)) THEN $
  minimum_displayed = min_display $
ELSE $
  minimum_displayed = 0.01

IF (NOT KEYWORD_SET(ABSOLUTE)) THEN BEGIN ; logarithmic scaling
  img = alog10(count > minimum_displayed(0)*result)
  img = bytscl(img, top = maxcolor, max = alog10(result))
  LT_min = WHERE((count LE minimum_displayed*result) OR $
    (count eq error_code) OR $
    (count eq error_code2))
  IF (LT_min(0) NE (-1)) THEN img(LT_min) = maxcolor
ENDIF ELSE BEGIN ; Then it's ABSOLUTE SCALING
  IF (keyword_set(force_min)) THEN BEGIN
    miok = where(count LE minimum_displayed*max(count, /nan))
    count(miok) = 0
  ENDIF
  IF (NOT keyword_set(max_val)) THEN max_val = max(count, /nan)
  max_GT_max_val = where(count GT max_val)
  IF (max_GT_max_val(0) NE (-1)) THEN count(max_GT_max_val) = 0
  min_n_zero = where((count GT 0) AND (count NE error_code))
  IF ((n_elements(min_n_zero) LT n_elements(count)) AND $
    (min_n_zero(0) NE (-1))) THEN $
    min_non_zero = min(count(min_n_zero)) ELSE $
    min_non_zero = min(count)
  img = (count > min_non_zero)
  img = bytscl(img, top = maxcolor, /NAN)

ENDELSE ; ABSOLUTE SCALING

new_img = map_image(img, startx, starty, xsize, ysize, $
  bilinear = bilin, /WHOLE_MAP, $
  latmin = min(y), latmax = max(y), $
  lonmin = min(x), lonmax = max(x), compress = 1)

; Now show the image
tv, new_img, startx, starty, xsize = xsize, ysize = ysize

```

```

IF (keyword_Set(lat_del)) THEN latd = lat_del ELSE latd = 15
IF (keyword_Set(lon_del)) THEN lond = lon_del ELSE lond = 15

IF (NOT keyword_set(map_Scale)) THEN BEGIN
  map_set, cent_map(0), cent_map(1), /mercator, $
  limit = [ymin, xmin, ymax, xmax], $
  color = 1, charsize = 1, /noerase, $
  position = [xregion(0), yregion(0), xregion(1), yregion(1)]
ENDIF ELSE BEGIN
  map_set, cent_map(0), cent_map(1), scale = map_scale, $
  color = 1, charsize = 1, /noerase, $
  position = [xregion(0), yregion(0), xregion(1), yregion(1)], $
  azimuthal = azmut, lambert = lamb, orthographic = ortho, satell
ite = sat
ENDELSE
map_grid, latdel = latd, londel = lond, latlab = -30., $
lonlab = 30, /label, $
color = 1, glinethick = 2, charsize = .75
map_continents, color = 1, mlinethick = 2, /USA, /continents
;*****

IF (keyword_Set(add_loc)) THEN BEGIN

  FOR loopxy = 0, n_elements(add_loc(*, 0))-1 DO BEGIN
    xyouts, add_loc(loopxy, 0), add_loc(loopxy, 1), $
    '*', color = 200, charsize = 3, charthick = 3, $
    alignment = 0.5
    xyouts, add_loc(loopxy, 0), add_loc(loopxy, 1), $
    '*', color = 1, charsize = 3, charthick = 3.5, $
    alignment = 0.5
  ENDFOR
ENDIF

;*****
IF (keyword_set(add_legend)) THEN BEGIN

; labeled intervals on legend (1.0=100%)
  labels = double([minimum_displayed])
  loop_lab = 0.
  loop_cnt = 1.
  WHILE (loop_lab LT 1) DO BEGIN
    labels = [labels, minimum_displayed*10^loop_cnt]
    loop_lab = minimum_displayed*10^loop_cnt
    loop_cnt = loop_cnt+1.
  ENDWHILE

  linear_labels = [0, .2, .4, .6, .8, 1.0] ; for linear
; MUST keep the initial value = 0 and final = 1.0 in labels and no other 0's
!
  IF (NOT KEYWORD_SET(absolute)) THEN BEGIN ; logarithmic scaling
    IF (KEYWORD_SET(pct_ct)) THEN BEGIN
      labels = pct_ct
    ENDIF
    temp = labels * result
    temp(0) = minimum_displayed * result
; spread to 255 as max
    temp3 = bytscl(aalog10(temp), top = maxcolor)
; temp4 = fraction of height of color key for each label
; = logarithmic % of count arrays.
    temp4 = float(temp3)/max(temp3, /nan)

```

```

temp4(0) = 0 ; because we define all values below min=col
or 0
color_key_positions = temp4
ENDIF ELSE BEGIN ; absolute scaling
IF (KEYWORD_SET(pct_ct)) THEN BEGIN
    labels = pct_ct
    color_key_positions = pct_ct
    print, 'THIS WONT WORK!!!'
    bell & bell & bell
ENDIF ELSE BEGIN
    IF (NOT keyword_set(max_val)) THEN max_val = max(count, /nan)
    IF (NOT abs(max_val) GE 0) THEN max_val = max(count, /nan)
    color_key_positions = linear_labels
    labels = (min_non_zero + (max_val-min_non_zero) * linear_labels)/100
.
    ENDELSE
ENDELSE

;COLOR KEY CALLED HERE*****
IF (NOT keyword_set(max_val)) THEN max_val = max(count, /nan)
IF (NOT keyword_Set(barfmt)) THEN BEGIN
    IF ((minimum_displayed LE (10e-4)) OR (max_val GE 10e4)) then $
        formatt = '(E7.1)' ELSE formatt = '(f5.1)'
ENDIF ELSE formatt = barfmt

IF (keyword_Set(barwd)) THEN barwidth = barwd ELSE barwidth = 0.25
COLOR_KEY_FOR_DIST, maxcolor, color_key_positions, labels, result, $
    range = [0., 1.], $
    barwidth = barwidth, ysize = ybarsz, $
    frmt = formatt, charsize = bchsz, charthick = bchtk,
$
    bar_unit_mult = bar_unit_mult
; , charthick = 4; , title = '% max', ysize = 5200,
; pos = [17100, 700], $

enDIF

; Close the PS output
IF (N_ELEMENTS(psfile) NE 0) THEN BEGIN
    device, /close
    set_plot, 'x'
ENDIF

;advance plot window
IF (t_window NE 0) THEN !p.multi(0) = t_window-1 ELSE $
    !p.multi(0) = n_windows-1

RETURN
END

```

B.2 Programs for loading and manipulating FLEXPART output

This section contains the programs created to load and manipulate the FLEXPART output.

B.2.1 `get_avg_flex_height.pro`



```

PRO get_avg_flex_height, input_grid, delta_level, output_grid, $
    lons = lons, lats = lats, gridded = gridded

;+
; NAME: get_avg_flex_height
;
;
; PURPOSE:
; This program will return the average height of an input grid
; along either the t, x (lon), or y(lat) dimension. Note that
; only 1 keyword can be set (time, lons, lats)
;
; CATEGORY:
; FLEXPART data analysis
;
; CALLING SEQUENCE:
; get_avg_flex_height, input_grid, delta_level, output_grid, $
;                                lons = lons, lats = lats gridded = gridded
; INPUTS:
; input_grid - an array with the dimensions [t, x, y, z]
;
;         delta_level - an array containing the height of each
;                        level. should match the z component of
;                        input_grid.
;
; KEYWORD PARAMETERS:
;     lons - if this keyword is set, the average height will be
;            given as a function of longitude (or x). otherwise,
;            average height will be given as a function of time.
;
;     lats - if this keyword is set, the average height will be
;            given as a function of latitude (or y). otherwise,
;            average height will be given as a function of time.
;
;     gridded - if this keyword is set, the average height will be
;               given as a function of latitude & longitude. otherwise,
;               average height will be given as a function of time.
;
; OUTPUTS:
;
;     output_grid - an array containing the average, sd, max, and
;                   min height as a function of time, longitude, or
;                   latitude, depending on which keyword is set. the
;                   array will be a 4 by n_element(averaging
;                   componenet) array. output_grid(0,*) is the
;                   average height, output_grid(1,*) is the
;                   standard deviation of height, output_grid(2,*)
;                   is the max height, output_grid(3,*) is the min
;                   height.
;
; EXAMPLES:
;
;-

    IF (n_elements(delta_level) NE n_elements(input_grid(0, 0, 0, *))) THEN BE
GIN
    print, 'Error - number of levels in delta level'
    print, 'do not match number of levels in input grid.'
    return
enDIF

```

```

n_t = n_elements(input_grid(*, 0, 0, 0))
n_x = n_elements(input_grid(0, *, 0, 0))
n_y = n_elements(input_grid(0, 0, *, 0))

;calculate the average height of each grid cell using all available dimensions
ns
weighted_levels = input_grid*0.0

;multiply each level's (RT, SVWRT) by the height
IF (NOT keyword_set(griddd)) THEN BEGIN
  FOR loop1 = 0, n_elements(delta_level)-1 DO $
    weighted_levels(*, *, *, loop1) = input_grid(*, *, *, loop1)*delta_level
  el(loop1)
;find the total column (RT, SVWRT)
  total_levels = total(input_grid, 4)

;find the total column (RT, SVWRT)
  average_level = total(weighted_levels, 4)/total_levels
  zero = where(average_level GT 0, COMPLEMENT=non_zero)
  average_level(non_zero) = !VALUES.F_NAN

;default, give the average height as a function of time
IF ((NOT keyword_set(lons)) AND (NOT keyword_set(lats))) THEN BEGIN
  output_grid = make_array(4, n_t)
  FOR loopt = 0, n_t-1 DO BEGIN
;calc Sd for all cells @ each time, excluding cells w/ less than 0.1% of
;the max value
;    lower_95_percential = where(input_grid(loopt, *, *, *) LE $
;                                (0.0001*max(input_grid(loopt, *, *, *)))
;  )
;    upper_95 = input_grid(loopt, *, *, *)
;    upper_95(lower_95_percential) = !VALUES.F_NAN
;    output_grid(*, loopt) = moment(upper_95, /nan)
;    iok = where(average_level(loopt, *, *) GE 0)
;    IF (n_elements(iok) GE 2) THEN begin
;      output_grid(*, loopt) = moment(average_level(loopt, *, *), /nan)
;    endif ELSE BEGIN
;      output_grid(1, loopt) = max(average_level(loopt, *, *), /nan)
;    ENDELSE
;      output_grid(2, loopt) = max(average_level(loopt, *, *), /nan)
;      output_grid(3, loopt) = min(average_level(loopt, *, *), /nan)
;    ENDFOR

;convert the SD from input units to percent
;    output_grid(1, *) = output_grid(1, *)/output_grid(0, *)

;compute average level, weighted by (RT, SVWRT)
  total_wt_level = total(total(total(weighted_levels, 2, /nan), 2, /nan)
, 2, /nan)
  total_level = total(total(total(input_grid, 2, /nan), 2, /nan), 2, /nan)
n)
  output_grid(0, *) = total_wt_level/total_level

;convert the SD from percent to height
;    output_grid(1, *) = output_grid(1, *)*output_grid(0, *)

  endif
endif

;need to fix other options! 11/18/2006

```

```

IF (keyword_set(lons)) THEN BEGIN
    output_grid = make_array(4, n_x)

    FOR loopx = 0, n_x-1 DO BEGIN

        iok = where(average_level(*, loopx, *) GE 0)
        IF (n_elements(iok) GE 2) THEN begin
            output_grid(*, loopx) = moment(average_level(*, loopx, *), /nan)
        endIF ELSE BEGIN
            output_grid(1, loopx) = max(average_level(*, loopx, *), /nan)
        ENDELSE
            output_grid(2, loopx) = max(average_level(*, loopx, *), /nan)
            output_grid(3, loopx) = min(average_level(*, loopx, *), /nan)
        ENDFOR

;compute average level, weighted by (RT, SVWRT)
        total_wt_level = total(total(total(weighted_levels, 1, /nan), 2, /nan)
, 2, /nan)
        total_level = total(total(total(input_grid, 1, /nan), 2, /nan), 2, /na
n)
        output_grid(0, *) = total_wt_level/total_level

    endIF

IF (keyword_set(lats)) THEN BEGIN
    output_grid = make_array(4, n_y)

    FOR loopy = 0, n_x-1 DO BEGIN

        iok = where(average_level(*, loopy, *) GE 0)
        IF (n_elements(iok) GE 2) THEN begin
            output_grid(*, loopy) = moment(average_level(*, loopy, *), /nan)
        endIF ELSE BEGIN
            output_grid(1, loopy) = max(average_level(*, *, loopy), /nan)
        ENDELSE
            output_grid(2, loopy) = max(average_level(*, *, loopy), /nan)
            output_grid(3, loopy) = min(average_level(*, *, loopy), /nan)
        ENDFOR

;compute average level, weighted by (RT, SVWRT)
        total_wt_level = total(total(total(weighted_levels, 1, /nan), 1, /nan)
, 2, /nan)
        total_level = total(total(total(input_grid, 1, /nan), 1, /nan), 2, /na
n)
        output_grid(0, *) = total_wt_level/total_level

    endIF

endIF ELSE BEGIN

    input_grid = total(input_grid, 1)

    FOR loop1 = 0, n_elements(delta_level)-1 DO $
        weighted_levels(*, *, loop1) = input_grid(*, *, loop1)*delta_level(loo
pl)

    total_levels = total(input_grid, 3)

    total_weighted_levels = total(weighted_levels, 3)

```

03/19/09

get_avg_flex_height.pr

```
average_level = total_weighted_levels/total_levels  
  
non_zero = where(average_level GT 0)  
average_level(non_zero) = !VALUES.F_NAN  
  
output_grid = average_level  
  
enDELSE  
  
return  
end
```


B.2.2 `get_flex_density.pro`



```

PRO get_flex_density, input_grid, output_grid, $
    lons = lons, lats = lats, $
    normalize = normalize, $
    height_grid = height_grid, $
    age_grid = age_grid

; ;+
; NAME: get_flex_density
;
;
; PURPOSE:
; This program will return the density by height of an input
; grid along either the t, x (lon), or y(lat)
; dimension. Note that only 1 keyword can be set (time, lons,
; lats). Alternately, it can return an xy density field of
; average age (keyword age_horiz_dens) or an ax density field
; of average height (keyword height_horiz_dens).
;
; CATEGORY:
; FLEXPART data analysis
;
; CALLING SEQUENCE:
; get_avg_flex_height, input_grid, output_grid, $
;                               lons = lons, lats = lats
; INPUTS:
; input_grid - an array with the dimensions [x, y, z, t]
;
;
; KEYWORD PARAMETERS:
;
;     lons - if this keyword is set, the average height will be
;            given as a function of longitude (or x). otherwise,
;            average height will be given as a function of time.
;
;     lats - if this keyword is set, the average height will be
;            given as a function of latitude (or y). otherwise,
;            average height will be given as a function of time.
;
;     normalize - if set, the values at each (time, lon, lat) will
;                 be normalized by the maximum value along at each
;                 index. this keyword is ignored if height_Grid or
;                 age_grid have been set.
;
;     height_grid - if set, an xy density grid of avg. height will
;                   be returned. the height grid is the top of each
;                   flexpart level in the z position of the input
;                   grid. this keyword ignores the normalize
;                   keyword.
;
;     age_grid - if set, an xy density grid of avg. age will be
;                returned. the age_grid is the julday corresponding
;                to each t in the inout grid. this keyword ignores
;                the normalize keyword.
;
;
; OUTPUTS:
;
;     output_grid - an [(t,x,y), z] array containing the density of
;                   the input grid or an [x,y] grid containing the
;                   avg. height or age in each xy cell, depending on
;                   the options set.
;
; EXAMPLES:

```

```

;
;-

IF ((NOT keyword_Set(age_grid)) AND $
    (NOT keyword_Set(height_grid))) THEN BEGIN

    IF ((NOT keyword_Set(lats)) AND (NOT keyword_Set(lons))) THEN BEGIN
        output_grid = total(total(input_grid, 1, /nan), 1, /nan)
    ENDIF ELSE BEGIN
        IF (keyword_set(lons)) THEN BEGIN
            output_grid = total(total(input_grid, 2, /nan), 3, /nan)
        ENDIF ELSE BEGIN
            output_grid = total(total(input_grid, 1, /nan), 3, /nan)
        ENDELSE
    ENDELSE

    IF (keyword_Set(normalize)) THEN BEGIN
        FOR loopn = 0, n_elements(output_grid(*, 0))-1 DO BEGIN
            output_grid(loopn, *) = output_grid(loopn, *)/$
            max(output_grid(loopn, *), /nan)
        ENDFOR
    ENDIF
ENDIF ELSE BEGIN                                ;then its an x-y density grid of age or heig
ht
    IF (keyword_set(age_grid)) THEN BEGIN ;it's avg age

        delt_t = abs(age_grid(0)-age_grid(1))

        sz = size(input_grid)
        output_grid = make_array(sz(1), sz(2))
        temp_grid = total(input_grid, 3)

        FOR loopage = 0, n_elements(age_grid)-1 DO begin
            output_grid(*, *) = output_grid(*, *)+$
            (temp_grid(*, *, n_elements(age_grid)-1-loopage)*$
            (loopage*delt_t))
        ENDFOR

        output_grid = output_grid/total(temp_grid, 3)

    ENDIF ELSE BEGIN                                ;then it's avg height

        delt_z = make_Array(n_elements(height_grid))
        delt_z(0) = height_grid(0)/2 ;assumes the 1st lev starts at ground.

        FOR looplev = 1, n_elements(delt_z)-1 DO $
            delt_z(looplev) = (height_grid(looplev)-height_grid(looplev-1))/2.

        adj_grid = height_grid-delt_z

        sz = size(input_grid)
        output_grid = make_array(sz(1),sz(2))
        temp_grid = total(input_grid, 4)

        FOR loop_z = 0, n_elements(adj_grid)-1 DO $
            output_grid = output_grid+$
            (temp_grid(*, *, loop_z)*adj_grid(loop_z))

        output_grid = output_grid/total(temp_grid, 3)

    ENDELSE

```

03/19/09

get_flex_density.pr

ENDELSE

return
END

B.2.3 `get_flex_height_density.pro`



```

PRO get_flex_height_density, input_grid, output_grid, $
    lons = lons, lats = lats, $
    normalize = normalize, $
    no_time = no_time, $
    time_first = time_first

; ;+
; NAME: get_flex_height_density
;
;
; PURPOSE:
; This program will return the density by height of an input
; grid along either the t, x (lon), or y(lat)
; dimension. Note that only 1 keyword can be set (time, lons,
; lats)
;
; CATEGORY:
; FLEXPART data analysis
;
; CALLING SEQUENCE:
; get_avg_flex_height, input_grid, output_grid, $
;                               lons = lons, lats = lats
; INPUTS:
; input_grid - an array with the dimensions [x, y, z, t]
;
;
; KEYWORD PARAMETERS:
;     lons - if this keyword is set, the average height will be
;            given as a function of longitude (or x). otherwise,
;            average height will be given as a function of time.
;
;     lats - if this keyword is set, the average height will be
;            given as a function of latitude (or y). otherwise,
;            average height will be given as a function of time.
;
;     normalize - if set, the values at each (time, lon, lat) will
;                 be normalized by the maximum value along at each
;                 index.
;
;     no_time - if set, the program assumes that t is not given
;
;     time_first - assumes that time is given first, i.e.,
;                  [t,x,y,z]. this is the typical format for the
;                  forward output, while the default is the typical
;                  format for the backward output from FLEXPART
;
; OUTPUTS:
;
;     output_grid - an [(t,x,y), z] array containing the density of
;                   the input grid, depending on the options set.
;
; EXAMPLES:
;
;-

IF (NOT keyword_set(time_first)) THEN begin:[x, y, z, t]
  IF ((NOT keyword_Set(lats)) AND (NOT keyword_Set(lons))) THEN BEGIN
    output_grid = total(total(input_grid, 1, /nan), 1, /nan)
  ENDIF ELSE BEGIN
    IF (keyword_set(lons)) THEN BEGIN
      IF (keyword_Set(no_time)) THEN BEGIN
        output_grid = total(input_grid, 2, /nan)
      ENDIF
    ENDIF
  ENDIF
ENDIF

```

```

        ENDIF ELSE BEGIN
            output_grid = total(total(input_grid, 2, /nan), 3, /nan)
        ENDELSE
    ENDIF ELSE BEGIN
        IF (keyword_Set(no_time)) THEN BEGIN
            output_grid = total(input_grid, 1, /nan)
        ENDIF ELSE BEGIN
            output_grid = total(total(input_grid, 1, /nan), 3, /nan)
        ENDELSE
    ENDELSE
ENDIF ELSE BEGIN ;[t, x, y, z]
    IF ((NOT keyword_Set(lats)) AND (NOT keyword_Set(lons))) THEN BEGIN
        output_grid = total(total(input_grid, 2, /nan), 2, /nan)
    ENDIF ELSE BEGIN
        IF (keyword_set(lons)) THEN BEGIN
            IF (keyword_Set(no_time)) THEN BEGIN
                output_grid = total(input_grid, 3, /nan)
            ENDIF ELSE BEGIN
                output_grid = total(total(input_grid, 3, /nan), 1, /nan)
            ENDELSE
        ENDIF ELSE BEGIN
            IF (keyword_Set(no_time)) THEN BEGIN
                output_grid = total(input_grid, 2, /nan)
            ENDIF ELSE BEGIN
                output_grid = total(total(input_grid, 2, /nan), 1, /nan)
            ENDELSE
        ENDELSE
    ENDELSE
;    output_grid = transpose(output_grid)
enDELSE

    IF (keyword_Set(normalize)) THEN BEGIN
        FOR loopn = 0, n_elements(output_grid(*, 0))-1 DO BEGIN
            output_grid(loopn, *) = output_grid(loopn, *)/max(output_grid(loopn, *
), /nan)
        enDFOR
    ENDIF

    return
END

```

B.2.4 `get_retro_folded_conc.pro`




```

PRO get_retro_folded_conc, retro_grid, jdr_grid, xr_grid, yr_grid, $
    zr_grid, species, source, concentration, $
    exponential_dist = exponential_dist, $
    levels = levels, n_days = n_days, $
    gridded = gridded, $
    custom_dist = custom_dist, $
    keep_levels = keep_levels, $
    refrm = refrm

;+
; NAME:
; get_folded_retro_conc
;
; PURPOSE:
; folds retroplume values with an emissions grid to calculate
; resultant ppb's @ pico (or other site) from a selected
; emissions source.
;
; CATEGORY:
; Data acquisition
;
; CALLING SEQUENCE:
; get_folded_retro_conc, retro_grid, jd_grid, z_grid, x_grid, $
; y_grid, source, concentration, exponential_dist = exponential_dist, $
; levels = levels, n_days = n_days, gridded = gridded
; INPUTS:
; retro_grid - an [x, y, z, t] grid of the values from FLEXPART
;               retroplume. The expected units are s*m^3/kg -
;               the default format from FLEXPART.
;
;       jd_grid - array of times in julian day, which should match the t
;               component from the retro_grid
;
;       x_grid, y_grid, z_grid - array of x,y,&z values, should match
;               the x,y,&z component from the
;               retro_grid
;
;       species - spec values are: 1 - Anth CO, 2 - Anth NOx, 3 - NA
;               Boreal CO, 4 - Siberian CO, 5 - Air
;               tracer (MVM calculated)
;
;       source - source values are: for 1 & 2 above:
;               1- North America, 2 - Europe, 3 -
;               Asia (Siberia for BB), 4 - South
;               America, 5 - Africa, 6 -
;               Australia, 7 - World
;
;               for 3 & 4 above:
;               1 - AG F, 2 - AG S, 3 - BG F, 4 -
;               BG S, 5 - all sources
;
;               for 5 above:
;               NA-1, EU-2, AS-3, SA-4, AF-5, AU-6,
;               AR-7, TR-8, ST-9
;
;       note: for any 1 species, a number of sources can be looped over
;
; KEYWORD PARAMETERS:
;       exponential_dist - setting this keyword will distribute
;               emissions according to an inverse
;               exponential distribution, mimicking the
;               distribution of air in the

```

```

;           atmosphere. otherwise, emissions will be
;           distributed linearly throughout the
;           column. This should NOT be set with spec =
;           5 (air tracer) - it already has an
;           exponential distribution!!!
;
; custom_dist - setting this keyword will distribute emissions
;               over the column according to user defined
;               distributions. note - the number of elements of
;               custom_dist must equal the number of elements in
;               zr_grid. this keyword will over ride
;               exponential_dist and will prevent a linear
;               distribution as well. this should be an array
;               containing the decimal percentage of emissions
;               to be released in each level. custom_dist =
;               [.5,.5] will result in 50% of the emissions
;               being released in the 1st layer and 50% of the
;               emissions being released in the second layer
;
;
; levels - if this keyword is set, only the specified levels
;           will be used for folding. otherwise, all levels
;           supplied in retro_grid will be used. levels should be
;           a 2 elements array, [st_level, en_level], specifying
;           the number of the corresponding level (e.g., 0 =
;           lowest level).
;
; n_days - if this keyword is set, only the first n_days of the
;           retroplume will be used for folding. if not set, all
;           days provided will be used for folding.
;
; gridded - if this keyword is set, an nx by ny by n_days
;           array will be returned with contribution from each
;           grid cell for each time period. otherwise, an array
;           with n_days elements will be returned with the
;           total emission over the grid for each time period
;
; keep_levels - this keyword will return concentration in a
;               format [jd,z] or [jd,x,y,z] if 'gridded' is
;               set. This preserves the contribution from
;               each level
;
; refrm - this keyword will reform the return array and get rid
;         of any single element position (e.g., if the array is
;         [1,82] = [source,jd] the array will be reformed to
;         [82] = [jd]. this should only affect arrays with only
;         one source identified in the call.
;
; OUTPUTS:
; concentration - array of the mass mixing ratio from folding -
;                 1 element for each jd given in the retroplume
;                 (jdr_grid) and 1 element for each source region
;
; EXAMPLES:
;
;-

;convert the array from [x,y,z,t] to [t,x,y,z]
temp_new_array = make_array(n_elements(retro_grid(0, 0, 0, *)), $
                           n_elements(retro_grid(*, 0, 0, 0)), $
                           n_elements(retro_grid(0, *, 0, 0)), $
                           n_elements(retro_grid(0, 0, *, 0)))

```

```

FOR loopr = 0, n_elements(retro_grid(0, 0, 0, *)) - 1 DO BEGIN
    temp_new_array(loopr, *, *, *) = retro_grid(*, *, *, loopr)
enDFOR

;handle options
;*****
;*****
;*****
;*****
;*****

;remove levels from the retroplume we aren't interested in using
IF (keyword_Set(levels)) THEN BEGIN
    folding_grid = temp_new_array(*, *, *, levels(0):levels(1))
    fzf_grid = zr_grid(levels(0):levels(1))
ENDIF ELSE BEGIN
    folding_grid = temp_new_array
    fzf_grid = zr_grid
ENDELSE

;*****
;*****
;*****
;*****
;*****

;remove times from retroplume greater than n_days
IF (keyword_set(n_days)) THEN BEGIN
    iok = where(jdr_grid GE (max(jdr_grid) - n_days))
    jdr_grid = jdr_grid(iok)
    folding_grid = folding_grid(*, *, iok, *)
ENDIF

;*****
;*****
;*****
;*****
;*****

;calculate the difference in heights between levels, assume the lowest
;level starts from the surface (doesn't make sense to start from some
;higher altitude)
; fg_sz = size(folding_grid)

IF (n_elements(folding_grid(0, 0, 0, *)) NE n_elements(fzf_grid)) THEN BEG
IN
    print, 'Error - n levels in folding_grid do not match n levels in z_grid
;
;    return
ENDIF

;get delta h for each level
delta_z = make_array(n_elements(fzf_grid))
delta_z(0) = fzf_grid(0)
FOR loopz = 1, n_elements(fzf_grid) - 1 DO $
    delta_z(loopz) = fzf_grid(loopz) - fzf_grid(loopz - 1)
;*****

```

```

*****
;*****
*****
;*****
*****

  IF (NOT keyword_set(custom_dist)) THEN BEGIN
;calculate the weighting function for emissions for all levels
    IF (keyword_set(exponential_dist)) THEN BEGIN ;using exponential weight
      press_hts = 1025.0*exp(-fzr_grid/7400.)
      delta_p = make_array(n_elements(fzr_grid))
      delta_p(0) = 1025.0-press_hts(0)
      FOR loopp = 1, n_elements(fzr_grid)-1 DO $
        delta_p(loopp) = press_hts(loopp-1)-press_hts(loopp)
      diff_p = 1025.0-min(press_hts)
      emiss_dist = delta_p/diff_p
    ENDIF ELSE BEGIN ;use linear distribution
      emiss_dist = delta_z(*)/max(fzr_grid)
    ENDELSE
  enDIF ELSE BEGIN
    IF ((n_elements(custom_dist) EQ n_elements(fzr_grid)) AND $
      ((total(custom_dist) GE 0.99999) AND $
      (total(custom_dist) LE 1.00001))) $
      THEN BEGIN
        emiss_dist = custom_dist
      enDIF ELSE BEGIN
        print, 'Error - either nmuber of elements in custom_dist '+$
          'do not equal the number of z levels'
        print, 'Error - or total of custom_dis does not equal 1 (100%).'
        print, 'Using linear distribution instead'
        emiss_dist = delta_z(*)/max(fzr_grid)
      ENDELSE
    ENDELSE

;*****
*****
;*****
*****
;*****
*****

;loading CO or NOx w/, contstant emissions
  IF ((species(0) EQ 1) OR (species(0) EQ 2)) THEN BEGIN
    CASE species(0) OF
      1: source_file = '~rcowen/rcowen_research/flex_dirs/plotting_pros/emissio
ns/tot_non_bb_emiss.idlsav'
      2: source_file = '~rcowen/rcowen_research/flex_dirs/plotting_pros/emissio
ns/tot_nox_non_bb_emiss.idlsav'
    enDCASE
    CASE species(0) OF
      1: conversion_factor = 28.95/28.01 ; convert mass/mass to vol/vol for
CO
      2: conversion_factor = 28.95/46.05 ; convert mass/mass to vol/vol for
NOx (as NO2)
    enDCASE
;restore the emissions file and the area of grid cells
    restore, source_file
    restore, '~rcowen/rcowen_research/flex_dirs/pros/nh_area_grid.idlsav'

    FOR loop_source = 0, n_elements(source)-1 DO begin

```

```

FOR loop_source = 0, n_elements(source)-1 DO begin

;select the source region we want emissions for
  source_index = reform(edgar_emissions(*, *, 1))
  x_arr = reform(edgar_emissions(*, *, 2))
  y_arr = reform(edgar_emissions(*, *, 3))
  these_emissions = where(source_index NE source(loop_source))
  emissions = double(reform(edgar_emissions(*, *, 0)))
  IF ((these_emissions(0) NE (-1)) AND (source(loop_source) NE 7)) $
  THEN emissions(these_emissions) = 0

;select only the part of the emissions grid that we have saved
;retroplume output for
  ok_x = where((x_arr(*, 0) LE xr_grid(0)) AND (x_arr(*, 0) GE xr_grid(1)))
  ok_y = where((y_arr(0, *) LE yr_grid(0)) AND (y_arr(0, *) GE yr_grid(1)))

  use_emissions = make_array(n_elements(ok_x), n_elements(ok_y))
  use_emissions = emissions(min(ok_x):max(ok_x), $
                           min(ok_y):max(ok_y))

;convert emissions from kg/yr to kg/s/m^2 for multiplication with res
;times (units of s*m^3/kg)
  use_emissions = (use_emissions(*, *))*(1/(365.*24.*60.*60.))
  emissions_per_area = use_emissions/area_grid

;transform emissions into emissions per level-grid box
  emissions_per_level = make_array(n_elements(ok_x), n_elements(ok_y), $
                                   n_elements(fzr_grid))
  FOR loopel = 0, n_elements(fzr_grid)-1 DO $
    emissions_per_level(*, *, loopel) = emissions_per_area(*, *)*$
    emiss_dist(loopel)/delta_z(loopel)

;loop over each day, folding emissions
  concentration = folding_grid(*, *, *, *)*0.0
  FOR loopd = 0, n_elements(jdr_grid)-1 DO $
    concentration(loopd, *, *, *) = folding_grid(loopd, *, *, *)*$
    emissions_per_level(*, *, *)*10.^9.

;get total for each day unless gridded keyword is set
  IF (NOT keyword_set(gridded)) THEN BEGIN
    concentration = total(concentration, 2)
    concentration = total(concentration, 2)
    IF (NOT keyword_Set(keep_levels)) THEN begin
      IF (n_elements(fzr_grid) GT 1) THEN concentration = total(concentr
ation, 2)
    ENDIF
  ENDIF ELSE BEGIN
    IF (NOT keyword_Set(keep_levels)) THEN begin
      IF (n_elements(fzr_grid) GT 1) THEN concentration = total(concentr
ation, 4)
    ENDIF
  endELSE

  IF (loop_source EQ 0) THEN BEGIN
    arr_size = size(concentration)
    make_sz = make_array(arr_size(0)+1)
    make_sz(0) = 1
    make_sz(1:arr_size(0)) = arr_size(1:arr_size(0))

```

```

save_conc = make_Array(make_sz)
save_conc(*) = concentration

ENDIF ELSE BEGIN

    arr_size = size(concentration)
    make_sz = make_array(arr_size(0)+1)
    make_sz(0) = 1
    make_sz(1:arr_size(0)) = arr_size(1:arr_size(0))
    tmp_save_conc = make_Array(make_sz)
    tmp_save_conc(*) = concentration

    save_conc = [save_conc, tmp_save_conc]

ENDELSE

ENDFOR;end loop over sources

concentration = save_conc

ENDIF                                     ;end using anthro CO or NOx

;*****
;*****
;*****
;*****
;*****

;loading biomass CO, emissions vary by day
IF ((species(0) EQ 3) OR (species(0) EQ 4)) THEN BEGIN

    conversion_factor = 28.95/28.01 ; convert mass/mass to vol/vol for CO

    source_file = '/home/darcy/rcowen/rco_research/flex_dirs/plotting_pros/emissions/'

    caldat, jdr_grid(0), mo, da, yr
    yr = strtrim(string(yr), 2)

;set source region for selecting emissions from emissions
;file. source(0) = 3 means NA emissions, want emi_source to = 1 for NA
;emissions (according to the standard for anth CO and NOx). source(0)
;= 4 means Siberian emissions, want emi_source to = 3 for asian
;emissions.
CASE species(0) OF
    3: emi_source = 1 ;north america
    4: emi_source = 3 ;asia
enDCASE

FOR loop_source = 0, n_elements(source)-1 DO begin

CASE source(loop_source) OF
    1: source_file = source_file+'co_ag_f_all_'
    2: source_file = source_file+'co_ag_s_all_'
    3: source_file = source_file+'co_bg_f_all_'
    4: source_file = source_file+'co_bg_s_all_'
    5: source_file = source_file+'co_all_'
enDCASE

```

```

source_file = source_file+yr+'_invent.idlsav'

;test to make sure file exists, otherwise, give an error and return to
;main program calling level
file_test = file_search(source_file)

;if file exists, continue, else give an error and return
IF (file_test NE '') THEN BEGIN

;restore the emissions file and the area of grid cells
restore, source_file
restore, '~rcowen/rco_research/flex_dirs/pros/nh_area_grid.idlsav'

;select the source region we want emissions for
source_index = reform(bb_invent(*, *, 1, *))
x_arr = reform(bb_invent(*, *, 2, 0))
y_arr = reform(bb_invent(*, *, 3, 0))
these_emissions = where(source_index NE emi_source)
emissions = double(reform(bb_invent(*, *, 0, *)))
IF (these_emissions(0) NE (-1)) THEN emissions(these_emissions) = 0

;select only the part of the emissions grid that we have saved
;retroplume output for
ok_x = where((x_arr(*, 0) LE xr_grid(0)) AND (x_arr(*, 0) GE xr_grid(
1)))
ok_y = where((y_arr(0, *) LE yr_grid(0)) AND (y_arr(0, *) GE yr_grid(
1)))

use_emissions = make_array(n_elements(ok_x), n_elements(ok_y), n_elem
ents(jd_Grid))
use_emissions = emissions(min(ok_x):max(ok_x), $
min(ok_y):max(ok_y), *)

;convert emissions from kg/day to kg/s/m^2 for multiplication with res
;times (units of s*m^3/kg)
use_emissions = (use_emissions(*, *, *))*(1/(24.*60.*60.))
jd_area_grid = make_array(n_elements(ok_x), n_elements(ok_y), n_eleme
nts(jd_Grid))
FOR loopjd = 0, n_elements(jd_Grid)-1 DO jd_area_grid(*, *, loopjd) =
area_grid
emissions_per_area = use_emissions/jd_area_grid

;transform emissions into emissions per level-grid box
emissions_per_level = make_array(n_elements(ok_x), n_elements(ok_y),
$
n_elements(jd_grid), n_elements(fzr_
grid))
FOR loopel = 0, n_elements(fzr_grid)-1 DO $
emissions_per_level(*, *, *, loopel) = emissions_per_area(*, *, *)*
$
emiss_dist(loopel)/delta_z(loopel)

;loop over each day, folding emissions
concentration = folding_grid(*, *, *, *)*0.0
FOR loopd = 0, n_elements(jdr_grid)-1 DO BEGIN

caldat, jdr_grid(loopd), this_mo, this_da, this_yr
this_jd = julday(this_mo, this_da, this_yr, 12, 00)
use_jd = where(jd_grid EQ this_jd)
IF (use_jd(0) NE (-1)) THEN $

```

```

        concentration(loopd, *, *, *) = folding_grid(loopd, *, *, *)*$
        emissions_per_level(*, *, use_jd, *)*10.^9.
    endFOR

;get total for each day unless gridded keyword is set
    IF (NOT keyword_set(gridded)) THEN BEGIN
        concentration = total(concentration, 2)
        concentration = total(concentration, 2)
        IF (NOT keyword_Set(keep_levels)) THEN begin
            IF (n_elements(fzr_grid) GT 1) THEN concentration = total(concent
ration, 2)
            ENDIF
        endIF ELSE BEGIN
            IF (NOT keyword_Set(keep_levels)) THEN begin
                IF (n_elements(fzr_grid) GT 1) THEN concentration = total(concent
ration, 4)
                ENDIF
            endELSE
        endIF ELSE BEGIN
            ;no emissions file available for specified ye
ar

            print, 'Error - no BB file for year specified'
            return

        ENDELSE

        IF (loop_source EQ 0) THEN BEGIN

            arr_size = size(concentration)
            make_sz = make_array(arr_size(0)+1)
            make_sz(0) = 1
            make_sz(1:arr_size(0)) = arr_size(1:arr_size(0))
            save_conc = make_Array(make_sz)
            save_conc(*) = concentration

        ENDIF ELSE BEGIN

            arr_size = size(concentration)
            make_sz = make_array(arr_size(0)+1)
            make_sz(0) = 1
            make_sz(1:arr_size(0)) = arr_size(1:arr_size(0))
            tmp_save_conc = make_Array(make_sz)
            tmp_save_conc(*) = concentration

            save_conc = [save_conc, tmp_save_conc]

        ENDELSE

    ENDFOR
    ;end loop over sources

    concentration = save_conc

ENDIF
;end using biomass burning from NA or SI

;*****
;*****
;*****

;loading air tracer w/, constant 'emissions'
    IF (species(0) EQ 5) THEN BEGIN

```



```

    source_file = '~rcowen/rco_research/flex_dirs/plotting_pros/emissions/air_tracer_gridded.idlsav'
;restore the emissions file and the area of grid cells
    restore, source_file
    restore, '~rcowen/rco_research/flex_dirs/pros/nh_area_grid.idlsav'

    conversion_factor = 1.

    FOR loop_source = 0, n_elements(source)-1 DO BEGIN

;select the source region we want emissions for
        source_index = reform(air_tracer(*, *, *, 4))
        x_arr = reform(air_tracer(*, *, *, 0))
        y_arr = reform(air_tracer(*, *, *, 1))
        z_arr = reform(air_tracer(*, *, *, 2))
        these_emissions = where(source_index NE source(loop_source))
        emissions = double(reform(air_tracer(*, *, *, 3)))

        IF (these_emissions(0) NE (-1)) THEN BEGIN
            emissions(these_emissions) = 0
        ENDIF ELSE BEGIN
            print, 'Error - source selected for air tracer folding not valid'
            return
        ENDELSE

;select only the part of the emissions grid that we have saved
;retroplume output for
        ok_x = where((x_arr(*, 0, 0) LE xr_grid(0)) AND (x_arr(*, 0, 0) GE xr_grid(1)))
        ok_y = where((y_arr(0, *, 0) LE yr_grid(0)) AND (y_arr(0, *, 0) GE yr_grid(1)))
        ok_z = where((z_arr(0, 0, *) LE max(fzr_grid)) AND (z_arr(0, 0, *) GE min(fzr_grid)))

        use_emissions = make_array(n_elements(ok_x), n_elements(ok_y), $
                                   n_elements(ok_z))
        use_emissions = emissions(min(ok_x):max(ok_x), $
                                   min(ok_y):max(ok_y), $
                                   min(ok_z):max(ok_z))

;convert emissions from kg/m^3 to kg/s/m^2 for multiplication with res
;times (units of s*m^3/kg)
        use_emissions = (use_emissions(*, *, *))*(1/(3.*60.*60.))

;loop over each day, folding emissions
        concentration = folding_grid(*, *, *, *)*0.0
        FOR loopd = 0, n_elements(jdr_grid)-1 DO $
            concentration(loopd, *, *, *) = folding_grid(loopd, *, *, *)*$
            use_emissions(*, *, *) ;*10.^9.

;get total for each day unless gridded keyword is set
        IF (NOT keyword_set(gridded)) THEN BEGIN
            concentration = total(concentration, 2)
            concentration = total(concentration, 2)
            IF (NOT keyword_set(keep_levels)) THEN BEGIN
                IF (n_elements(fzr_grid) GT 1) THEN concentration = total(concentration, 2)
            ENDIF
        ENDIF ELSE BEGIN

```

```

        IF (NOT keyword_Set(keep_levels)) THEN begin
            IF (n_elements(fzr_grid) GT 1) THEN concentration = total(concentr
ation, 4)
            ENENDIF
        ENDELSE

;convert from mass/mass to vol/vol using specific conversion factors
        concentration = concentration*conversion_factor

        IF (loop_source EQ 0) THEN BEGIN

            arr_size = size(concentration)
            make_sz = make_array(arr_size(0)+1)
            make_sz(0) = 1
            make_sz(1:arr_size(0)) = arr_size(1:arr_size(0))
            save_conc = make_Array(make_sz)
            save_conc(*) = concentration

        ENENDIF ELSE BEGIN

            arr_size = size(concentration)
            make_sz = make_array(arr_size(0)+1)
            make_sz(0) = 1
            make_sz(1:arr_size(0)) = arr_size(1:arr_size(0))
            tmp_save_conc = make_Array(make_sz)
            tmp_save_conc(*) = concentration

            save_conc = [save_conc, tmp_save_conc]

        ENDELSE

    ENDFOR                                ;end loop over sources

    concentration = save_conc

    ENENDIF                                ;end using anthro CO or NOx

    IF (keyword_set(refrm)) THEN concentration = reform(concentration)

    return
end

```

B.2.5 `load_flex_gridded_data.pro`



03/19/09

load_flex_gridded_data.pr

```
PRO load_flex_gridded_data, st_time, en_time, spec, source, $
    gridded_conc, jdg_grid, xg_grid, $
    yg_grid, zg_grid, $
    truncate = truncate, $
    convert_conc = convert_conc

;+
;10/02/2006 by RCO
; NAME:
; load_flex_gridded_data
;
; PURPOSE:
; Loads gridded flexpart data from forward simulations.
;
;
; CATEGORY:
; Data loading
;
; CALLING SEQUENCE:
; load_flex_gridded_data, st_time, en_time, spec, source, $
;                                gridded_conc, jdg_grid, xg_grid, $
;                                yg_grid, zg_grid, truncate=truncate, $
;                                convert_conc = convert_conc
; INPUTS:
; st_time - start time for gridded data in juldays
; en_time - end time for gridded data in juldays
; spec - species to be loaded: CO = 1, NO2 T = 2, NO2 W = 3
; source - source for species to be loaded: NAA = 1, NAB/AGF =
;         2, NAB/BGS = 3, NAB/TOT = 4
; KEYWORD PARAMETERS:
; truncate - if this keyword is set, the gridded_conc and
;            jdg_grid will be truncated to match the start and
;            end times specified in the calling sequence,
;            otherwise, whole months will be returned.
;
; convert_conc - if this keyword is set, the concentration
;                (ng/m^3) will be returned. otherwise, the
;                column density will be returned (ng/m^2)
; OUTPUTS:
; gridded_conc - array containing the gridded
;                concentrations. format of [jd, x, y, z]
; jdg_grid - array of julian day time corresponding to each
;            gridded concentration
; xg_grid, yg_grid - array with the max, min, number of points
;                  and delta between points for the x & y
;                  grid of the gridded concentrations
; zg_grid - array of the top of each level for gridded
;           concentrations
; EXAMPLES:
;
;-

;setup data directories based off selections of sources/species
CASE source OF
  1: source_dir = 'NAA'
  2: source_dir = 'NAB'
  3: source_dir = 'NAB'
  4: source_dir = 'NAB'
ENDCASE
CASE source OF
  1: subdir = ''
  2: subdir = '/7500/AGF'
  3: subdir = '/7500/BGS'
  4: subdir = '/7500/TOT'
```

```

ENDCASE
CASE spec OF
  1: type_dir = 'CO'
  2: type_dir = 'NO2T'
  3: type_dir = 'NO2W'
ENDCASE

data_dir = '/local/reh/rcowen/flex_forward_running_output/'$
+source_dir+'-'+type_dir+subdir+'/gridded_conc/'
file_ext = '_gridded_conc.idlsav'

;get start and end dates of files to load
caldat, st_time, smon, sday, syr
caldat, en_time, emon, eday, eyr

this_jd = julday(smon, 1, syr)
last_jd = julday(emon, 1, eyr)

;loop over months and load gridded concentrations
WHILE (this_jd LE last_jd) DO begin

  caldat, this_jd, tmon, tday, tyr
  IF (tmon LE 9) THEN tmon = '0'+strtrim(string(tmon), 2) ELSE $
    tmon = strtrim(string(tmon), 2)
  this_file = strtrim(string(tyr), 2)+tmon
  fname = data_dir+this_file+file_ext
  test_file = file_search(fname)
  IF (test_file NE '') THEN BEGIN
    restore, fname
  ENDIF ELSE BEGIN
    print, 'WARNING - no data for ', tyr, ' ', tmon
    print, fname
  ENDELSE

  IF (keyword_set(temp_grid_conc)) THEN BEGIN
    temp_grid_conc = [temp_grid_conc, gridded_conc]
    temp_jd = [temp_jd, jd_grid]
  ENDIF ELSE BEGIN
    temp_grid_conc = gridded_conc
    temp_jd = jd_grid
  ENDELSE

  this_jd = julday(tmon+1, tday, tyr)

ENDWHILE

;if no data has been loaded, return to main program
IF (NOT keyword_set(temp_grid_conc)) THEN return

;if truncate is set, then only keep the range of jd's requested,
;otherwise keep all of the month's requested

IF (keyword_set(truncate)) THEN BEGIN
  iok = where((temp_jd GE st_time) AND (temp_jd LE en_time))
  junk = temporary(gridded_conc)
  junk = temporary(jd_grid)
  gridded_conc = temp_grid_conc(iok, *, *, *)
  jd_grid = temp_jd(iok)
ENDIF ELSE BEGIN
  junk = temporary(gridded_conc)
  junk = temporary(jd_grid)
  gridded_conc = temp_grid_conc

```

```
    jdg_grid = temp_jd
  ENDELSE

  IF (keyword_set(convert_conc)) THEN BEGIN
    FOR loopz = 0, n_elements(z_grid)-1 DO BEGIN
      IF (loopz EQ 0) THEN delta_lev = float(z_grid(loopz)) ELSE $
        delta_lev = float(z_grid(loopz)-z_grid(loopz-1))
    ;    print, delta_lev, z_grid(loopz)
      gridded_conc(*, *, *, loopz) = gridded_conc(*, *, *, loopz)/delta_lev
    endfor
  enDIF

  xg_grid = x_grid
  yg_grid = y_grid
  zg_grid = z_grid

  return
end
```

B.2.6 `load_flex_pico_ppb_data.pro`



```

PRO load_flex_pico_ppb_data, st_time, en_time, spec, source, $
    pico_conc, jdp_grid, xp_grid, $
    yp_grid, ap_grid, $
    all = all, lat = lat, $
    lon = lon, height = height, $
    custom_dir = custom_dir

;+
;10/02/2006 by RCO
; NAME:
; load_flex_pico_ppb_data
;
; PURPOSE:
; Loads ppb near pico flexpart data from forward simulations.
;
; CATEGORY:
; Data loading
;
; CALLING SEQUENCE:
; load_flex_pico_ppb_data, st_time, en_time, spec, source, $
;                                pico_conc, jdp_grid, xp_grid, $
;                                yp_grid, ap_grid, all = all, $
;                                lat = lat, lon = lon, height = height
;
; INPUTS:
; st_time - start time for gridded data in juldays
; en_time - end time for gridded data in juldays
; spec - species to be loaded: CO = 1, NO2 T = 2, NO2 W = 3,
;       O3 = 4
; source - source for species to be loaded: NAA = 1, NAB/AGF =
;       2, NAB/BGS = 3, NAB/TOT = 4, STR = 5
; KEYWORD PARAMETERS:
; all - if this keyword is set, then all available ppb near pico
;       values are returned
;
; lat, lon, height - if any of these keywords are set, the
;                    concentration of the specified cell will be
;                    returned, otherwise the cell with pico is
;                    assumed. 1 2, or 3 of these keywords may be
;                    set, any not set will assume pico
;                    coordinates.
;
; OUTPUTS:
; pico_con - array containing the gridded
;            concentrations. format of [jd, ac] unless 'all'
;            keyword is set, then format is [jd, x, y, z, ac]
; jdg_grid - array of julian day time corresponding to each
;            gridded concentration
; xg_grid, yg_grid - array with the lower, left coordinate of
;                   the x & y location of the
;                   grid of the gridded concentrations
; ag_grid - array of the top of each level for age classes
;            concentrations
; EXAMPLES:
;
;-

IF (keyword_set(custom_dir)) THEN BEGIN

    data_dir = custom_dir+'/pico_ppb/'
    file_ext = '_near_pico_ppb.idlsav'

```



```

ENDIF ELSE BEGIN

;setup data directories based off selections of sources/species
CASE source OF
  1: source_dir = 'NAA'
  2: source_dir = 'NAB'
  3: source_dir = 'NAB'
  4: source_dir = 'NAB'
  5: source_dir = 'STR'
ENDCASE
CASE source OF
  1: subdir = ''
  2: subdir = '/7500/AGF'
  3: subdir = '/7500/BGS'
  4: subdir = '/7500/TOT'
  5: subdir = ''
ENDCASE
CASE spec OF
  1: type_dir = 'CO'
  2: type_dir = 'NO2T'
  3: type_dir = 'NO2W'
  4: type_dir = 'O3'
ENDCASE

data_dir = '/local/reh/rcowen/flex_forward_running_output/'$
+source_dir+'-'+type_dir+subdir+'/pico_ppb/'
file_ext = '_near_pico_ppb.idlsav'

ENDELSE

;get start and end dates of files to load
caldat, st_time, smon, sday, syr
caldat, en_time, emon, eday, eyr

this_jd = julday(smon, 1, syr)
last_jd = julday(emon, 1, eyr)

;loop over months and load gridded concentrations
WHILE (this_jd LE last_jd) DO begin

  caldat, this_jd, tmon, tday, tyr
  IF (tmon LE 9) THEN tmon = '0'+strtrim(string(tmon), 2) ELSE $
    tmon = strtrim(string(tmon), 2)
  this_file = strtrim(string(tyr), 2)+tmon
  fname = data_dir+this_file+file_ext
  test_file = file_search(fname)
  IF (test_file NE '') THEN BEGIN
    restore, fname
  ENDIF ELSE BEGIN
    print, 'WARNING - no data for ', tyr, ' ', tmon
    print, fname
  ENDELSE

  IF (keyword_set(temp_ages)) THEN BEGIN
    IF (n_elements(PICO_PPB_AGES(0, 0, 0, 0, *)) EQ $
      n_elements(temp_ages(0, 0, 0, 0, *))) THEN begin
      temp_ages = [temp_ages, pico_ppb_ages]
      temp_jd = [temp_jd, jd_grid]
    ENDIF ELSE BEGIN
;need to match age classes
      FOR loopa = 0, n_elements(age_grid)-1 DO BEGIN
        iok = where(temp_age_grid EQ age_grid(loopa))

```

```

        IF (iok(0) NE (-1)) THEN BEGIN
            IF (keyword_Set(start_ages)) THEN $
                sub_ages = [sub_ages, iok] ELSE BEGIN
                    sub_ages = iok
                    start_ages = 1
                ENDELSE
            ENDIF
        ENDFOR

        FOR loopa = 0, n_elements(temp_age_grid)-1 DO BEGIN
            iok = where(age_grid EQ temp_age_grid(loopa))
            IF (iok(0) NE (-1)) THEN BEGIN
                IF (keyword_Set(start_ages2)) THEN $
                    sub_ages2 = [sub_ages2, iok] ELSE BEGIN
                        sub_ages2 = iok
                        start_ages2 = 1
                    ENDELSE
                ENDIF
            ENDFOR
            temp_ages = [temp_ages(*, *, *, *, sub_ages), $
                        pico_ppb_ages(*, *, *, *, sub_ages2)]
            temp_jd = [temp_jd, jd_grid]
            temp_age_grid = age_grid(sub_ages2)
        ENDELSE
    ENDIF ELSE BEGIN
        temp_ages = pico_ppb_ages
        temp_jd = jd_grid
        temp_age_grid = age_grid
    ENDELSE
    this_jd = julday(tmon+1, tday, tyr)
ENDWHILE

;if no data has been loaded, return to main program
IF (NOT keyword_set(temp_ages)) THEN return

;handle options
IF (NOT keyword_set(all)) THEN BEGIN
    IF (keyword_set(lat)) THEN BEGIN
        y_loc = (where(y_grid EQ (lat)))
        IF (y_loc(0) EQ (-1)) THEN y_loc = (where(y_grid EQ (38)))
    ENDIF ELSE y_loc = (where(y_grid EQ (38)))

    IF (keyword_set(lon)) THEN BEGIN
        x_loc = (where(x_grid EQ (lon)))
        IF (x_loc(0) EQ (-1)) THEN x_loc = (where(x_grid EQ (-29)))
    ENDIF ELSE x_loc = (where(x_grid EQ (-29)))

    IF (keyword_set(height)) THEN BEGIN
        z_loc = (where(z_grid EQ (height)))
        IF (x_loc(0) EQ (-1)) THEN z_loc = (where(z_grid EQ (2500)))
    ENDIF ELSE z_loc = (where(z_grid EQ (2500)))

    xp_grid = x_grid(x_loc)
    yp_grid = y_grid(y_loc)
    zp_grid = z_grid(z_loc)

    pico_conc = reform(temp_ages(*, x_loc, y_loc, z_loc, *))
    jdp_grid = temp_jd
    ap_grid = age_grid
ENDIF ELSE BEGIN

```

03/19/09

load_flex_pico_ppb_data.pr

```
pico_conc = temp_ages

xp_grid = x_grid
yp_grid = y_grid
zp_grid = z_grid
jdp_grid = temp_jd
ap_grid = age_grid

enDELSE

return
END
```

B.2.7 `load_flex_retro_data.pro`



03/19/09

load_flex_retro_data.pr

```
PRO load_flex_retro_data, get_time, $
    rp_grid, jdr_grid, xr_grid, yr_grid, zr_grid, $
    levels = levels, time_first = time_first, $
    use_orig_fmt = use_orig_fmt, $
    wet_remov = wet_remov, $
    custom_dir = custom_dir, $
    verbose = verbose

;+
;10/02/2006 by RCO
; NAME:
; load_flex_retro_data
;
; PURPOSE:
; Loads gridded flexpart data from backward simulations. This
; will currently only load 1 time period and return an rp_grid
; with multiple levels (as specified)
;
;
; CATEGORY:
; Data loading
;
; CALLING SEQUENCE:
; load_flex_gridded_data, get_time, $
;                                     rp_grid, jdr_grid, xr_grid, yg_grid, zg_grid
; $
;                                     levels = levels, time_first = time_first, $
;                                     use_orig_fmt = use_orig_fmt, $
;                                     wet_remov = wet_remov, $
;                                     custom_dir = custom_dir, $
;                                     verbose = verbose
; INPUTS:
; get_time - requested load time in julian days
; KEYWORD PARAMETERS:
;     levels - if set, only the specified levels will be returned,
;              otherwise all levels will be returned, excluding the
;              summed column
;
;     time_first - if set, the returned format of rp_grid will be
;                  [jd, x, y, z], otherwise it will be [x, y, z, jd]
;
;     wet_remov - if set, the retroplume with wet removal will be
;                  loaded, otherwise the retroplumes without removal
;                  will be loaded
;
;     use_orig_fmt - the original program returned the retroplume
;                    grid in the form [x,y,jd,z]. this has been
;                    changed in the most recent version to
;                    [x,y,z,jd]. this option has been created for
;                    compatibility purposes and can be set to get
;                    the original format.
;
;     custom_dir - if this option is set, the program will attempt
;                  to load retroplumes in the specified
;                  directory. please note that the
;                  retro_file_info.idlsav file must be present in
;                  the requested directory. this file can be created
;                  by calling the program make_retro_filelist, which
;                  also has a custom_dir option.
;
;     verbose - print the name of each file being loaded.
;
; OUTPUTS:
; rp_grid - array containing the gridded
```

```

;               retroplume values. format of [x, y, jd, z]
;               unless time_first is set, then the format will
;               be [jd, x, y, z]
;       jdr_grid - array of julian day time corresponding to each
;               gridded concentration. format of
;       xr_grid, yr_grid - array with the max, min, number of points
;               and delta between points for the x & y
;               grid of the gridded concentrations
;       zr_grid - array of the top of each level for gridded
;               concentrations
; EXAMPLES:
;
;
;-

;load retro file info
  IF (keyword_set(custom_dir)) THEN $
    restore, custom_dir+'retro_file_info.idlsav' ELSE BEGIN
      IF (keyword_set(wet_remov)) THEN type = 'pico_std_wet/' $
      ELSE type = 'pico_std_dry/'

      restore, '/local/reh/rcowen/retro_flex_idl_output/'+$
        type+'retro_file_info.idlsav'
    ENDELSE

;check to make sure time exists
  min_u_jd = min(abs(uniq_jd-get_time(0)))
  IF (min_u_jd GT 1e-4) THEN BEGIN
    print, 'error - no data for this period'
    return
  ENDIF

  IF keyword_set(rp_grid) THEN junk = temporary(rp_grid)

;pick which levels we are going to load - all but total column unless
;                               levels keyword is set

  IF (keyword_set(levels)) THEN BEGIN
    use_levels = levels
    n_levs = n_elements(levels)
  ENDIF ELSE BEGIN
    use_levels = where(retro_file_info(0, *) GT 0, n_levs)
    use_levels = reform(retro_file_info(0, use_levels(0):use_levels(n_levs-1)
  ))
  ENDELSE

;loop over all levels, load the data and append it to the output array
  these_files = where(abs(all_jd - get_time(0)) LT 1e-4)
  FOR loop1 = 0, n_levs-1 DO BEGIN
    this_file = where(all_levels(these_files) EQ use_levels(loop1))
    load_file = fnames(these_files(this_file))
    restore, load_file
    IF (keyword_Set(verbose)) THEN print, load_file

    IF (keyword_set(rp_grid)) THEN BEGIN
      rp_grid(*, *, *, loop1) = res_times(*, *, *)
    ENDIF ELSE BEGIN

      rp_grid = make_array(n_elements(res_times(*, 0, 0)), $
        n_elements(res_times(0, *, 0)), $

```

```

                                n_elements(res_times(0, 0, *)), $
                                n_levs)
    rp_grid(*, *, *, loopl) = res_times(*, *, *)

    enDELSE

    zr_grid = use_levels

endfor

xr_grid = x_grid
yr_grid = y_grid
jdr_grid = jd_grid

IF (NOT keyword_set(use_orig_fmt)) THEN BEGIN
  IF (keyword_set(time_first)) THEN BEGIN
    temp_new_array = make_array(n_elements(rp_grid(0, 0, *, 0)), $
                                n_elements(rp_grid(*, 0, 0, 0)), $
                                n_elements(rp_grid(0, *, 0, 0)), $
                                n_elements(rp_grid(0, 0, 0, *)))

    FOR loopr = 0, n_elements(rp_grid(0, 0, *, 0))-1 DO BEGIN
      temp_new_array(loopr, *, *, *) = rp_grid(*, *, loopr, *)
    enDFOR

    rp_grid = temp_new_array

  enDIF ELSE BEGIN

    temp_new_array = make_array(n_elements(rp_grid(*, 0, 0, 0)), $
                                n_elements(rp_grid(0, *, 0, 0)), $
                                n_elements(rp_grid(0, 0, 0, *)), $
                                n_elements(rp_grid(0, 0, *, 0)))

    FOR loopr = 0, n_elements(rp_grid(0, 0, *, 0))-1 DO BEGIN
      temp_new_array(*, *, *, loopr) = rp_grid(*, *, loopr, *)
    enDFOR

    rp_grid = temp_new_array

  ENDELSE
enDIF

  return
end

```

B.3 Programs for running and processing forward FLEXPART simulations

This section contains the programs created to run forward FLEXPART simulations and process the ASCII output into idl files.

B.3.1 `convert_forward_temps_to_gridded_final.pro`



03/19/09

convert_forward_temps_to_gridded_final

```
PRO convert_forward_temps_to_gridded_final, sourcedir, go_pptv, $
                                     go_conc, go_flux, savedir

;+
;NAME: convert_forward_temps_to_gridded_final
;
;PURPOSE: This program reads the intermediate FLEXPART files and saves
;the final, month-long files to the appropriate directory.
;
;CATEGORY:
;
;FLEXPART program control
;
;CALLING SEQUENCE:
;convert_forward_temps_to_gridded_final, sourcedir, go_pptv, $
;                                     go_conc, go_flux, savedir
;
;INPUTS:
;
; sourcedir - the directory containing the intermediate FLEXPART idlsav
;            files.
;
; go_pptv - if set to 1, the mixing ratio files will be processed.
;
; go_conc - if set to 1, the concentration files will be processed.
;
; go_flux - if set to 1, the flux files will be processed.
;
; savedir - directory that the final output files will be saved to. The
;          savedir must have the following sub-directories:
;          flux_special, flux_standard, gridded_conc, pico_ppb.
;
;KEYWORD PARAMETERS:
;
;none
;
; OUTPUTS:
;
;
;
; EXAMPLES:
;
;-
  restore, sourcedir+'header.idlsav'

  IF (go_pptv EQ 1) THEN BEGIN
;process the near pico ppb files

;find the temp idlsav files
  idlsaves = file_Search(sourcedir+'*_temp_ppb_near_pico_grid.idlsav', $
                        count = n_idlsaves)

;get the names of just the files (no directories)
  idlsave_names = file_basename(idlsaves)

;get list of dates
  yrs = STRMID(idlsave_names, 0, 4)
  mos = STRMID(idlsave_names, 4, 2)
  das = STRMID(idlsave_names, 6, 2)
  hrs = STRMID(idlsave_names, 8, 2)
;make list of uniq yrs, months for processing
  yrs_1 = yrs[UNIQ(yrs, SORT(yrs))]
  mos_1 = mos[UNIQ(mos, SORT(mos))]
```

03/19/09

convert_forward_temps_to_gridded_final

```
;loop over each year, month to make permanent save files (save files
;will be 1 month long)
  FOR loopy = 0, n_elements(yrs_l)-1 DO BEGIN
    FOR loopm = 0, n_elements(mos_l) -1 DO BEGIN

      this_mo = where((yrs EQ yrs_l(loopy)) AND (mos EQ mos_l(loopm)))
      IF (this_mo(0) NE (-1)) THEN begin
        jds = make_array(n_elements(this_mo))
        jd_grid = julday(mos(this_mo), das(this_mo), yrs(this_mo), hrs(thi
s_mo))
        pico_ppb_ages = make_array(n_elements(this_mo), 7, 5, 11, n elemen
ts(age_grid))

;loop over all temp files for this time period and add them to the
;final array
        FOR loopf = 0, n_elements(this_mo)-1 DO BEGIN
          restore, idlsaves(this_mo(loopf))
          pico_ppb_ages(loopf, *, *, *, *) = pico_xyz_ages
        ENDFOR

;check to see if a file exists for this period to add to it (other
;wise it would be replaced)
        fname = yrs_l(loopy)+mos_l(loopm)+'_near_pico_ppb.idlsav'

        x_grid = grid_info(0:6)
        y_grid = grid_info(7:11)
        z_grid = grid_info(12:22)

        check_owrite = file_Search(savedir+'pico_ppb/'+fname)

        IF (check_owrite NE '') THEN BEGIN
          print, 'do you wish to overwrite the file '
          print, savedir+'pico_ppb/'+fname
          answer = ''
          read, answer, PROMPT = 'y or n '
          IF ((answer EQ 'y') OR (answer EQ 'Y')) THEN BEGIN
            save, pico_ppb_ages, x_grid, y_grid, z_grid, age_grid, jd_grid
, $
              filename = savedir+'pico_ppb/'+fname, /compress
            ENDIF
          ENDIF ELSE BEGIN
            save, pico_ppb_ages, x_grid, y_grid, z_grid, age_grid, jd_grid, $
              filename = savedir+'pico_ppb/'+fname, /compress
          ENDELSE
        ENDIF

      ENDFOR
    ENDFOR

    cp_command = 'cp /local/reh/rcowen/flex_forward_running_output/NAA-CO/pi
co_ppb/@README '+$
      savedir+'pico_ppb/'
    spawn, cp_command

  ENDFOR

  IF (go_conc EQ 1) THEN BEGIN
;process the gridded conc files
    level_list = make_array(n_elements(z_grid), /string)
```

```

level_delta = make_array(n_elements(z_grid))

FOR loop11 = 0, (n_elements(z_grid)-1) DO BEGIN
    lev_name = strtrim(string(fix(z_grid(loop11))), 2)

    IF (z_grid(loop11) LT 1000) THEN BEGIN
        lev_name = '00'+lev_name
    ENDIF ELSE BEGIN
        IF (z_grid(loop11) LT 10000) THEN lev_name = '0'+lev_name
    ENDELSE
    print, lev_name
    level_list(loop11) = lev_name

    IF (loop11 EQ 0) THEN level_delta(0) = z_grid(0) ELSE $
        level_delta(loop11) = z_grid(loop11)-z_grid(loop11-1)

endFOR

;find the temp idlsav files
idlsaves = file_search(sourcedir+'*_temp_gridded_conc_*.idlsav', $
    count = n_idlsaves)

;get the names of just the files (no directories)
idlsave_names = file_basename(idlsaves)

;get list of dates
yrs = STRMID(idlsave_names, 0, 4)
mos = STRMID(idlsave_names, 4, 2)
das = STRMID(idlsave_names, 6, 2)
hrs = STRMID(idlsave_names, 8, 2)
jds = julday(mos, das, yrs, hrs)

;make list of uniq yrs, months for processing
yrs_1 = yrs[UNIQ(yrs, SORT(yrs))]
mos_1 = mos[UNIQ(mos, SORT(mos))]

;loop over each year, month to make permanent save files (save files
;will be 1 month long)
FOR loopy = 0, n_elements(yrs_1)-1 DO BEGIN
    FOR loopm = 0, n_elements(mos_1) -1 DO BEGIN

        jd_grid_s = 0

        this_mo = where((yrs EQ yrs_1(loopy)) AND (mos EQ mos_1(loopm)))
        IF (this_mo(0) NE (-1)) THEN BEGIN

            hrs_1 = hrs(this_mo)
            jds_1 = jds(this_mo)
            jds_1 = jds_1[UNIQ(jds_1, SORT(jds_1))]

;check to make sure first file starts at a save time and last
;file ends on an "off time" save times being 00, 06, 12, 18, off times
;being 03, 09, 15, 21
            IF ((hrs_1(0) MOD 6) EQ 0) THEN start_1 = 0 ELSE start_1 = 1

            IF ((hrs_1(n_elements(hrs_1)-1) MOD 6) EQ 0) THEN $
                end_1 = 4 ELSE end_1 = 2

            IF ((start_1 EQ 1) OR (end_1 EQ 2)) THEN adj = 0 ELSE adj = (1)

            gridded_conc = make_array(((n_elements(jds_1)-adj)/2), 360, 90, n_

```

```

03/19/09                                convert_forward_temps_to_gridded_final↵
_elements(level_list))
                                ;gridded_conc = [time, x, y, z]

;loop over all jd times, 2 at a time because we avg the 3-hr grids to
;get 6-hr grids

    FOR loopjd = start_1, n_elements(jds_1)-adj-1, 2 DO BEGIN
        IF (jd_grid_s(0) EQ 0) THEN jd_grid_s = jds_1(loopjd) ELSE $
            jd_grid_s = [jd_grid_s, jds_1(loopjd)]
        this_jd = jds_1(loopjd)
        next_jd = jds_1(loopjd+1)
        caldat, this_jd, this_mo, this_da, this_yr, this_hr
        caldat, next_jd, next_mo, next_da, next_yr, next_hr

        this_yr = strtrim(string(this_yr), 2)
        IF (this_hr LE 9) THEN this_hr = '0'+strtrim(string(this_hr), 2)↵
    ELSE $
        this_hr = strtrim(string(this_hr), 2)
        IF (this_da LE 9) THEN this_da = '0'+strtrim(string(this_da), 2)↵
    ELSE $
        this_da = strtrim(string(this_da), 2)
        IF (this_mo LE 9) THEN this_mo = '0'+strtrim(string(this_mo), 2)↵
    ELSE $
        this_mo = strtrim(string(this_mo), 2)

        next_yr = strtrim(string(next_yr), 2)
        IF (next_hr LE 9) THEN next_hr = '0'+strtrim(string(next_hr), 2)↵
    ELSE $
        next_hr = strtrim(string(next_hr), 2)
        IF (next_da LE 9) THEN next_da = '0'+strtrim(string(next_da), 2)↵
    ELSE $
        next_da = strtrim(string(next_da), 2)
        IF (next_mo LE 9) THEN next_mo = '0'+strtrim(string(next_mo), 2)↵
    ELSE $
        next_mo = strtrim(string(next_mo), 2)

;loop over release levels, averaging each level, adding it to the
    FOR looplev = 0, n_elements(level_list)-1 DO BEGIN

        this_file_name = this_yr+this_mo+this_da+this_hr+'_temp_gridde↵
d_conc_'+'$
            level_list(looplev)+'_idlsav'

        next_file_name = next_yr+next_mo+next_da+next_hr+'_temp_gridde↵
d_conc_'+'$
            level_list(looplev)+'_idlsav'

        restore, sourcedir+this_file_name
        temp_gridded_conc = GRIDDED_PPB
        restore, sourcedir+next_file_name

;NOTE: The next line converts from ng/m^3 to ng/m^2!!!
        temp_gridded_conc = (temp_gridded_conc+GRIDDED_PPB)*level_delt↵
a(looplev)/2

        gridded_conc(loopjd/2, *, *, looplev) = temp_gridded_conc

    ENDFOR                                ;end loop over levels
ENDFOR                                ;end loop over jd times

fname = yrs_1(loopy)+mos_1(loopm)+'_gridded_conc.idlsav'

```

```

fname = yrs_l(loopy)+mos_l(loopm)+'_gridded_conc.idlsav'
x_grid = x_info
y_grid = y_info
z_grid = fix(level_list)
jd_grid = jd_grid_s

IF (x_grid(1) EQ (-179)) THEN BEGIN

    temp_gridded_conc = gridded_conc(*, *, *, *)*0.0
    temp_gridded_conc(*, 1:359, *, *) = gridded_conc(*, 0:358, *, *)
    temp_gridded_conc(*, 0, *, *) = gridded_conc(*, 359, *, *)
    gridded_conc = float(temporary(temp_gridded_conc))

    x_grid(0:1) = x_grid(0:1)-1

ENDIF

check_owrite = file_Search(savedir+'gridded_conc/'+fname)

IF (check_owrite NE '') THEN BEGIN
    print, 'do you wish to overwrite the file '
    print, savedir+'gridded_conc/'+fname
    answer = ''
    read, answer, PROMPT = 'y or n '
    IF ((answer EQ 'y') OR (answer EQ 'Y')) THEN BEGIN
        save, gridded_conc, x_grid, y_grid, z_grid, jd_grid, $
            filename = savedir+'gridded_conc/'+fname, /compress
    ENDIF
ENDIF ELSE BEGIN
    save, gridded_conc, x_grid, y_grid, z_grid, jd_grid, $
        filename = savedir+'gridded_conc/'+fname, /compress
ENDELSE
ENDIF
ENDFOR
ENDFOR

cp_command = 'cp /local/reh/rcowen/flex_forward_running_output'+$
    '/NAA-CO/gridded_conc/@README '+$
    savedir+'gridded_conc/'
spawn, cp_command

ENDIF

IF (go_flux EQ 1) THEN BEGIN

;find the temp idlsav files
    idlsaves_std = file_Search(sourcedir+'*_temp_fluxes.idlsav', $
        count = n_idlsaves)

    idlsaves_spc = file_Search(sourcedir+'*_temp_special_fluxes.idlsav', $
        count = n_idlsaves)

;get the names of just the files (no directories)
    idlsave_names_std = file_basename(idlsaves_std)
    idlsave_names_spc = file_basename(idlsaves_spc)

;get list of dates
    yrs = STRMID(idlsave_names_std, 0, 4)
    mos = STRMID(idlsave_names_std, 4, 2)

```

03/19/09

convert_forward_temps_to_gridded_final

```
das = STRMID(idlsave_names_std, 6, 2)
hrs = STRMID(idlsave_names_std, 8, 2)
jds = julday(mos, das, yrs, hrs)

;make list of uniq yrs, months for processing
yrs_l = yrs[UNIQ(yrs, SORT(yrs))]
mos_l = mos[UNIQ(mos, SORT(mos))]

;loop over each year, month to make permanent save files (save files
;will be 1 month long)
FOR loopy = 0, n_elements(yrs_l)-1 DO BEGIN
  FOR loopm = 0, n_elements(mos_l) -1 DO BEGIN

    jd_grid_s = 0
    this_mo = where((yrs EQ yrs_l(loopy)) AND (mos EQ mos_l(loopm)))
    IF (this_mo(0) NE (-1)) THEN begin
      hrs_l = hrs(this_mo)
      jds_l = jds(this_mo)
      jds_l = jds_l[UNIQ(jds_l, SORT(jds_l))]
      restore, sourcedir+idlsave_names_std(0)
      sz_std = size(save_fluxes)
      std_flux = make_array(n_elements(jds_l), sz_std(1), sz_std(2), $
                           sz_std(3))
      ;      std_flux = make_array(n_elements(jds_l), sz_std(1), sz_std(2))

      restore, sourcedir+idlsave_names_spc(0)
      sz_spc = size(save_fluxes2)
      spc_flux = make_array(n_elements(jds_l), sz_spc(1), sz_spc(2), $
                           sz_spc(3), sz_spc(4))
      ;      spc_flux = make_array(n_elements(jds_l), sz_spc(1), sz_spc(2), $
      ;                           sz_spc(3))
    ;loop over all jd times

    FOR loopjd = 0, n_elements(jds_l)-1 DO BEGIN

      IF (jd_grid_s(0) EQ 0) THEN jd_grid_s = jds_l(loopjd) ELSE $
        jd_grid_s = [jd_grid_s, jds_l(loopjd)]
      this_jd = jds_l(loopjd)
      caldat, this_jd, this_mo, this_da, this_yr, this_hr

      this_yr = strtrim(string(this_yr), 2)
      IF (this_hr LE 9) THEN this_hr = '0'+strtrim(string(this_hr), 2)
    ELSE $
      this_hr = strtrim(string(this_hr), 2)
    IF (this_da LE 9) THEN this_da = '0'+strtrim(string(this_da), 2)
    ELSE $
      this_da = strtrim(string(this_da), 2)
    IF (this_mo LE 9) THEN this_mo = '0'+strtrim(string(this_mo), 2)
    ELSE $
      this_mo = strtrim(string(this_mo), 2)

    this_file_name = this_yr+this_mo+this_da+this_hr+'_temp_fluxes.i
dlsav'
    restore, sourcedir+this_file_name
    std_flux(loopjd, *, *, *) = SAVE_FLUXES
    ;      std_flux(loopjd, *, *) = SAVE_FLUXES

    this_file_name = this_yr+this_mo+this_da+this_hr+'_temp_special_
fluxes.idlsav'
    restore, sourcedir+this_file_name
    spc_flux(loopjd, *, *, *, *) = SAVE_FLUXES2
    ;      spc_flux(loopjd, *, *, *, *) = SAVE_FLUXES2
```

```

        enDFOR                                ;end loop over jd times

;check to see if a file exists for this period to add to it (other
;wise it would be replaced)
        curtains = [[-70, -70, 20, 60], [-50, -50, 20, 60], $
                    [-10, -10, 20, 60], [-70, -10, 60, 60], $
                    [-70, -10, 20, 20], [-70, -10, 20, 60]]

        fname1 = yrs_l(loopy)+mos_l(loopm)+'_std_flux.idlsav'
        jd_grid = jd_grid_s

        check_owrite = file_Search(savedir+'flux_standard/'+fname1)

        IF (check_owrite NE '') THEN BEGIN
            print, 'do you wish to overwrite the file '
            print, savedir+'flux_standard/'+fname1
            answer = ''
            read, answer, PROMPT = 'y or n '
            IF ((answer EQ 'y') OR (answer EQ 'Y')) THEN BEGIN

                save, std_flux, z_grid, jd_grid, curtains, flux_name, $
                    filename = savedir+'flux_standard/'+fname1, /compress

                fname2 = yrs_l(loopy)+mos_l(loopm)+'_spc_flux.idlsav'
                x_grid = [-70, -60, -50, -40, -30, -20, -10, 0]
                y_grid = [20, 25, 30, 35, 40, 45, 50, 55, 60]

                save, spc_flux, x_grid, y_grid, z_grid, jd_grid, $
                    filename = savedir+'flux_special/'+fname2, /compress

            ENDIF
        ENDIF ELSE BEGIN

            save, std_flux, z_grid, jd_grid, curtains, flux_name, age_grid, $
            $
                filename = savedir+'flux_standard/'+fname1, /compress

            fname2 = yrs_l(loopy)+mos_l(loopm)+'_spc_flux.idlsav'
            x_grid = [-70, -60, -50, -40, -30, -20, -10, 0]
            y_grid = [20, 25, 30, 35, 40, 45, 50, 55, 60]

            save, spc_flux, x_grid, y_grid, z_grid, jd_grid, age_grid, $
            filename = savedir+'flux_special/'+fname2, /compress

        ENDELSE
    ENDIF
ENDFOR
ENDFOR

        cp_command = 'cp /local/reh/rcowen/flex_forward_running_output/NAA-CO/fl
ux_standard/@README '+$
        savedir+'flux_standard/'
        spawn, cp_command

        cp_command = 'cp /local/reh/rcowen/flex_forward_running_output/NAA-CO/fl
ux_special/@README '+$
        savedir+'flux_special/'
        spawn, cp_command

    ENDIF

```

03/19/09

convert_forward_temps_to_gridded_final ↗

```
    return  
END
```


B.3.2 read_forward_flex_output.pro



03/19/09

read_forward_flex_output.pr

```
PRO read_forward_flex_output, outdir, go_pptv, go_conc, go_flux, $
    gzip = gzip, delete = delete

;+
;NAME: read_forward_flex_output
;
;PURPOSE:
;This program reads formatted (ascii) FLEXPART output, as designed by
;RCO for flexpart model runs @ MTU, and creates temporary idlsav files
;(to be processed by convert_forward_temps_to_gridded_final). It is
;highly recommended that either the gzip or delete keyword be set or
;the output directory will quickly grow very large. the delete keyword
;will ultimately be quicker, but gzip can be useful if some debugging
;is required.
;
;
;CATEGORY:
;
;FLEXPART program control
;
;CALLING SEQUENCE:
;
;read_forward_flex_output, outdir, go_pptv, go_conc, go_flux, $
;    gzip = gzip, delete = delete
;INPUTS:
;
;outdir - directory containing the ascii FLEXPART output files.
;
;go_pptv - if set to 1, then the mixing ratio files will be processed.
;
;go_conc - if set to 1, then the concentration files will be processed.
;
;go_flux - if set to 1, then the flux files will be processed.
;
; KEYWORD PARAMETERS:
;
;delete - if set, the original text files will be deleted as they are
;    processed.
;
;gzip - if set, the original text files will be gzipped as they are
;    processed.
;
; OUTPUTS:
;
;
; EXAMPLES:
;
;-

junk = ''

;read settings from header file
openr, infile, outdir+'header.txt', /get_lun

startline = ''
readf, infile, startline
readf, infile, format = '(3A10)', loutstep,loutaver,loutsample
readf, infile, format = '(6A8)', outlon0, outlat0, $
    numxgrid, numygrid, dxout, dyout
heightline = ''
readf, infile, heightline

readf, infile, timeline
```

```

readf, infile, format = '(I15)', n_spec_x_3
spec_arr = make_array(n_spec_x_3/3, /string)
numzgrid = 0
FOR loops = 0, (n_spec_x_3/3)-1 DO BEGIN
    readf, infile, junk
    readf, infile, junk
    readf, infile, junk
    spec_arr(loops) = strtrim(strmid(string(junk), 5, 20), 2)
ENDFOR
; FOR ijunk = 0, 4 DO BEGIN
;   readf, infile, junk
; ENDFOR
readf, infile, junk
ageline = ''
readf, infile, ageline

free_lun, infile

;extract dimensions of outgrid
x_grid = ((indgen(numxgrid)*dxout)+outlon0)
y_grid = ((indgen(numygrid)*dyout)+outlat0)

;extract height levels from heightline
nheights = fix(strtrim(strmid(heightline, 0, 9), 2))
z_grid = make_array(nheights)

FOR looph = 0, nheights-1 DO BEGIN
    z_grid(looph) = fix(strtrim(strmid(heightline, (10+10*looph), 10), 2))
ENDFOR

;extract start date from startline
start_date = strmid(strtrim(startline, 2), 0, 8)
start_sim = [fix(strmid(start_date, 0, 4)), $
             fix(strmid(start_date, 4, 2)), $
             fix(strmid(start_date, 6, 2))]

;extract age classes from ageline, ages are in seconds
nage = fix(strtrim(strmid(ageline, 0, 10), 2))

age_grid = make_array(nage)
FOR loopa = 0, nage-1 DO BEGIN
    age_grid(loopa) = (strtrim(strmid(ageline, (10+10*loopa), 10), 2))
ENDFOR

save, age_grid, z_grid, x_grid, y_grid, spec_arr, file = outdir+'header.id
lsav'

IF (go_pptv EQ 1) THEN BEGIN

;get file list of files in output dir
pptv_files = findfile(outdir+'grid_pptv_*.txt', count = n_pptv)
filebreak, pptv_files, name = pptv_names

;array to assist in naming idlsav files
pptv_string_names = strmid(pptv_names(*), 10, 10)

temp_read_array = make_array(n_elements(x_grid), n_elements(y_grid), $
                             n_elements(z_grid), n_elements(age_grid))

pptv_read_line = make_array(n_elements(y_grid), /float)

```

```

FOR loopp = 0, n_pptv-1 DO BEGIN
    jd = julday(strmid(pptv_names(loopp), 14, 2), strmid(pptv_names(loopp)
, 16, 2), $
                                strmid(pptv_names(loopp), 10, 4), strmid(pptv_names(loopp)
, 18, 2))

    print, 'opening file ', pptv_files(loopp)
    openr, infile, pptv_files(loopp), /get_lun
    readf, infile, junk

    WHILE (NOT(EOF(infile))) DO BEGIN
        readf, infile, junk

        IF ((junk NE ' 2 gridded') and $
            (junk NE '-999 999.0 999.0')) THEN BEGIN
;136             write(unitoutgridppt, '(4I5,90F15.5)')
;             + ix,kz,k,nage,
                x_lev = fix(strmid(junk, 0, 5))
                z_lev = fix(strmid(junk, 5, 5))
                spec = fix(strmid(junk, 10, 5))
                age_c = fix(strmid(junk, 15, 5))
                start_pos = 20
                FOR loopy = 0, (n_elements(y_grid)-1) DO begin
                    pptv_read_line(loopy) = float(strmid(junk, start_pos+loopy*15, 1
5))

                    ENDFOR
                    temp_read_array(x_lev, *, z_lev-1, age_c-1) = $
                        pptv_read_line/1000.
                    ENDF
                ENDWHILE

                free_lun, infile

;for ppt files, need to extract grid near pico w/ age classes. find the
;location of the cell w/ pico
                x_val = where(x_grid LE -28.4)
                x_val = max(x_val)
                y_val = where(y_grid LE 38.47)
                y_val = max(y_val)
                z_val = where(z_grid LE 2225)
                z_val = max(z_val)
                pico_xyz_ages = temp_read_array(x_val-3:x_val+3, y_val-2:y_val+2, $
                    *, *)
;
;                               z_val-2:z_val-2, *)
                fname = pptv_string_names(loopp)+'_temp_ppb_near_pico_grid.idlsav'
                grid_info = [x_grid(x_val-3:x_val+3), y_grid(y_val-2:y_val+2), $
                    z_grid]
;
;                               z_grid(z_val-2:z_val-2)]
                save, pico_xyz_ages, grid_info, age_grid, jd, filename = outdir+fname

                IF ((keyword_set(gzip)) AND (NOT keyword_Set(delete))) THEN BEGIN
                    command_zip_ppb = 'gzip '+pptv_files(loopp)
                    spawn, command_zip_ppb
                ENDF
                IF ((keyword_set(delete)) AND (NOT keyword_set(gzip))) THEN BEGIN
                    command_rm_ppb = '\rm '+pptv_files(loopp)
                    spawn, command_rm_ppb
                ENDF

            ENDFOR
;end loop over pptv files

```

```

enDIF

IF (go_conc EQ 1) THEN BEGIN

;get file list of files in output dir
conc_files = findfile(outdir+'grid_conc_*.txt', count = n_conc)
filebreak, conc_files, name = conc_names

;array to assist in naming idlsav files
conc_string_names = strmid(conc_names(*), 10, 10)

temp_read_array = make_array(n_elements(x_grid), n_elements(y_grid), $
                             n_elements(z_grid), n_elements(age_grid))

conc_read_line = make_array(n_elements(y_grid), /float)
FOR loopp = 0, n_conc-1 DO BEGIN

    jd = julday(strmid(conc_names(loopp), 14, 2), strmid(conc_names(loopp)
, 16, 2), $
               strmid(conc_names(loopp), 10, 4), strmid(conc_names(loopp)
, 18, 2))

    print, 'opening file ', conc_files(loopp)
    openr, infile, conc_files(loopp), /get_lun
    readf, infile, junk

    WHILE (NOT(EOF(infile))) DO BEGIN
        readf, infile, junk

        IF ((junk NE ' 2 gridded') and $
            (junk NE '-999 999.0 999.0')) THEN BEGIN
;136             write(unitoutgridppt, '(4I5,90F15.5)')
;             + ix,kz,k,nage,
                x_lev = fix(strmid(junk, 0, 5))
                z_lev = fix(strmid(junk, 5, 5))
                spec = fix(strmid(junk, 10, 5))
                age_c = fix(strmid(junk, 15, 5))
                start_pos = 20
                FOR loopy = 0, (n_elements(y_grid)-1) DO begin
                    conc_read_line(loopy) = float(strmid(junk, start_pos+loopy*15, 1
5))
                ENDFOR
                temp_read_array(x_lev, *, z_lev-1, age_c-1) = conc_read_line
            ENDFIF
        ENDWHILE

        free_lun, infile

;save gridded values, each level
        x_info = [max(x_grid), min(x_grid), n_elements(x_grid), abs(x_grid(1)-
x_grid(2))]
        y_info = [max(y_grid), min(y_grid), n_elements(y_grid), abs(y_grid(1)-
y_grid(2))]
        FOR loopl = 0, (n_elements(z_grid)-1) DO BEGIN
            gridded_ppb = total(temp_read_array(*, *, loopl, *), 4)
            z_lev = z_grid(loopl)
            IF ((z_lev LT 10000) AND (z_lev GE 1000)) THEN z_levn = '0'+strtrim(
string(fix(z_lev)), 2)

```

```

        IF (z_lev LT 1000) THEN z_levn = '00'+strtrim(string(fix(z_lev)), 2)
        IF (z_lev GE 10000) THEN z_levn = strtrim(string(fix(z_lev)), 2)
        fname = conc_string_names(loopp)+'_temp_gridded_conc_'+z_levn+'.idls'
av'
        save, gridded_ppb, x_info, y_info, z_lev, jd, file = outdir+fname
    ENDFOR

;save gridded values, total column
;NOT USED, commented out RCO, 3/31/2008
;   gridded_ppb = total(temp_read_array, 4)
;   gridded_ppb = total(gridded_ppb, 3)
;   z_levn = '00000'
;   z_lev = 0
;   fname = conc_string_names(loopp)+'_temp_gridded_conc_'+z_levn+'.idls'
v'
;   save, gridded_ppb, x_info, y_info, z_lev, jd, file = outdir+fname

    IF ((keyword_set(gzip)) AND (NOT keyword_set(delete))) THEN BEGIN
        command_zip_conc = 'gzip '+conc_files(loopp)
        spawn, command_zip_conc
    ENDIF
    IF ((keyword_set(delete)) AND (NOT keyword_set(gzip))) THEN BEGIN
        command_rm_conc = '\rm '+conc_files(loopp)
        spawn, command_rm_conc
    ENDIF

    ENDFOR                                ;end loop over conc files

ENDIF

    IF (go_flux EQ 1) THEN BEGIN

;get file list of files in output dir
        flux_files = findfile(outdir+'grid_flux*.txt', count = n_flux)
        filebreak, flux_files, name = flux_names

;array to assist in naming idlsav files
        flux_string_names = strmid(flux_names(*), 10, 10)

        temp_read_array = make_array(n_elements(x_grid), n_elements(y_grid), $
                                     n_elements(z_grid), n_elements(age_grid), 6
        )

        flux_read_line = make_array(n_elements(y_grid), /float)
        FOR loopp = 0, n_flux-1 DO BEGIN

            jd = julday(strmid(flux_names(loopp), 14, 2), $
                        strmid(flux_names(loopp), 16, 2), $
                        strmid(flux_names(loopp), 10, 4), $
                        strmid(flux_names(loopp), 18, 2))

            print, 'opening file ', flux_files(loopp)
            openr, infile, flux_files(loopp), /get_lun
            readf, infile, junk

            WHILE (NOT(EOF(infile))) DO BEGIN
                readf, infile, junk

                IF ((junk NE ' 2 gridded') and $

```

```

        (junk NE ' -999 999.0 999.0')) THEN BEGIN
;36      write(unitflux, '(5I5,90F15.5)')
;      +      1,k,nage,kz,ix,
;      +      (1.e12*fluxe(ix,jy,kz,k,nage)/
;      +      areaeast(ix,jy,kz)/outstep,jy=0,numygrid-1)
      flux_dir = fix(strmid(junk, 0, 5))
      spec = fix(strmid(junk, 5, 5))
      age_c = fix(strmid(junk, 10, 5))
      z_lev = fix(strmid(junk, 15, 5))
      x_lev = fix(strmid(junk, 20, 5))

      start_pos = 25
      FOR loopy = 0, (n_elements(y_grid)-1) DO begin
5))        flux_read_line(loopy) = float(strmid(junk, start_pos+loopy*15, 1
          ENDFOR
          temp_read_array(x_lev, *, z_lev-1, age_c-1, flux_dir-1) = $
            flux_read_line
        ENDIF
      ENDWHILE

      free_lun, infile

;standard flux requests
;loop over each flux of interest, select cells of interest
; 2 east, 1, west, 1, north, 1 south, 3 down
;curtain = [lon1,lon2,lat1,lat2,height1,height2]
;      flux_num = [1, 2, 3, 4, 5, 6]
;      flux_name = ['east', 'west', 'south', 'north', 'up', 'down']
      flux_name = ['east1', 'east2', 'west', 'south', 'north', 'down']
      flux_order = [1, 1, 2, 3, 4, 6]
      curtains = [[-70, -70, 20, 60], [-50, -50, 20, 60], $
                  [-10, -10, 20, 60], [-70, -10, 60, 60], $
                  [-70, -10, 20, 20], [-70, -10, 20, 60]]
      save_fluxes = make_array(n_elements(flux_order), n_elements(z_grid), $
                              n_elements(age_grid))
      FOR loopf = 0, n_elements(flux_order)-1 DO BEGIN

        x_loc = where((x_grid GE curtains(0, loopf)) AND $
                      (x_grid LE curtains(1, loopf)))
        y_loc = where((y_grid GE curtains(2, loopf)) AND $
                      (y_grid LE curtains(3, loopf)))
        flux_curtain = temp_read_array(x_loc, y_loc, *, *, flux_order(loopf))
-1)      flux_curtain = total(flux_curtain, 1);sum lon values
          flux_curtain = total(flux_curtain, 1);sum lat values
;      flux_curtain = total(flux_curtain, 2);sum age classes
          save_fluxes(loopf, *, *) = flux_curtain

      ENDFOR

      fname = flux_string_names(loopf)+'_temp_fluxes.idlsav'
      save, save_fluxes, z_grid, jd, flux_name, filename = outdir+fname

      d_lon = 10
      lon_r = [-70, 0]
      n_lon = ((lon_r(1)-lon_r(0))/d_lon)+1
      lon = lon_r(0)
      d_lat = 5
      lat_r = [20, 60]
      n_lat = ((lat_r(1)-lat_r(0))/d_lat)+1
      lat = lat_r(0)

```

```

save_fluxes2 = make_Array(n_lon, n_lat, nheights, n_elements(age_grid
))

FOR looplon = 0, (n_lon-1) DO BEGIN
  FOR looplat = 0, (n_lat-1) DO BEGIN
    x_loc = where((x_grid GE (lon_r(0)+looplon*d_lon)) $
                  AND (x_grid LE (lon_r(0)+(looplon+1)*d_lon)))
    y_loc = where((y_grid GE (lat_r(0)+looplat*d_lat)) $
                  AND (y_grid LE (lat_r(0)+(looplat+1)*d_lat)))

    flux_curtain = temp_read_array(x_loc, y_loc, *, *, 0)
    flux_curtain = total(flux_curtain, 1) ;sum lon values
    flux_curtain = total(flux_curtain, 1) ;sum lat values
    flux_curtain = total(flux_curtain, 2) ;sum age classes
    save_fluxes2(looplon, looplat, *, *) = flux_curtain

  enDFOR
ENDFOR

fname = flux_string_names(loopp)+'_temp_special_fluxes.idlsav'
save, save_fluxes2, z_grid, jd, filename = outdir+fname

IF ((keyword_Set(gzip)) AND (NOT keyword_set(delete))) THEN BEGIN
  command_zip_flux = 'gzip '+flux_files(loopp)
  spawn, command_zip_flux
ENDIF
IF ((keyword_set(delete)) AND (NOT keyword_set(gzip))) THEN BEGIN
  command_rm_flux = '\rm '+flux_files(loopp)
  spawn, command_rm_flux
ENDIF

ENDFOR ;end loop over flux files

ENDIF

return
END

```


B.3.3 run_forward_flexpart.pro



```

PRO run_forward_flexpart, starttr, endr, region, index, spec, $
    emiss_source = emiss_source, $
    use_pre_release = use_pre_release, $
    maxpt = maxpt, $
    DO_NOT_run_flex = DO_NOT_run_flex, $
    instant = instant, no_kernel = no_kernel, $
    uctl = uctl, uifine = uifine, $
    uvlevels = uvlevels, uages = uages

;+
;NAME: run_forward_flexpart
;
;PURPOSE:
;
;This program writes the files required to run FLEXPART and then
;compiles and runs FLEXPART. The program is able to automatically
;create the emissions for each continent and has the option to load a
;pre-made releases file. This program has reasonable defaults for most
;of the settings required (e.g., the CTL time step) and focuses on the
;few options that will need to be changed frequently between
;simulations (e.g., CO or NO2).
;
;CATEGORY:
;
;FLEXPART program control
;
;CALLING SEQUENCE:
;
;run_forward_flexpart, starttr, endr, region, index, $
;    emiss_source = emiss_source, $
;    use_pre_release = use_pre_release, $
;    spec = spec, maxpt = maxpt, $
;    do_not_run_flex = do_not_run_flex, $
;    instant = instant, no_kernel = no_kernel
;INPUTS:
;
;starttr - start date of the simulation in julian days. Make sure this
;    date includes model spinup, i.e., the difference between the
;    start date and the day of desired output is equal to or
;    greater than the maximum age of tracer in the program,
;    typically 20 days.
;
;endr - end date of the simulation in julian days.
;
;region - location code for EDGAR emission. 1=NA, 2=EU, 3=AS, 4=SA,
;    5=AF, 6=AU, 7=World
;
;index - the output folder number (string format). As of 8/19/2008
;    these can be "" (blank), "2", "5", "8", "11", "12", and
;    "13". this number indicates the directory number output will
;    be saved to:
;
;    '/local/reh/rcowen/flex_forward_running_output/processing'+index
;
;spec - the FLEXPART species number (string format). This input
;    affects age class distributions saved, the inclusion or
;    exclusion of wet removal (not NOx) and the inclusion or
;    exclusion of daily and weekly emissions variation).
;    The options are:
;    "3" (stratospheric O3)
;    "22" (CO w/o emissions variation, can be used for biomass
;    burning emissions),
;    "25" (CO w/ anthropogenic emissions variation)

```

```

;      "23" (NOx tracer, no removal and emissions variation)
;      "29" (NOx tracer, wet removal as NO2 and emissions variation)
;      Other species can be identified, but must be included in the
;      FLEXPART SPECIES file, which will use the appropriate removal
;      and emissions variation and a default age distributions that
;      will be saved.
;
;
; KEYWORD PARAMETERS:
;
;emiss_source - optional emissions source file. if not specified, the
;                EDGAR emissionf file for CO or NOx will be used,
;                according to species set in the spec input.
;
;
;use_pre_release - this is the location and name (in string format) of
;                  a pre-made RELEASES file. If specified, this
;                  program will not generate a RELEASES file and the
;                  'index' input will be ignored. the 'spec' input
;                  will still determine the age class, but the species
;                  that FLEXPART will use is specified in the RELEASES
;                  file.
;
;maxpt - the number of particles to use in the simulation. the default
;        is 5 million particles IN THE MODEL AT ONE TIME and AFTER
;        MODEL SPINUP.
;
;do_not_run_flex - If set, FLEXPART will be not compiled and run in the
;                  directory determined by the 'index'. Otherwise, FLEXPART
;                  will be compiled and run. this can be useful if you have
;                  some settings that are different than the defaults
;                  provided in this program (e.g., only want concentrations
;                  and not mixing ratios). if set, you will need to compile
;                  (gmake) and execute (./FLEXPART) in the appropriate
;                  directory.
;
;instant - if set, instantaneous output is given. otherwise, averaged
;          output is given.
;
;no_kernel - if set, the kernel, which distributes the mass of each
;            particle across 4 grid cells, will not be used.
;
;uctl and uifine - allows one to specify a larger ctl or ifine. the
;                  default values are 2 for ctl and 6 for ifine.
;
;uvlevels - allows the user to set a different vertical resolution
;           than the default.
;
;uages - allows users to specify age classes other than the
;        defaults. this is an n element array, where each entry is the
;        maximum age in each class, in days. for example, uages =
;        [1,2,3,4] would produce for age classes, 0-1, 1-2, 2-3, and
;        3-4 days old.
;
; OUTPUTS:
;
;
; EXAMPLES:
;
;-

```

```

spawn, 'sleep 20'
;This section will compile the various FLEXPART runtime options. Each
;selection will be briefly explained, with some recommendations for
;basic options. Further details will need to be obtained from the
;FLEXPART users manual.

direction = (1)                ; 1= forward

; starttr AND endr should be in julday, convert them to
caldat, starttr, mos, das, yrs, hrs
caldat, endr, moe, dae, yre, hre
start_run = [yrs, mos, das, hrs] ;[YYYY,MM,DD,HH]
end_run = [yre, moe, dae, hre] ;[YYYY,MM,DD,HH]

emissions_region = region      ; 1=NA, 2=EU, 3=AS, 4=SA, 5=AF, 6=AU, 7=W
orld

;see below for explanations of these options. they are placed here
;because they are also commonly modified options

IF (keyword_Set(uctl)) THEN ctl = uctl ELSE ctl = '2'
IF (keyword_Set(uifine)) THEN ifine = uifine ELSE ifine = '6'
t_output = 3 ;3=every 3 hrs
iout = '3' ;3=all, concentrations and mixing ratios.
IF (keyword_set(uvlevels)) THEN vlevels = uvlevels ELSE $
vlevels = [300, 1000, 1500, 2000, 2500, 3000, 4000, 5000, 7500, 10000, 1
5000]

IF (keyword_set(spec)) THEN BEGIN
species = spec
ENDIF ELSE BEGIN
species = ['22']
ENDELSE

IF (keyword_set(uages)) THEN age_classes = uages ELSE BEGIN
CASE (species(0)) OF
;CO w/o emission variation 22=
'22': age_classes = [4, 5, 6, 7, 8, 9, 10, 12, 14, 16, 18, 20]
;Anth (emiss variation) 25=CO
'25': age_classes = [3, 4, 5, 6, 7, 8, 9, 10, 11, 12, 14, 16, 18, 20]
;Stratosphere O3
'2': age_classes = [1, 2, 3, 4, 5, 6, 7, 8, 9, 10, 12, 14, 16, 18, 20]
;Anth (emiss variation) 23=NOxT, 29=NOxW
'23': age_classes = [3, 4, 5, 6, 7, 8, 9, 10, 11, 12, 14, 16, 18, 20]
'29': age_classes = [3, 4, 5, 6, 7, 8, 9, 10, 11, 12, 14, 16, 18, 20]
ELSE: age_classes = [3, 4, 5, 6, 7, 8, 9, 10, 12, 14, 16, 18, 20]
ENDCASE
ENDELSE

IF (keyword_set(maxpt)) THEN begin
;this option would be used to determine the max particles needed to
;maintain a certain number of particles for a run that the releases
;file is created automatically
n_days = endr-starttr
max_days = float(max(age_classes))
maxpart = maxpt*n_days/max_days
maxpart = strtrim(string(maxpart), 2)
ENDIF ELSE maxpart = '5000000' ; max is 20,000,000 w/ 20 age classes, 11 h

```

ht

```

IND_SOURCE = '1'                ;1=mass units, 2=mass mixing ratio units
IND_RECEPTOR = '1'            ;1=mass units, 2=mass mixing ratio units

;Options for the OUTGRID file. OUTGRID defines the grid for which the
;model outputs data for. this grid must be less (smaller) than the
;input met grid. the default ECMWF grid is -179 to 180 long, -90 to 90
;lat.

;OUTLONLEFT is the GEOGRAPHICAL LONGITUDE OF LOWER LEFT CORNER OF
;OUTPUT GRID.
    OUTLONLEFT = (-179)          ;need to change to -180
    OUTLATLOWER = (0)

;NUMXGRID is the NUMBER OF GRID POINTS IN X DIRECTION (longitude).
    NUMXGRID = (360)
    NUMYGRID = (90)

;DXOUTLON is the delta in the x direction (longitude). if
;outlonleft = -179, numxgrid = 360, and DXOUTLON = 1.0, then the
;longitudinal grid covers the globe from -179 to 180
    DXOUTLON = 1.0
    DYOUTLAT = 1.0

;END options for OUTGRID

;frequency of output. this is an important parameter and should be
;chosen carefully. the recommended value is either 3 hrs or 6 hrs. If
;this is too
;large, you will get a very course response (think trajectory end
;points output only every 2 days). this parameter should match the
;next parameter for best results. it should also be noted that
;doubling this number will double the number of output files (and
;output size). time is in hours below.
; t_output = 3

;averaging time of output. should match above parameter and is set
;that way by default. i do not know reasons why you would set this
;differently.
    t_average = t_output

;sample rate of output. this parameter is how often the model data is
;sampld to produce the average. if the data is sampled @ 0.25 hrs
;with a 3 hr avg time (above), then you have 12 "data points" in the
;averaged result. therefor, the sample rate should be some small
;fraction of the averaging time (~1/10th). for a 3 hrs average time, a
;15 min (0.25) sample rate is recommended
    sample_rate = 0.25

;time constant for splitting particles. this should not be changed.
    part_split = '999999'

;SYNCHRONISATION INTERVAL OF FLEXPART. All processes are synchronized
;with this time interval, all other time constants must be multiples
;of this value. Output interval and time average of output must be at
;least twice this value. Generally, this is set to the sample rate
;(above), as thats usually the smallest time contant.
    sync_time = 0.25

;CTL (see FLEXPART documentation for more information). A negative

```

```

;CTL (see FLEXPART documentation for more information). A negative
;value produces the fastest model run, but mostly ignores
;turbulence. A larger, positive value describes turbulence better, but
;takes longer to run. Negative value is probably OK for US anthro
;forward sim, but should be positive for more specific applications
;(e.g., retroplumes, modelling individual fires).
;ctl = '2'

;IFINE. The reduction factor for time step used for vertical
;wind. having this be a larger number will speed computational times
;with large values of ctl (above), but still give a good
;representation of turbulence.
;ifine = '4'

;IOUT, determines how the output shall be made: concentration (ng/m3,
;Bq/m3), mixing ratio (pptv), or both, or plume trajectory mode, or
;concentration + plume trajectory mode. In plume trajectory mode,
;output is in the form of average trajectories. 1 CONC. (RESID. TIME
;FOR BACKWARD RUNS) OUTPUT,2 MIX. RATIO OUTPUT,3 BOTH,4 PLUME
;TRAJECT.,5=1+4. for backward runs, 5 is highly recommended.
; iout = '2'

;IPOUT, determines whether particle positions are outputted (in
;addition to the gridded concentrations or mixing ratios) or not. 0=no
;output, 1 output every output interval, 2 only at end of the
;simulation. This is an advanced setting. For backward runs, this
;should be 0 as backward runs are short and redone quickly if the
;program crashes. If you are doing longer runs, this should be 1, if
;you are doing continued runs (forward simulations that are longer
;than ~1 month), this should be either 1 or 2 (2 recommended for space
;considerations).
  ipout = '0'

;LSUBGRID, LCONVECTION, these parameters turn on and off the use of
;topography (LSUBGRID) and the convection scheme
;(LCONVECTION). generally speaking, these should always be ON (value
;of 1), though turning them off will speed up a model run. LSUBGRID
;may not be as important for backward runs from PICO.
  LSUBGRID = '1' ;(1 = on, 0 = off)
  LCONVECTION = '1' ;(1 = on, 0 = off)

;LAGESPECTRA, this turns on and off the calculation of age classes. if
;this is turned off, the file options/AGECLASSES must be available and
;updated. this is normally turned on for forward runs, though not
;exclusively. if it is off, particles will be carried in the model
;until the model run ends. options for AGECLASSES are below.
  LAGESPECTRA = '1' ;(1 = on, 0 = off)

;IPIN, this continues a model run with dumped particle locations. this
;is an advanced option and would be used with ipout (above). this is
;generally turned off.
  IPIN = '0' ;(1 = on, 0 = off)

;IFLUX, this calculates fluxes in each grid cell. this is normally
;turned off.
  IFLUX = '1' ;(1 = on, 0 = off)

;MDOMAINFILL, this turns on and off the calculation of domain filling
;trajectories. this is normally turned off.
  IF (spec EQ '2') THEN MDOMAINFILL = '2' ELSE MDOMAINFILL = '0' ;(1 = on, 0
= off)

```

```

;IND_SOURCE, switches between different units for concentrations at
;the source. NOTE that in backward simulations the release of
;computational particles takes place at the "receptor" and the
;sampling of particles at the "source". this is gnerally set to 1
;(mass units).
;IND_SOURCE = '1' ;1=mass units (bwd=concentration), 2=mass mixing ratio uni
ts

;IND_RECEPTOR, switches between different units for concentrations at
;the receptor. this is normally not used because we dont calculate
;receptor concentrations because they are calculated at the surface
;and PICO isnt in the topography.
;IND_RECEPTOR = '1' ;1=mass units (for bwd-runs = concentration), 2=mass mi
xing ratio units

;MQUASILAG, indicates whether particles shall be numbered
;consecutively (1) or with their release location number (0). The
;first option allows tracking of individual particles using the
;partposit output files. this is an advanced setting and is normally
;not on.
    MQUASILAG = '0' ;(1 = on, 0 = off)

;NESTED_OUTPUT decides whether model output shall be made also for a
;nested output field (normally with higher resolution). this is an
;advanced setting and is never used, unless we have additional met
;data.
    NESTED_OUTPUT = '0' ;(1 = on, 0 = off)

;END of options for the COMMAND file

;options for the includepar files in the main program directory.
;includepar has 4 variables that should be modified for each
;run. these are maxpart, maxpoint, maxspec, maxpointspec (lines
;169 & 170).
;maxpart = '1000000' ; maximum number of particles allowed. releases
; will automatically be scalled to this value
; maxpoint = '30' ; number of releases. for backward runs, ma
x is around
; ; 60 for some reason. for forward runs, need
d ~3000.
    maxspec = strtrim(string(long(n_elements(species))), 2)
;END of options for includepar

;this section will create a release file for a standard
;CO forward run. the following will read an emissions file from the
;EDGAR emission inventory (2000 estimated values) for all
;anthropogrnic emissions. you can edit the program called
;(write_emissions.pro) to use other emissions files for CO (e.g.,
;biomass burning). you can select which continents for which to model
;CO emissions.
    edgar_start_date = start_run
    edgar_end_date = END_run

;options for pathnames. please select a location for your
;ouput. WARNING: if a new folder is not selected for output, then any
;previous results WILL be over written :WARNING.
    outpath = '/local/reh/rcowen/flex_forward_running_output/processing'+index
+ '/'

;if you are using your own folder to compile and run this program,

```

```

;then you will need to specify the location of this folder
;here. otherwise, you are running this in the group directory.
    filepath = '/local/reh/rcowen/group_flex_idl/REH_group_FLEXPART'+index+'/'

;*****
;*****
;*****
;END OF OPTIONS. The rest of the program uses the options set above to
;compile and run FLEXPART. Nothing below this point should be
;modified. Plotting procedures are elsewhere.
;*****
;*****
;*****

;copy all the flexpart program files to the specified directory
    source_dir = '/local/reh/rcowen/group_flex_idl/FLEXPART_source/'

;first copy all the "default" files (e.g., those that do not need to
;be modified). this will actually copy over ALL files, then rewrite
;the ones that need to be modified.
    command1 = '\cp -r '+source_dir+'* '+filepath+'.'
    spawn, command1

;copy the non-run specific files that have been modified to the
;program directory. these are calcmatrix.f, writeheader.f,
;conccoutput.f, and makefile
    mod_dir = '/local/reh/rcowen/group_flex_idl/modified/'
    MOD_list = ['calcmatrix.f', 'writeheader.f', 'conccoutput.f', $
    'makefile', 'readwind.f', 'fluxoutput.f']

    IF (keyword_set(instant)) THEN MOD_list = [MOD_list, 'timemanager_instant.
f']

    FOR MOD_loop = 0, (n_elements(MOD_list)-1) DO BEGIN
        command_mod = '\cp '+MOD_dir+MOD_list(MOD_loop)+' '+filepath
        spawn, command_mod
    ENDFOR

    IF (keyword_set(no_kernel)) THEN $
        spawn, '\cp '+MOD_dir+'conccalc_no_kernel.f'+ ' '+filepath+$
        'conccalc.f'

;copy the right makefile for the system to the program path
    makef = 'makefile_L_64_RHEL5'

    command_make_mv = '\cp '+MOD_dir+makef+' '+filepath+'makefile'
    spawn, command_make_mv

;Write all the run-specific files, as dictated by the options set
;above.

;*****write releases file*****
    IF (keyword_set(use_pre_release)) THEN BEGIN
        command = '\cp '+use_pre_release+' '+filepath+'options/RELEASES'
        spawn, command
        numpart = file_lines(filepath+'options/RELEASES')
        maxpoint = strtrim(string(long(numpart)), 2)
        print, 'using releases file'
        print, use_pre_release
    ENDIF ELSE BEGIN

```



```

line_releases = make_array(14, 2, /string)

line_releases(0, 0) = '*****'
*****
line_releases(1, 0) = '*'
*
line_releases(2, 0) = '*'
*
line_releases(3, 0) = '*'
*
line_releases(4, 0) = '*   Input file for the Lagrangian particle disper
sion model FLEXPART'
line_releases(5, 0) = '*   Please select your optio
ns
line_releases(6, 0) = '*'
*
line_releases(7, 0) = '*'
*
line_releases(8, 0) = '*'
*
line_releases(9, 0) = '*****'
*****
line_releases(10, 0) = '++++++'
++++++
openw, outfile, filepath+'options/RELEASES', /GET_LUN

FOR i = 0, 10 DO BEGIN
  printf, outfile, line_releases(i, 0)
ENDFOR

line_releases(11, 0) = maxspec+'                               Total number o
f species emitted'
printf, outfile, line_releases(11, 0)
FOR loops = 0, (n_elements(species)-1) DO BEGIN
  line_releases(12, 0) = species(loops)+'                               Index
of species in file SPECIES'
  printf, outfile, line_releases(12, 0)
ENDFOR
line_releases(13, 0) = '=====
=====

printf, outfile, line_releases(13, 0)

IF (spec NE '2') THEN BEGIN

;read in EDGAR emissions and return emissions on a lat/lon
;grid. emissions are in kg CO/yr or NOx/yr. these will need to be
;scaled for the release file for the actual length of the simulation.
  IF NOT(keyword_set(emiss_source)) THEN BEGIN
    CASE spec OF
      '22': emissions_source = '/home/darcy/rcowen/rco_research/flex_dir
s/forward_pros/edgar/tot_sparse_non_bb_emissions.idlsav'
      '25': emissions_source = '/home/darcy/rcowen/rco_research/flex_dir
s/forward_pros/edgar/tot_sparse_non_bb_emissions.idlsav'
      '23': emissions_source = '/home/darcy/rcowen/rco_research/flex_dir
s/forward_pros/edgar/tot_sparse_nox_non_bb_emissions.idlsav'
      '29': emissions_source = '/home/darcy/rcowen/rco_research/flex_dir
s/forward_pros/edgar/tot_sparse_nox_non_bb_emissions.idlsav'

    ENDCASE
  ENDIF ELSE emissions_source = emiss_source

```

```

print, 'restoring file for emissions:'
print, emissions_source
restore, emissions_source, /verbose

IF (emissions_region NE 7) THEN BEGIN
  use_edgar = where(edgar_emissions(3, *) EQ emissions_region(0))
  emissions = edgar_emissions(*, use_edgar)
ENDIF ELSE emissions = edgar_emissions

edgar_start_jd = julday(edgar_start_date(1), edgar_start_date(2), $
                        edgar_start_date(0), edgar_start_date(3))
edgar_end_jd = julday(edgar_end_date(1), edgar_end_date(2), $
                      edgar_end_date(0), edgar_end_date(3))

n_days = edgar_end_jd - edgar_start_jd
yr_Days = julday(1, 1, edgar_start_date(0), 00, 00) - $
          julday(1, 1, edgar_start_date(0) - 1, 00, 00)
frac = n_days / yr_Days
emissions(2, *) = emissions(2, *) * frac
tot_mass = emissions(2, *)
tot_mass = total(tot_mass)
numpart = (maxpart * (emissions(2, *) / tot_mass))

string_edgar_start_date = strtrim(string(edgar_start_date), 2)
FOR j = 0, 3 DO IF (string_edgar_start_date(j) LE 9) THEN $
  string_edgar_start_date(j) = '0' + string_edgar_start_date(j)

string_edgar_end_date = strtrim(string(edgar_end_date), 2)
FOR j = 0, 3 DO IF (string_edgar_end_date(j) LE 9) THEN $
  string_edgar_end_date(j) = '0' + string_edgar_end_date(j)

;loop over each release and write to file
FOR i = 0., (n_elements(emissions(0, *))-1) DO BEGIN
  IF (numpart(i) GE 1) THEN BEGIN

    printf, outfile, string_edgar_start_date(0) + $
      string_edgar_start_date(1) + string_edgar_start_date(2) + $
      ' ' + string_edgar_start_date(3) + '0000 '

    printf, outfile, string_edgar_end_date(0) + $
      string_edgar_end_date(1) + string_edgar_end_date(2) + $
      ' ' + string_edgar_end_date(3) + '0000 '

    printf, outfile, format = '(f9.4)', emissions(0, i)
    printf, outfile, format = '(f9.4)', emissions(1, i)
    printf, outfile, format = '(f9.4)', (emissions(0, i) + 1)
    printf, outfile, format = '(f9.4)', (emissions(1, i) + 1)

    printf, outfile, format = '(i9)', 2

    printf, outfile, format = '(f10.3)', 0
    printf, outfile, format = '(f10.3)', 300

;num part
    printf, outfile, format = '(i9)', numpart(i)
;mass part,
    e9.4
    FOR loops = 0, (n_elements(species)-1) DO BEGIN
      printf, outfile, format = '(e11.4)', emissions(2, i)
    ENDFOR
;    printf, outfile, format = '(e11.4)', emissions(2, i)
;name of release, format='(a40)'
    printf, outfile, 'pico_forw_lon_' + strtrim(string(long(emissions(0,
i))), 2) + $

```

03/19/09

run_forward_flexpart.pr

```

        '_lat_'+strtrim(string(long(emissions(1, i))), 2)+'_sp'+sp
ecies(0)
        printf, outfile, '++++++'
++++++'
        ENDDIF
    ENDFOR
    free_lun, outfile
;end writing RELEASES file*****

    maxpoint = strtrim(string(long(n_elements(numpart))), 2)

enDIF ELSE BEGIN                ;making strat ozone, make different release
file

    edgar_start_jd = julday(edgar_start_date(1), edgar_start_date(2), $
                           edgar_start_date(0), edgar_start_date(3))
    edgar_end_jd = julday(edgar_end_date(1), edgar_end_date(2), $
                          edgar_end_date(0), edgar_end_date(3))

    n_days = edgar_end_jd-edgar_start_jd
    yr_Days = julday(1, 1, edgar_start_date(0), 00, 00)-$
              julday(1, 1, edgar_start_date(0)-1, 00, 00)

    string_edgar_start_date = strtrim(string(edgar_start_date), 2)
    FOR j = 0, 3 DO IF (string_edgar_start_date(j) LE 9) THEN $
        string_edgar_start_date(j) = '0'+string_edgar_start_date(j)

    string_edgar_end_date = strtrim(string(edgar_end_date), 2)
    FOR j = 0, 3 DO IF (string_edgar_end_date(j) LE 9) THEN $
        string_edgar_end_date(j) = '0'+string_edgar_end_date(j)

    printf, outfile, string_edgar_start_date(0)+$
               string_edgar_start_date(1)+string_edgar_start_date(2)+$
               ' '+string_edgar_start_date(3)+'0000 '

    printf, outfile, string_edgar_end_date(0)+$
               string_edgar_end_date(1)+string_edgar_end_date(2)+$
               ' '+string_edgar_end_date(3)+'0000 '

    printf, outfile, format = '(f9.4)', -179 ;lon1
    printf, outfile, format = '(f9.4)', 10 ;lat1
    printf, outfile, format = '(f9.4)', 178 ;lon2
    printf, outfile, format = '(f9.4)', 89 ;lat2
    printf, outfile, format = '(i9)', 2
    printf, outfile, format = '(f10.3)', 1000
    printf, outfile, format = '(f10.3)', 20000
;num part
    printf, outfile, format = '(i9)', maxpart
;mass part,      e9.4
    printf, outfile, format = '(e11.4)', 1e0
;name of release, format='(a40)'
    printf, outfile, 'pico_strat'
    printf, outfile, '++++++'
+++++'
    free_lun, outfile

    maxpoint = '100'

ENDELSE
ENDELSE
```

```

;end writing RELEASES file*****

;write OUTGRID file*****

line_outgrid = make_array(32, 2, /string)

line_outgrid(0, 0) = '*****'
*****
line_outgrid(2, 0) = '*'
      *
line_outgrid(3, 0) = '*'      Input file for the Lagrangian particle disper
sion model FLEXPART      *
line_outgrid(4, 0) = '*'      Please specify your output g
rid      *
line_outgrid(5, 0) = '*'
      *
line_outgrid(6, 0) = '*****'
*****
line_outgrid(7, 0) = ''
line_outgrid(8, 0) = '1.  -----.----      4X,F11.4'
line_outgrid(9, 0) = ''
line_outgrid(9, 1) = '(A4,F11.4)'
line_outgrid(10, 0) = '      OUTLONLEFT      (left boundary of the first g
rid cell - not its centre, 0)'
line_outgrid(11, 0) = ''
line_outgrid(12, 0) = '2.  -----.----      4X,F11.4'
line_outgrid(13, 0) = ''
line_outgrid(13, 1) = '(A4,F11.4)'
line_outgrid(14, 0) = '      OUTLATLOWER      (lower boundary of the first
grid cell - not its centre, 0)'
line_outgrid(15, 0) = ''
line_outgrid(16, 0) = '3.  -----      4X,I5'
line_outgrid(17, 0) = ''
line_outgrid(17, 1) = '(A4,I5)'
line_outgrid(18, 0) = '      NUMXGRID'
line_outgrid(19, 0) = ''
line_outgrid(20, 0) = '4.  -----      4X,I5'
line_outgrid(21, 0) = ''
line_outgrid(21, 1) = '(A4,I5)'
line_outgrid(22, 0) = '      NUMYGRID'
line_outgrid(23, 0) = ''
line_outgrid(24, 0) = '5.  -----.----      4X,F10.3'
line_outgrid(25, 0) = ''
line_outgrid(25, 1) = '(A4,F10.3)'
line_outgrid(26, 0) = '      DXOUTLON'
line_outgrid(27, 0) = ''
line_outgrid(28, 0) = '6.  -----.----      4X,F10.3'
line_outgrid(29, 0) = ''
line_outgrid(29, 1) = '(A4,F10.3)'
line_outgrid(30, 0) = '      DYOUTLAT'
line_outgrid(31, 0) = ''

outgrid_options = [OUTLONLEFT, OUTLATLOWER, NUMXGRID, NUMYGRID, DXOUTLON,
DYOUTLAT]

openw, outfile, filepath+'options/OUTGRID', /GET_LUN
j_out = 0
FOR i = 1, 31 DO BEGIN
  IF (line_outgrid(i, 1) EQ '') THEN BEGIN
    printf, outfile, line_outgrid(i, 0)
  
```

```

        ENDIF ELSE BEGIN
            printf, outfile, format = line_outgrid(i, 1), line_outgrid(i, 0), outg
rid_options(j_out)
            j_out = j_out+1
        ENDELSE
    ENDFOR

    FOR i = 0, (n_elements(vlevels)-1) DO BEGIN
        printf, outfile, 'XX. -----' 4X, F7.1'
        printf, outfile, format = '(A4,F7.1)', '', vlevels(i)
        printf, outfile, '    LEVEL X    HEIGHT OF LEVEL (UPPER BOUNDARY)'
        printf, outfile, ''
    ENDFOR

    printf, outfile, ''
    free_lun, outfile

;END print OUTGRID*****

;write COMMAND file*****

    line_command = make_array(31, /string)

    line_command(1) = '*****'
    line_command(2) = '*'
    line_command(3) = '*    Input file for the Lagrangian particle dispersio
n model FLEXPART    *'
    line_command(4) = '*    Please select your options *'
    line_command(5) = '*'
    line_command(6) = '*****'
    line_command(7) = ''
    line_command(8) = '1'

    start_string = strtrim(string(long(start_run)), 2)
    FOR l = 1, 3 DO BEGIN
        IF (start_run(l) LE 9) THEN start_string(l) = '0'+start_string(l)
    ENDFOR

    line_command(9) = start_string(0)+''+start_string(1)+''+start_string(2)+'$
    ' '+start_string(3)+'0000'

    END_string = strtrim(string(long(end_run)), 2)
    FOR l = 1, 3 DO BEGIN
        IF (end_run(l) LE 9) THEN end_string(l) = '0'+end_string(l)
    ENDFOR

    line_command(10) = end_string(0)+end_string(1)+end_string(2)+'$
    ' '+end_string(3)+'0000'

    line_command(11) = strtrim(string(long(t_output*3600)), 2)
    line_command(12) = strtrim(string(long(t_average*3600)), 2)
    line_command(13) = strtrim(string(long(sample_rate*3600)), 2)
    line_command(14) = part_split
    line_command(15) = strtrim(string(long(sync_time*3600)), 2)
    line_command(16) = ctl

```

```

line_command(17) = ifine
line_command(18) = iout
line_command(19) = ipout
line_command(20) = LSUBGRID
line_command(21) = LCONVECTION
line_command(22) = LAGESPECTRA
line_command(23) = IPIN
line_command(24) = IFLUX
line_command(25) = MDOMAINFILL
line_command(26) = IND_SOURCE
line_command(27) = IND_RECEPTOR
line_command(28) = MQUASILAG
line_command(29) = NESTED_OUTPUT

openw, outfile, filepath+'options/CMD', /GET_LUN
FOR i = 1, 29 DO BEGIN
  printf, outfile, line_command(i)
ENDFOR
free_lun, outfile

;END write command file*****

;write pathnames file*****
line1 = filepath+'options/'
line2 = outpath
line3 = '/local/reh/rcowen/group_flex_idl/ECMWF'+index+'/'
line4 = '/local/reh/rcowen/group_flex_idl/ECMWF'+index+'/AVAILABLE'
line5 = '===== '
line6 = '

openw, outfile, filepath+'pathnames', /GET_LUN
printf, outfile, line1
printf, outfile, line2
printf, outfile, line3
printf, outfile, line4
printf, outfile, line5
printf, outfile, line6
free_lun, outfile

;END writing pathnames file

;write includepar

maxag = strtrim(string(long(n_elements(age_classes))), 2)
maxz = strtrim(string(long(n_elements(vlevels))), 2)
line0 = '          parameter(maxageclass='+maxag+',maxzgrid='+maxz+',nclassunc
=1)'
openw, outfile, '/local/reh/rcowen/group_flex_idl/modified/templ.txt', /ge
t_lun
printf, outfile, line0
free_lun, outfile

maxpart = maxpt*1.055
maxpart = strtrim(string(maxpart, format = '(I8)'), 2)
line1 = '          parameter(maxpart='+maxpart+')'

maxpointspec = 'maxspec'

line2 = '          parameter(maxpoint='+maxpoint+',maxspec='+maxspec+$
',maxpointspec='+maxpointspec+')'

```

```

    openw, outfile, '/local/reh/rcowen/group_flex_idl/modified/temp.txt', /get
_lun
    printf, outfile, line1
    printf, outfile, line2
    free_lun, outfile

    IF (min(ysr) LE 2005) THEN incpar_fil = 'includepar_top' ELSE incpar_fil =
'includepar_top0'

    rm_command = '\rm '+filepath+'includepar'
    cat_command = 'cat /local/reh/rcowen/group_flex_idl/modified/'+incpar_fil+
$
    ' /local/reh/rcowen/group_flex_idl/modified/templ.txt '+$
    '/local/reh/rcowen/group_flex_idl/modified/includepar_mid '+$
    '/local/reh/rcowen/group_flex_idl/modified/temp.txt '+$
    '/local/reh/rcowen/group_flex_idl/modified/includepar_bottom'+$
    ' > '+filepath+'includepar'

    spawn, rm_command
    spawn, cat_command

;END write includepar

;END writing files for modification

;link the necessary met files to the temp met directory and write the
;AVAILABLE file
    print, 'copying met files'
    met_dir = '/local/reh/rcowen/group_flex_idl/ECMWF'+index+'/'

    met_source = '/local/reh/ecmwf/'

    start_met = julday(start_run(1), start_run(2), start_run(0), start_run(3))
    end_met = julday(end_run(1), end_run(2), end_run(0), end_run(3))

    this_met = start_met-1.

    command_met1 = '\rm '+met_dir+'EN*'
    spawn, command_met1

    WHILE (this_met LE (end_met+1.)) DO BEGIN

        caldat, this_met, met_mon, met_day, met_yr, met_hr
;form is EN03032600 ENYYMMDDHH
        yr = strmid(strtrim(string(met_yr), 2), 2, 2)
        mo = strtrim(string(met_mon), 2)
        IF (mo LE 9) THEN mo = '0'+mo
        da = strtrim(string(met_day), 2)
        IF (da LE 9) THEN da = '0'+da
        hr = strtrim(string(met_hr), 2)
        IF (hr LE 9) THEN hr = '0'+hr

        met_source_file = met_source+'EN'+yr+mo+da+hr
        test_file_exist = findfile(met_source_file)
        IF (test_file_exist NE '') THEN BEGIN
            command_metloop = 'ln -s '+met_source+'EN'+yr+mo+da+hr+' '+met_dir+'.'
            spawn, command_metloop
        ENDIF ELSE print, 'missing '+met_source_file

        this_met = this_met+0.125

```

```

ENDWHILE

;write AVAILABLE file in the*****
met_dir = '/local/reh/rcowen/group_flex_idl/ECMWF'+index+'/'
write_available, met_dir

;write ageclasses file*****

openw, outfile, filepath+'options/AGECLASSES', /GET_LUN

printf, outfile, '*****'
printf, outfile, '*'
printf, outfile, '*Lagrangian particle dispersion model FLEXPART*'
printf, outfile, '*           Please select your options           *'
printf, outfile, '*'
printf, outfile, '*This file determines the ageclasses to be used*'
printf, outfile, '*'
printf, outfile, '*Ages are given in seconds. The first class *'
printf, outfile, '*starts at age zero and goes up to the first *'
printf, outfile, '*age specified. The last age gives the maximum *'
printf, outfile, '*time a particle is carried in the simulation. *'
printf, outfile, '*'
printf, outfile, '*****'

n_age = strtrim(string(long(n_elements(age_classes))), 2)
printf, outfile, n_age+'           Integer           Number of age classes'

age_classes_out = age_classes*60.*60.*24.
printf, outfile, format = '(I10)', age_classes_out

free_lun, outfile

IF (NOT keyword_set(do_not_run_flex)) THEN begin

    cd_command = 'cd '+filepath

;remove old FLEXPART executable so we dont run in case there is a
;compile problem
    rm_command = cd_command+'; \rm FLEXPART'
    spawn, rm_command

;compile and execute FLEXPART*****
    compile_command = 'gmake -C '+filepath
    spawn, compile_command

    execute_command = cd_command+' ; FLEXPART'
    spawn, execute_command
ENDIF

return
end

```


B.4 Programs for running and processing backward FLEX- PART simulations

This section contains the programs created to run backward FLEXPART simulations and process the ASCII output into idl files.

B.4.1 `convert_retro_temps_to_gridded_final.pro`



03/19/09

convert_retro_temps_to_gridded_final.p4

PRO convert_retro_temps_to_gridded_final, outdir, res_time = res_time

```

;+
;NAME: convert_retro_temps_to_gridded_final
;
;PURPOSE: This program reads the intermediate FLEXPART files and saves
;the final retroplume files to the working directory (where the
;intermediate files are also located).
;
;CATEGORY:
;
;FLEXPART program control
;
;CALLING SEQUENCE:
;
;convert_retro_temps_to_gridded_final, outdir, res_time = res_time
;
;INPUTS:
;
;savedir - the working directory (where the intermediate files are
;          located).
;
;KEYWORD PARAMETERS:
;
;res_time - set this keyword if the retroplumes are saving residence
;           time as opposed to the default specific volume weighted
;           residence time.
;
; OUTPUTS:
;
;
;
; EXAMPLES:
;
;-

;restore info from the header and releases file
  restore, file = outdir+'header.idlsav', /verbose
  restore, file = outdir+'releases.idlsav', /verbose

  x_grids = x_grid
  y_grids = y_grid

;grid info to save with each idlsav file
  x_info = [max(x_grid), min(x_grid), n_elements(x_grid), $
            ABS((max(x_grid)-min(x_grid))/(n_elements(x_grid)-1))]
  y_info = [max(y_grid), min(y_grid), n_elements(y_grid), $
            ABS((max(y_grid)-min(y_grid))/(n_elements(y_grid)-1))]

;find the temp idlsav files
  idlsaves = file_search(outdir+'*temp*.idlsav', count = n_idlsaves)

;get the names of just the files (no directories)
  filebreak, idlsaves, file = idlsave_names

;get a list of the the save times
  time_periods = STRMID(idlsave_names, 32, 10)
;get a list of the release names, but dimensioned the same as the file
;names to make it easier to pick the releases we want to read
  idlsave_names_short = STRMID(temporary(idlsave_names), 0, 26)
```

```

;loop over all the releases
  FOR loopn = 0, n_elements(rel_name)-1 DO BEGIN

    print, 'processing retro file ', rel_name(loopn)
    spawn, 'date'
;pick which files belong to the release we are sorting
    these_releases = where(idlsave_names_short EQ rel_name(loopn))
    these_time_periods = time_periods(these_releases)
    these_time_periods = these_time_periods[UNIQ(these_time_periods, $
                                                SORT(these_time_periods))]
;get julday for each output time to save with each array
    these_yrs = strmid(these_time_periods, 0, 4)
    these_months = strmid(these_time_periods, 4, 2)
    these_days = strmid(these_time_periods, 6, 2)
    these_hours = strmid(these_time_periods, 8, 2)
    these_min = strmid(these_time_periods, 10, 2)
    jd_grid = julday(temporary(these_months), temporary(these_days), $
                    temporary(these_yrs), temporary(these_hours), $
                    temporary(these_min))

;make array to hold all data, to be saved later by height, each height
;including all times
    all_res_times = make_array(n_elements(x_grids), n_elements(y_grids), $
                              n_elements(z_grid), n_elements(these_time_per
iods), $
                              /float)

;loop over all the files for this release, read them in, save the data
;to the all_res_time grid
    FOR loopt = 0, n_elements(these_releases)-1 DO BEGIN
;restore the file
        restore, file = idlsaves(these_releases(loopt));, /verbose

;get the time position (should match loopt, but just in case!)
        iot = where(these_time_periods EQ these_time_periods(loopt))

;loop over the elements in the save file and put the data in the
    FOR loopr = 0., (n_elements(lon_lat_height_time(0, *))-1) DO BEGIN
;get the x, y, z position FOR saving in all_res_times
        iox = where(x_grids EQ lon_lat_height_time(0, loopr))
        ioy = where(y_grids EQ lon_lat_height_time(1, loopr))
        ioz = where(z_grid EQ lon_lat_height_time(2, loopr))
;fill all_res_times array with the res time value
        all_res_times(iox, ioy, ioz, iot) = lon_lat_height_time(3, loopr)

    ENDFOR
  ENDFOR

;loop over each height level and save
    delta_lev = make_array(n_elements(z_grid))
    delta_lev(0) = z_grid(0)
    FOR loopzg = 1, n_elements(z_grid)-1 DO $
        delta_lev(loopzg) = z_grid(loopzg) - z_grid(loopzg-1)

    running_res_times = reform(all_res_times(*, *, 0, *)) * 0.0

    FOR looph = 0, n_elements(z_grid)-1 DO BEGIN

        lvl = strtrim(string(fix(z_grid(looph))), 2)
        IF (lvl LT 1000) THEN BEGIN
            lvl = '00'+lvl
        ENDIF ELSE BEGIN

```

```

        IF (lvl LT 10000) THEN BEGIN
            lvl = '0'+lvl
        ENDIF

    ENDELSE

    fil_name = outdir+rel_name(loopn)+'_lvl_'+lvl+'.idlsav'

    z_lev = z_grid(loopn)
    x_grid = x_info
    y_grid = y_info
    res_times = reform(all_res_times(*, *, loopn, *))

;adjust x_grid so that min is -180 and not -179. for std retroplumes
    IF ((x_grid(1) EQ (-179)) AND (x_grid(2) EQ 360)) THEN BEGIN
        temp_res_times = res_times(*, *, *)*0.0
        temp_res_times(1:359, *, *) = res_times(0:358, *, *)
        temp_res_times(0, *, *) = res_times(359, *, *)
        res_times = float(temporary(temp_res_times))
        x_grid(0:1) = x_grid(0:1)-1
    ENDIF

;adjust x_grid so that min is -181.25 and not -179.25. for GEOS
;retroplumes
    IF ((x_grid(1) EQ (-178.75)) AND (x_grid(2) EQ 144)) THEN BEGIN
        temp_res_times = res_times(*, *, *)*0.0
        temp_res_times(1:143, *, *) = res_times(0:142, *, *)
        temp_res_times(0, *, *) = res_times(143, *, *)
        res_times = float(temporary(temp_res_times))
        x_grid(0:1) = x_grid(0:1)-2.5
    ENDIF

    IF (keyword_set(res_time)) THEN $
        running_res_times = running_res_times+res_times $
    ELSE $
        running_res_times = running_res_times+res_times*delta_lev(loopn)

    save, res_times, x_grid, y_grid, z_lev, jd_grid, $
        file = fil_name, /compress

    ENDFOR

;save the total column res time for easier processing later
    res_times = temporary(running_res_times(*, *, *))
    IF (NOT keyword_set(res_time)) THEN $
        res_times = res_times/max(z_grid)
        z_lev = 0
        lvl = '00000'
        fil_name = outdir+rel_name(loopn)+'_lvl_'+lvl+'.idlsav'
        save, res_times, x_grid, y_grid, z_lev, jd_grid, $
            file = fil_name, /compress

    ENDFOR

    return
END

```

B.4.2 doit_dry_auto_retro.pro



03/19/09

doit_dry_auto_retro.pr

PRO doit_dry_auto_retro

```
;these need to be set everytime. the end_loop is 1 retroplume release
;time after the last retroplume (i.e., 00 UT will mean the last
;restroplume will be at 21 UT the day before
    start_loop = julday(8, 25, 2005, 00, 00)
    end_loop = julday(8, 26, 2005, 00, 00)

;delta days can be increased up to 5 days to decrease the overall run
;time to completion. a value of 2 means that 2 days of retroplumes
;will be run at time. PLEASE NOTE: the number of days (start_loop -
;end_loop) must be a multiple of delta_days, or extra days will be
;run. if met data does not exist for these extra days, the program
;will crash.
    delta_days = 1
;do not change for this program. 0 = not wet removal.
    wet_remov = 0
;default value of 1. longer means a longer retroplume release time
    duration = 1.
;default of 3. frequency of retroplume release, centered on (00 UT +
;start_offset)
    frequency = 6.
;offset value for retroplume initiation. if 0, the first retroplume
;will be at 00 UT, 3 will be 03 UT, etc.
    start_offset = 0.

    index = '#';makes sure we dont overwrite another directory
    n_days = end_loop-start_loop ;determine the total number of days to run
    FOR loopd = 0., (n_days-1), delta_days DO BEGIN ;loop over days, delta_day
s at a time

        this_date = start_loop+loopd ;determine start date for this loop
        this_date_plus = this_date+(24/24.*delta_days) ;determine end date for t
his loop

;find an empty directory to use for this loop, checking for the lock.file
        indexes = ['1', '2', '3', '4', '5', '6', '7', '8', '9']
        dir = '/local/reh/rcowen/retro_flex_idl_output/processing'
        loopi = 0
        while (loopi LT n_elements(indexes)) DO BEGIN
            check_lock = file_search(dir+indexes(loopi)+'lock.file')
            IF (check_lock EQ '') THEN BEGIN
                index = indexes(loopi)
                loopi = n_elements(indexes)
            ENDIF
            loopi = loopi+1
        ENDWHILE

        outdir = dir+index+ '/'
        print, index
;rm any leftover files from previous runs
        spawn, '\rm '+outdir+'*'

;create lock.file to prevent other runs from writing to this directory
        touch_command = 'touch '+outdir+'lock.file'
        spawn, touch_command

;compile and run FLEXPART
        run_retro_flex, this_date, this_date_plus, duration, frequency, index, $
            start_offset = start_offset, wet_remov = wet_remov, $
            max_age = max_age, $
            instant = instant, no_kernel = no_kernel
```

```
;process ascii FLEXPART output files
  read_retro_flex_output, outdir

;process intermediate files into final output files
  convert_retro_temps_to_gridded_final, outdir

;move any completed files to the final storage directory & remove old files
  command = '\mv '+outdir+'*lvl* '+$
            '/local/reh/rcowen/retro_flex_idl_output/pico_std_dry/.'
  spawn, command
  spawn, '\rm '+outdir+'*0000*temp*'
  spawn, '\rm '+outdir+'*0300*temp*'
  spawn, '\rm '+outdir+'*0600*temp*'
  spawn, '\rm '+outdir+'*0900*temp*'
  spawn, '\rm '+outdir+'*1200*temp*'
  spawn, '\rm '+outdir+'*1500*temp*'
  spawn, '\rm '+outdir+'*1800*temp*'
  spawn, '\rm '+outdir+'*2100*temp*'
  spawn, '\rm '+outdir+'*'

  enDFOR

;update the tracker file that records all available retroplume
;times. this adds time to this stage (creating the final files), but
;decreases processing time for the end user (using the retroplumes).
  make_retro_filelist, wet_remov = wet_remov

  return
END
```

B.4.3 doit_wet_auto_retro.pro



03/19/09

doit_wet_auto_retro.pr

PRO doit_wet_auto_retro

```
;these need to be set everytime. the end_loop is 1 retroplume release
;time after the last retroplume (i.e., 00 UT will mean the last
;restroplume will be at 21 UT the day before
  start_loop = julday(8, 25, 2005, 00, 00)
  end_loop = julday(8, 26, 2005, 00, 00)

;delta days can be increased up to 5 days to decrease the overall run
;time to completion. a value of 2 means that 2 days of retroplumes
;will be run at time. PLEASE NOTE: the number of days (start_loop -
;end_loop) must be a multiple of delta_days, or extra days will be
;run. if met data does not exist for these extra days, the program
;will crash.
  delta_days = 1
;do not change for this program. 0 = not wet removal.
  wet_remov = 1
;default value of 1. longer means a longer retroplume release time
  duration = 1.
;default of 3. frequency of retroplume release, centered on (00 UT +
;start_offset)
  frequency = 3.
;offset value for retroplume initiation. if 0, the first retroplume
;will be at 00 UT, 3 will be 03 UT, etc.
  start_offset = 0.

  index = '#';makes sure we dont overwrite another directory
  n_days = end_loop-start_loop ;determine the total number of days to run
  FOR loopd = 0., (n_days-1), delta_days DO BEGIN ;loop over days, delta_day
s at a time

    this_date = start_loop+loopd ;determine start date for this loop
    this_date_plus = this_date+(24/24.*delta_days) ;determine end date for t
his loop

;find an empty directory to use for this loop, checking for the lock.file
    indexes = ['1', '2', '3', '4', '5', '6', '7', '8', '9']
    dir = '/local/reh/rcowen/retro_flex_idl_output/processing'
    loopi = 0
    while (loopi LT n_elements(indexes)) DO BEGIN
      check_lock = file_search(dir+indexes(loopi)+'lock.file')
      IF (check_lock EQ '') THEN BEGIN
        index = indexes(loopi)
        loopi = n_elements(indexes)
      ENDIF
      loopi = loopi+1
    ENDWHILE

    outdir = dir+index+'/'
    print, index
;rm any leftover files from previous runs
    spawn, '\rm '+outdir+'*'

;create lock.file to prevent other runs from writing to this directory
    touch_command = 'touch '+outdir+'lock.file'
    spawn, touch_command

;compile and run FLEXPART
    run_retro_flex, this_date, this_date_plus, duration, frequency, index, $
      start_offset = start_offset, wet_remov = wet_remov, $
      max_age = max_age, $
      instant = instant, no_kernel = no_kernel
```

```
;process ascii FLEXPART output files
  read_retro_flex_output, outdir

;process intermediate files into final output files
  convert_retro_temps_to_gridded_final, outdir

;move any completed files to the final storage directory & remove old files
  command = '\mv '+outdir+'*lvl* '+$
            '/local/reh/rcowen/retro_flex_idl_output/pico_std_dry/.'
  spawn, command
  spawn, '\rm '+outdir+'*0000*temp*'
  spawn, '\rm '+outdir+'*0300*temp*'
  spawn, '\rm '+outdir+'*0600*temp*'
  spawn, '\rm '+outdir+'*0900*temp*'
  spawn, '\rm '+outdir+'*1200*temp*'
  spawn, '\rm '+outdir+'*1500*temp*'
  spawn, '\rm '+outdir+'*1800*temp*'
  spawn, '\rm '+outdir+'*2100*temp*'
  spawn, '\rm '+outdir+'*'

  enDFOR

;update the tracker file that records all available retroplume
;times. this adds time to this stage (creating the final files), but
;decreases processing time for the end user (using the retroplumes).
  make_retro_filelist, wet_remov = wet_remov

  return
END
```

B.4.4 read_retro_flex_output.pro



```
PRO read_retro_flex_output, outdir
```

```

;+
;NAME: read_retro_flex_output
;
;PURPOSE:
;This program reads formatted (ascii) FLEXPART output, as designed by
;RCO for flexpart model runs @ MTU, and creates temporary idlsav files
;(to be processed by convert_retro_temps_to_gridded_final).
;
;
;CATEGORY:
;
;FLEXPART program control
;
;CALLING SEQUENCE:
;
;read_retro_flex_output, outdir
;
;INPUTS:
;
;outdir - directory containing the ascii FLEXPART output files.
;
; KEYWORD PARAMETERS:
;
;none
;
; OUTPUTS:
;
;
;
; EXAMPLES:
;
;-

```

```

junk = ''

;read settings from header file
openr, infile, outdir+'header.txt', /get_lun

startline = ''
readf, infile, startline
readf, infile, format = '(3A10)', loutstep,loutaver,loutsample
readf, infile, format = '(6A8)', outlon0, outlat0, $
    numxgrid, numygrid, dxout, dyout
heightline = ''
readf, infile, heightline

readf, infile, timeline
readf, infile, format = '(I15)', n_spec_x_3
spec_arr = make_array(n_spec_x_3/3,/string)
numzgrid = 0
FOR loops = 0, (n_spec_x_3/3)-1 DO BEGIN
    readf, infile, junk
    readf, infile, junk
    readf, infile, junk
    spec_arr(loops) = strtrim(strmid(string(junk), 5, 20), 2)
ENDFOR

readf, infile, junk
ageline = ''
readf, infile, ageline

```

```

;close header file

free_lun, infile

;extract dimensions of outgrid
x_grid = ((indgen(numxgrid)*dxout)+outlon0)
y_grid = ((indgen(numygrid)*dyout)+outlat0)

;extract height levels from heightline
nheights = fix(strtrim(strmid(heightline, 0, 9), 2))
z_grid = make_array(nheights)

FOR looph = 0, nheights-1 DO BEGIN
    z_grid(looph) = fix(strtrim(strmid(heightline, (10+10*looph), 10), 2))
ENDFOR

;extract start date from startline
start_date = strmid(strtrim(startline, 2), 0, 8)
start_sim = [fix(strmid(start_date, 0, 4)), $
             fix(strmid(start_date, 4, 2)), $
             fix(strmid(start_date, 6, 2))]

;extract age classes from ageline, ages are in seconds
nage = fix(strtrim(strmid(ageline, 0, 10), 2))

age_grid = make_array(nage)
FOR loopa = 0, nage-1 DO BEGIN
    age_grid(loopa) = (strtrim(strmid(ageline, (10+10*loopa), 10), 2))
ENDFOR

;save the header info into an idlsav file
save, x_grid, y_grid, z_grid, age_grid, $
    file = outdir+'header.idlsav', /compress

;open releases file
openr, infile, outdir+'releases.txt', /get_lun
readf, infile, n_releases

rel_times = make_array(3, n_releases)
rtt = indgen(n_releases)+1
rel_times(2, *) = rtt
rel_grid = make_array(6, n_releases)
rel_part = make_array(n_releases)
rel_name = make_array(n_releases, /string)

FOR loopr = 0, n_releases-1 DO BEGIN
    readf, infile, format = '(3I10)', start_rel, end_rel, kindz
    rel_times(0, loopr) = start_rel
    rel_times(1, loopr) = end_rel
    readf, infile, format = '(6F10.3)', xp1,yp1,xp2,yp2,zpoint1,zpoint2
    rel_grid(0, loopr) = xp1
    rel_grid(0, loopr) = yp1
    rel_grid(0, loopr) = xp2
    rel_grid(0, loopr) = yp2
    rel_grid(0, loopr) = zpoint1
    rel_grid(0, loopr) = zpoint2
    readf, infile, format = '(2I10)', npart,one
    rel_part(loopr) = npart
    compoint = ''
    readf, infile, format = '(A40)', compoint
    rel_name(loopr) = strtrim(compoint, 2)
    FOR loopk = 0, 2 DO readf, infile, junk

```

```

ENDFOR

free_lun, infile
;save info from the releases file
save, rel_times, rel_grid, rel_name, rel_part, $
    file = outdir+'releases.idlsav', /compress

;get file list of files in output dir
time_files = findfile(outdir+'grid_time_*', count = n_time)
filebreak, time_files, name = time_names

;array to assist in naming idlsav files
time_string_names = strmid(time_names(*), 10, 10)

FOR loopp = 0, n_time-1 DO BEGIN
    print, 'processing file #' + strtrim(string(loopp+1), 2) + ' out of ' + $
        strtrim(string(n_time), 2)
    spawn, 'date'
    jd = julday(strmid(time_names(loopp), 14, 2), $
        strmid(time_names(loopp), 16, 2), $
        strmid(time_names(loopp), 10, 4), $
        strmid(time_names(loopp), 18, 2))

;determine the number of data lines to make the array
    openr, infile, time_files(loopp), /get_lun
    n_lines = 0.
    WHILE (NOT(EOF(infile))) DO BEGIN
        readf, infile, junk
        IF ((junk NE ' 1 sparse') and $
            (junk NE '-999 999.0 999.0')) THEN $
            n_lines = n_lines+1
    ENDWHILE
    free_lun, infile

;make array to hold the data until it's written to a temp idlsav file
;array_format = ['jd', 'lon', 'lat', 'height', 'rel_num', 'res_time']
gridded_flex_time = make_array(6, n_lines, /double)

    openr, infile, time_files(loopp), /get_lun
    readf, infile, junk
    loopf = 0.
    WHILE (NOT(EOF(infile))) DO BEGIN
        readf, infile, junk

        IF ((junk NE ' 1 sparse') and $
            (junk NE '-999 999.0 999.0')) THEN BEGIN
;            write(unitoutgrid, '(5I5,I10,2F20.10)')
;            ix, jy, kz, k, nage,

            x_cell = fix(strmid(junk, 0, 5))
            y_cell = fix(strmid(junk, 5, 5))
            z_lev = fix(strmid(junk, 10, 5))
            rel_n = fix(strmid(junk, 15, 5))
            r_time = float(strmid(junk, 35, 15))
            time_line = [jd, x_grid(x_cell), y_grid(y_cell), $
                z_grid(z_lev-1), rel_n, r_time]
            gridded_flex_time(*, loopf) = time_line
            loopf = loopf+1.
        ENDIF
    ENDWHILE

```

```

free_lun, infile

;pick the data that is associated with each release and write a
;temporary idlsav file, to be combined with other idlsav files

FOR loopr = 1, n_releases DO BEGIN
    this_release = where(gridded_flex_time(4, *) EQ loopr)
    IF (this_release(0) NE (-1)) THEN BEGIN

        lon_lat_height_time = make_array(4, n_elements(this_release))
        lon_lat_height_time(0, *) = gridded_flex_time(1, this_release)
        lon_lat_height_time(1, *) = gridded_flex_time(2, this_release)
        lon_lat_height_time(2, *) = gridded_flex_time(3, this_release)
        lon_lat_height_time(3, *) = gridded_flex_time(5, this_release)
        curr_time = gridded_flex_time(0, this_release(0))
        this_time_name = time_string_names(loopr)
        this_rel_name = rel_name(loopr-1)
        this_temp_name = this_rel_name+'_temp_'+this_time_name+'.idlsav'

        save, lon_lat_height_time, curr_time, $
            file = outdir+this_temp_name

    ENDIF
ENDFOR

ENDFOR

return
END

```

B.4.5 run_retro_flex.pro



03/19/09

run_retro_flex.pr

```
PRO run_retro_flex, starttr, endr, rel_duration, rel_frequency, index, $
    start_offset = start_offset, wet_remov = wet_remov, $
    max_age = max_age, $
    instant = instant, no_kernel = no_kernel

;+
;NAME: run_retro_flex
;
;PURPOSE:
;
;This program writes the files required to run FLEXPART and then
;compiles and runs FLEXPART. The program assumes a release centered
;around the Pico Mountain observatory. This program has reasonable
;defaults for most of the settings required (e.g., the CTL time step)
;and focuses on the few options that will need to be changed
;frequently between simulations (e.g., wet or dry removal).
;
;CATEGORY:
;
;FLEXPART program control
;
;CALLING SEQUENCE:
;
;run_retro_flex, starttr, endr, rel_duration, rel_frequency, index, $
;                start_offset = start_offset, wet_remov = wet_remov, $
;                max_age = max_age, $
;                instant = instant, no_kernel = no_kernel
;
;INPUTS:
;
;starttr - start date of the simulation in julian days. Make sure this
;          date includes model spinup, i.e., the difference between the
;          start date and the day of desired output is equal to or
;          greater than the maximum age of tracer in the program,
;          typically 20 days.
;
;endr - end date of the simulation in julian days.
;
;rel_duration - the length of time for the retroplume
;               release, typically 1-3 hours. The release will be
;               centered on the release time.
;
;rel_frequency - the frequency of retroplume release. typical times
;               are 3 and 6 hours.
;
;index - the number of the directory to save output
;        to. /local/reh/rcowen/retro_flex_idl_output#, where # is
;        currently 1, 3, 4, 6, 7, 9, and 10.
;
; KEYWORD PARAMETERS:
;
;start_offset - determines the start hour of the first retroplume
;               release. the default is 00 UT.
;
;wet_remov - if set, the retroplume will use the wet removal rate of
;            NO2. Otherwise, not removal is used.
;
;max_age - if set, this keyword will over ride the maximum age of
;           retroplume (20 days). units are days, decimals are allowed.
;
;instant - if set, instantaneous output is given. otherwise, averaged
;          output is given.
;
```

```

;no_kernel - if set, the kernel, which distributes the mass of each
;              particle across 4 grid cells, will not be used.
;
; OUTPUTS:
;
;
;
; EXAMPLES:
;
;-

;  startr AND endr should be in julday, convert them to
;  stop
caldat, startr, mos, das, yrs, hrs
caldat, endr, moe, dae, yre, hre
ndyz = endr-startr+0.125
;make sure dates fall within the dates set in the COMMAND file
back_start_date = [yrs, mos, das, hrs] ;[YYYY,MM,DD,HH]
back_end_date = [yre, moe, dae, hre] ;[YYYY,MM,DD,HH]
use_pico_backward = 1 ;0 = no, 1 = yes

n_ages = 1.
IF (keyword_set(max_age)) THEN age_classes = [max_age] $
ELSE age_classes = [20.]

;direction, 1 = forward, -1 = backwards
direction = (-1) ;,(-1)

;start date of simulation (must be before the first release in the
;RELEASE file)
startb = startr-age_classes(0)-0.25
endb = endr+0.25

caldat, startb, mosb, dasb, yrsb, hrsb
caldat, endb, moeb, daeb, yreb, hreb

start_run = [yrsb, mosb, dasb, hrsb] ;[YYYY,MM,DD,HH]
end_run = [yreb, moeb, daeb, hreb] ;[YYYY,MM,DD,HH]

;see below for explanations of these options. they are placed here
;because they are also commonly modified options
;  ifine = '1'
;  ctl = '1'
ifine = '10'
ctl = '10'

t_output = 6

;[source = 1,  recept = 2] => ppb when folded w/ emissions
IND_SOURCE = '1' ;1=mass units (bwd=concentration), 2=mass mi
xing ratio units
IND_RECEPTOR = '2' ;1=mass units (for bwd-runs = concentration)
, 2=mass mixing ratio units

;the interval is length of each release. if the interval is 3 hours
;and the start date is 2001, 07, 10, 00, then the first retroplume
;will be released from 2001, 07, 10, 00 to 2001, 07, 10, 03.
back_interval = rel_frequency

```

```

;the box is the "default" box, but can be changed if some reason is
;identified (e.g., using zt from Jan's analysis as the center height).
  back_box = [-29., -28., 38., 39., 2000, 2500]
  back_num_particles = 4000      ;recommend at least 4000 part/hr of release
  maxpart = strtrim(string(ULONG(ndyz*back_num_particles*rel_duration*(24/rele
l_frequency))), 2)
;stop
;END options for the RELEASES file

;Options for the OUTGRID file. OUTGRID defines the grid for which the
;model outputs data for. this grid must be less (smaller) than the
;input met grid. the default ECMWF grid is -179 to 180 long, -90 to 90
;lat.

;OUTLONLEFT is the GEOGRAPHICAL LONGITUDE OF LOWER LEFT CORNER OF
;OUTPUT GRID.
  OUTLONLEFT = (-179)
  OUTLATLOWER = (0)

;NUMXGRID is the NUMBER OF GRID POINTS IN X DIRECTION (longitude).
  NUMXGRID = (360)
  NUMYGRID = (90)

;DXOUTLON is the delta in the x direction (longitude). if
;outlonleft = -179, numxgrid = 360, and DXOUTLON = 1.0, then the
;longitudinal grid covers the globe from -179 to 180
  DXOUTLON = 1.0
  DYOUTLAT = 1.0

;vlevels are the upper limites for each vertical level for output. the
;levels are the UPPER limit for each level. the first level goes from
;the surface up to vlevels(0) meter, the second level goes from
;vlevels(0) to vlevels(1), etc. note, levels are in METERS
  vlevels = [300, 1000, 1500, 2000, 2500, 3000, 4000, 5000, 7500, 10000, 15000]

;END options for OUTGRID

;frequency of output. this is an important parameter and should be
;chosen carefully. the recommended value is either 3 hrs or 6 hrs. If
;this is too
;large, you will get a very course response (think trajectory end
;points output only every 2 days). this parameter should match the
;next parameter for best results. it should also be noted that
;doubling this number will double the number of output files (and
;output size). time is in hours below.

;t_output = 6

;averaging time of output. should match above parameter and is set
;that way by default. i do not know reasons why you would set this
;differently.

  t_average = t_output

;sample rate of output. this parameter is how often the model data is

```

```

;sampled to produce the average. if the data is sampled @ 0.25 hrs
;with a 3 hr avg time (above), then you have 12 "data points" in the
;averaged result. therefor, the sample rate should be some small
;fraction of the averaging time (~1/10th). for a 3 hrs average time, a
;15 min (0.25) sample rate is recommended

```

```

sample_rate = 0.25

```

```

;time constant for splitting particles. this should not be changed.

```

```

part_split = '999999'

```

```

;SYNCHRONISATION INTERVAL OF FLEXPART. All processes are synchronized
;with this time interval, all other time constants must be multiples
;of this value. Output interval and time average of output must be at
;least twice this value. Generally, this is set to the sample rate
;(above), as thats usually the smallest time constant.

```

```

sync_time = 0.25

```

```

;CTL (see FLEXPART documentation for more information). A negative
;value produces the fastest model run, but mostly ignores
;turbulence. A larger, positive value describes turbulence better, but
;takes longer to run. Negative value is probably OK for US anthro
;forward sim, but should be positive for more specific applications
;(e.g., retroplumes, modelling individual fires).

```

```

;ctl = '1'

```

```

;IFINE. The reduction factor for time step used for vertical
;wind. having this be a larger number will speed computational times
;with large values of ctl (above), but still give a good
;representation of turbulence.

```

```

;ifine = '6'

```

```

;IOUT, determines how the output shall be made: concentration (ng/m3,
;Bq/m3), mixing ratio (pptv), or both, or plume trajectory mode, or
;concentration + plume trajectory mode. In plume trajectory mode,
;output is in the form of average trajectories. 1 CONC. (RESID. TIME
;FOR BACKWARD RUNS) OUTPUT,2 MIX. RATIO OUTPUT,3 BOTH,4 PLUME
;TRAJECT.,5=1+4. for backward runs, 5 is highly recommended.

```

```

iout = '1'

```

```

;IPOINT, determines whether particle positions are outputted (in
;addition to the gridded concentrations or mixing ratios) or not. 0=no
;output, 1 output every output interval, 2 only at end of the
;simulation. This is an advanced setting. For backward runs, this
;should be 0 as backward runs are short and redone quickly if the
;program crashes. If you are doing longer runs, this should be 1, if
;you are doing continued runs (forward simulations that are longer
;than ~1 month), this should be either 1 or 2 (2 recommended for space
;considerations).

```

```

ipout = '0'

```

```

;LSUBGRID, LCONVECTION, these parameters turn on and off the use of
;topography (LSUBGRID) and the convection scheme
;(LCONVECTION). generally speaking, these should always be ON (value
;of 1), though turning them off will speed up a model run. LSUBGRID
;may not be as important for backward runs from PICO.

```

```

LSUBGRID = '1' ;(1 = on, 0 = off)

```

```

LCONVECTION = '1' ;(1 = on, 0 = off)

;LAGESPECTRA, this turns on and off the calculation of age classes. if
;this is turned off, the file options/AGECLASSES must be available and
;updated. this is normally turned on for forward runs, though not
;exclusively. if it is off, particles will be carried in the model
;until the model run ends. options for AGECLASSES are below.
LAGESPECTRA = '1' ;(1 = on, 0 = off)

;IPIN, this continues a model run with dumped particle locations. this
;is an advanced option and would be used with ipout (above). this is
;generally turned off.
IPIN = '0' ;(1 = on, 0 = off)

;IFLUX, this calculates fluxes in each grid cell. this is normally
;turned off.
IFLUX = '0' ;(1 = on, 0 = off)

;MDOMAINFILL, this turns on and off the calculation of domain filling
;trajectories. this is normally turned off.
MDOMAINFILL = '0' ;(1 = on, 0 = off)

;IND_SOURCE, switches between different units for concentrations at
;the source. NOTE that in backward simulations the release of
;computational particles takes place at the "receptor" and the
;sampling of particles at the "source". this is generally set to 1
;(mass units).
;IND_SOURCE = '1' ;1=mass units (bwd=concentration), 2=mass mixing ratio units

;IND_RECEPTOR, switches between different units for concentrations at
;the receptor. this is normally not used because we don't calculate
;receptor concentrations because they are calculated at the surface
;and PICO isn't in the topography.
;IND_RECEPTOR = '2' ;1=mass units (for bwd-runs = concentration), 2=mass mixing ratio units

;MQUASILAG, indicates whether particles shall be numbered
;consecutively (1) or with their release location number (0). The
;first option allows tracking of individual particles using the
;partposit output files. this is an advanced setting and is normally
;not on.
MQUASILAG = '0' ;(1 = on, 0 = off)

;NESTED_OUTPUT decides whether model output shall be made also for a
;nested output field (normally with higher resolution). this is an
;advanced setting and is never used, unless we have additional met
;data.
NESTED_OUTPUT = '0' ;(1 = on, 0 = off)

;END of options for the COMMAND file

;options for the includepar files in the main program directory.
;includepar has 4 variables that should be modified for each
;run. these are maxpart, maxpoint, maxspec, maxpointspec (lines
;169 & 170).

;maxpart = '1000000' ; maximum number of particles allowed. releases
; will automatically be scaled to this value

maxpoint = '60' ; number of releases. for backward runs, max

```

03/19/09

run_retro_flex.pr

```
x is around          ; 60 for some reason. for forward runs, need ~3000.

maxspec = '1'          ; max number of chemical species in the rele
ase file.              ; typically only 1, but max of 3.

;END of options for includepar

;options for pathnames. please select a location for your
;ouput. WARNING: if a new folder is not selected for output, then any
;previous results WILL be over written :WARNING.

  outpath = '/local/reh/rcowen/retro_flex_idl_output/processing'+index+'/'

;if you are using your own folder to compile and run this program,
;then you will need to specify the location of this folder
;here. otherwise, you are running this in the group directory.
  filepath = '/local/reh/rcowen/group_flex_idl/REH_group_FLEXPART'+index+'/'

;*****
;*****
;*****
;END OF OPTIONS. The rest of the program uses the options set above to
;compile and run FLEXPART. Nothing below this point should be
;modified. Plotting procedures are elsewhere.
;*****
;*****
;*****

;copy all the flexpart program files to the specified directory

  source_dir = '/local/reh/rcowen/group_flex_idl/FLEXPART_source/'

;first copy all the "default" files (e.g., those that do not need to
;be modified). this will actually copy over ALL files, then rewrite
;the ones that need to be modified.

  command1 = '\cp -r '+source_dir+'* '+filepath+'.'

  spawn,  command1

;copy the non-run specific files that have been modified to the
;program directory. these are calcmatrix.f, writeheader.f,
;conccoutput.f, and makefile

  mod_dir = '/local/reh/rcowen/group_flex_idl/modified/'
  MOD_list = ['calcmatrix.f', 'writeheader.f', 'conccoutput.f', 'readwind.f']
  IF (keyword_set(instant)) THEN print, 'INSTANT'
  IF (keyword_set(instant)) THEN MOD_list = [MOD_list, 'timemanager_instant.
f']
  FOR MOD_loop = 0, (n_elements(MOD_list)-1) DO BEGIN
    command_mod = '\cp '+MOD_dir+MOD_list(MOD_loop)+' '+filepath
    spawn, command_mod
  ENDFOR
  IF (keyword_set(no_kernel)) THEN print, 'NO KERNEL'
  IF (keyword_set(no_kernel)) THEN $
    spawn, '\cp '+MOD_dir+'conccalc_no_kernel.f'+ ' '+filepath+$
```

```

spawn, '\cp '+MOD_dir+'conccalc_no_kernel.f'+ ' '+filepath+'
'conccalc.f'

;copy the right makefile for the system to the program path
makef = 'makefile_L64_RHEL5'

command_make_mv = '\cp '+MOD_dir+makef+' '+filepath+'makefile'
spawn, command_make_mv

;Write all the run-specific files, as dictated by the options set
;above.

;write releases file

  IF (keyword_set(start_offset)) THEN delta_start = double(start_offset) ELSE
E delta_start = 0.0
  line_releases = make_array(14, 2, /string)

  line_releases(0, 0) = '*****'
  line_releases(1, 0) = '*'
  line_releases(2, 0) = '*'
  line_releases(3, 0) = '*'
  line_releases(4, 0) = '*' Input file for the Lagrangian particle dispersi
on model FLEXPART
  line_releases(5, 0) = '*' Please select your options
  line_releases(6, 0) = '*'
  line_releases(7, 0) = '*'
  line_releases(8, 0) = '*'
  line_releases(9, 0) = '*****'
  line_releases(10, 0) = '+++++'
  openw, outfile, filepath+'options/RELEASES', /GET_LUN

  FOR i = 0, 10 DO BEGIN
    printf, outfile, line_releases(i, 0)
  ENDFOR

  IF ((direction EQ (-1)) AND (use_pico_backward EQ 1)) THEN BEGIN
;backward run
    line_releases(11, 0) = '1 Total number of speci
es emitted'

    IF (keyword_set(wet_remov)) THEN $
      line_releases(12, 0) = '32 Index of species in
file SPECIES' $
    ELSE $
      line_releases(12, 0) = '24 Index of species in
file SPECIES'
    line_releases(13, 0) = '=====
=====

  FOR i = 11, 13 DO printf, outfile, line_releases(i, 0)

```

```

;determine the number of releases
    jd_start_back = julday(back_start_date(1), back_start_date(2), $
                          back_start_date(0), back_start_date(3))
    jd_end_back = julday(back_end_date(1), back_end_date(2), $
                        back_end_date(0), back_end_date(3))
    n_days = jd_end_back - jd_start_back
    n_periods = (n_days*24)/back_interval

;loop over each release and write to file
    FOR i = 0., (n_periods-1) DO BEGIN
        this_period = make_array(5, 2)

;calculate the start and end of each release and change format for
;printing to file
        this_jd_period = jd_start_back + (i)*(back_interval/24.) - $
                        double(rel_duration/2./24.) + (delta_start/24)

        this_jd_period_end = this_jd_period + double(rel_duration/24.)

        caldat, this_jd_period, a, b, c, d, e
        this_period(1, 0) = a
        this_period(2, 0) = b
        this_period(0, 0) = c
        this_period(3, 0) = d

;force time period to end on either 00 or 30 min
        check_30 = e MOD 30.
        IF (((check_30 GE 15) AND (e LE 30)) OR $
            ((check_30 LE 15) AND (e GE 30))) $
            THEN e = 30 ELSE e = 0
        this_period(4, 0) = e

        caldat, this_jd_period_end, a, b, c, d, e
        this_period(1, 1) = a
        this_period(2, 1) = b
        this_period(0, 1) = c
        this_period(3, 1) = d
;force time period to end on either 00 or 30 min
        check_30 = e MOD 30.
        IF (((check_30 GE 15) AND (e LE 30)) OR $
            ((check_30 LE 15) AND (e GE 30))) $
            THEN e = 30 ELSE e = 0
        this_period(4, 1) = e

        this_period = strtrim(string(long(this_period(*))), 2)

        FOR j = 0, 9 DO IF (this_period(j) LE 9) THEN this_period(j) = '0'+this_
s_period(j)

        printf, outfile, this_period(0)+this_period(1)+$
            this_period(2)+' '+this_period(3)+this_period(4)+'00 '

        printf, outfile, this_period(5)+this_period(6)+$
            this_period(7)+' '+this_period(8)+this_period(9)+'00 '

        printf, outfile, format = '(f9.4)', back_box(0)
        printf, outfile, format = '(f9.4)', back_box(2)
        printf, outfile, format = '(f9.4)', back_box(1)
        printf, outfile, format = '(f9.4)', back_box(3)

        printf, outfile, format = '(i9)', 2

```



```

printf, outfile, format = '(f10.3)', back_box(4)
printf, outfile, format = '(f10.3)', back_box(5)

printf, outfile, format = '(i9)', back_num_particles

printf, outfile, '    1.00000'

                                ;format='(a40)'

this_period_name = (this_jd_period+this_jd_period_end)/2.
caldat, this_period_name, a, b, c, d, e
this_period_n = make_array(5)
this_period_n(1) = a
this_period_n(2) = b
this_period_n(0) = c
this_period_n(3) = d
check_30 = e MOD 30.
IF ((check_30 GE 15) AND (e LE 30)) OR $
    ((check_30 LE 15) AND (e GE 30)) $
    THEN e = 30 ELSE e = 0
this_period_n(4) = e
this_period_n = strtrim(string(long(this_period_n(*))), 2)
FOR j = 0, 4 DO IF (this_period_n(j) LE 9) THEN this_period_n(j) = '0'
+this_period_n(j)
printf, outfile, 'pico_retro_'+this_period_n(0)+'_'+this_period_n(1)+$
    '_' +this_period_n(2)+'_'+this_period_n(3)+this_period_n(4)

printf, outfile, '++++++'
++++++

ENDFOR

free_lun, outfile

ENDIF

;write OUTGRID file

line_outgrid = make_array(32, 2, /string)

line_outgrid(0, 0) = '*****'
*****
line_outgrid(2, 0) = '*'
*
line_outgrid(3, 0) = '*'      Input file for the Lagrangian particle disper
sion model FLEXPART
line_outgrid(4, 0) = '*'      Please specify your output g
rid
line_outgrid(5, 0) = '*'
*
line_outgrid(6, 0) = '*****'
*****
line_outgrid(7, 0) = ''
line_outgrid(8, 0) = '1.  -----.----      4X,F11.4'
line_outgrid(9, 0) = ''
line_outgrid(9, 1) = '(A4,F11.4)'
line_outgrid(10, 0) = '      OUTLONLEFT      (left boundary of the first g
rid cell - not its centre, 0)'
line_outgrid(11, 0) = ''

```

```

line_outgrid(12, 0) = '2.  -----.---- 4X,F11.4'
line_outgrid(13, 0) = ' '
line_outgrid(13, 1) = '(A4,F11.4)'
line_outgrid(14, 0) = '      OUTLATLOWER (lower boundary of the first
grid cell - not its centre, 0)'
line_outgrid(15, 0) = ' '
line_outgrid(16, 0) = '3.  ----- 4X,I5'
line_outgrid(17, 0) = ' '
line_outgrid(17, 1) = '(A4,I5)'
line_outgrid(18, 0) = '      NUMXGRID'
line_outgrid(19, 0) = ' '
line_outgrid(20, 0) = '4.  ----- 4X,I5'
line_outgrid(21, 0) = ' '
line_outgrid(21, 1) = '(A4,I5)'
line_outgrid(22, 0) = '      NUMYGRID'
line_outgrid(23, 0) = ' '
line_outgrid(24, 0) = '5.  -----.---- 4X,F10.3'
line_outgrid(25, 0) = ' '
line_outgrid(25, 1) = '(A4,F10.3)'
line_outgrid(26, 0) = '      DXOUTLON'
line_outgrid(27, 0) = ' '
line_outgrid(28, 0) = '6.  -----.---- 4X,F10.3'
line_outgrid(29, 0) = ' '
line_outgrid(29, 1) = '(A4,F10.3)'
line_outgrid(30, 0) = '      DYOUTLAT'
line_outgrid(31, 0) = ' '

outgrid_options = [OUTLONLEFT, OUTLATLOWER, NUMXGRID, NUMYGRID, DXOUTLON,
DYOUTLAT]

openw, outfile, filepath+'options/OUTGRID', /GET_LUN
j_out = 0
FOR i = 1, 31 DO BEGIN
  IF (line_outgrid(i, 1) EQ '') THEN BEGIN
    printf, outfile, line_outgrid(i, 0)
  ENDEF ELSE BEGIN
    printf, outfile, format = line_outgrid(i, 1), line_outgrid(i, 0), outg
rid_options(j_out)
    j_out = j_out+1
  ENDEF ELSE
  ENDFOR

FOR i = 0, (n_elements(vlevels)-1) DO BEGIN
  printf, outfile, 'XX. -----.---- 4X, F7.1'
  printf, outfile, format = '(A4,F7.1)', '', vlevels(i)
  printf, outfile, '      LEVEL X      HEIGHT OF LEVEL (UPPER BOUNDARY)
,
  printf, outfile, ''
ENDEFOR

printf, outfile, ''
free_lun, outfile

;END print OUTGRID

;write COMMAND file

line_command = make_array(31, /string)

line_command(1) = '*****

```

03/19/09

run_retro_flex.pr

```
*****
line_command(2) = '*'
line_command(3) = '*'      Input file for the Lagrangian particle dispersio
n model FLEXPART          *
line_command(4) = '*'      Please select your options
line_command(5) = '*'
line_command(6) = '*****'
*****
line_command(7) = ''
line_command(8) = substrim(string(long(direction)), 2)

start_string = substrim(string(long(start_run)), 2)
FOR l = 1, 3 DO BEGIN
  IF (start_run(l) LE 9) THEN start_string(l) = '0'+start_string(l)
ENDFOR

line_command(9) = start_string(0)+''+start_string(1)+''+start_string(2)+$
  ''+start_string(3)+'0000'

END_string = substrim(string(long(end_run)), 2)
FOR l = 1, 3 DO BEGIN
  IF (end_run(l) LE 9) THEN end_string(l) = '0'+end_string(l)
ENDFOR

line_command(10) = end_string(0)+end_string(1)+end_string(2)+$
  ''+end_string(3)+'0000'

line_command(11) = substrim(string(long(t_output*3600)), 2)
line_command(12) = substrim(string(long(t_average*3600)), 2)
line_command(13) = substrim(string(long(sample_rate*3600)), 2)
line_command(14) = part_split
line_command(15) = substrim(string(long(sync_time*3600)), 2)
line_command(16) = ctl
line_command(17) = ifine
line_command(18) = iout
line_command(19) = ipout
line_command(20) = LSUBGRID
line_command(21) = LCONVECTION
line_command(22) = LAGESPECTRA
line_command(23) = IPIN
line_command(24) = IFLUX
line_command(25) = MDOMAINFILL
line_command(26) = IND_SOURCE
line_command(27) = IND_RECEPTOR
line_command(28) = MQUASILAG
line_command(29) = NESTED_OUTPUT

openw, outfile, filepath+'options/COMMAND', /GET_LUN
FOR i = 1, 29 DO BEGIN
  printf, outfile, line_command(i)
ENDFOR
free_lun, outfile

;END write command file

;write pathnames file
line1 = filepath+'options/'
```

```

line1 = filepath+'options/'
line2 = outpath
line3 = '/local/reh/rcowen/group_flex_idl/ECMWF'+index+'/'
line4 = '/local/reh/rcowen/group_flex_idl/ECMWF'+index+'/AVAILABLE'
line5 = '===== '
line6 = '

openw, outfile, filepath+'pathnames', /GET_LUN
printf, outfile, line1
printf, outfile, line2
printf, outfile, line3
printf, outfile, line4
printf, outfile, line5
printf, outfile, line6
free_lun, outfile

;END writing pathnames file


;write includepar
line1 = '      parameter(maxpart='+maxpart+')'

IF (direction EQ 1) THEN maxpointspec = 'maxspec' ELSE $
maxpointspec = 'maxpoint'

line2 = '      parameter(maxpoint='+maxpoint+',maxspec='+maxspec+$
',maxpointspec='+maxpointspec+')'

openw, outfile, '/local/reh/rcowen/group_flex_idl/modified/temp.txt', /get
_lun
printf, outfile, line1
printf, outfile, line2
free_lun, outfile

IF (min(yrs) LE 2005) THEN incpar_fil = 'includepar_top1' ELSE incpar_fil =
= 'includepar_top3'

rm_command = '\rm '+filepath+'includepar'
cat_command = 'cat /local/reh/rcowen/group_flex_idl/modified/'+incpar_fil+
$
' /local/reh/rcowen/group_flex_idl/modified/temp.txt '+$
'/local/reh/rcowen/group_flex_idl/modified/includepar_bottom'+$
' > '+filepath+'includepar'

spawn, rm_command
spawn, cat_command

;END write includepar


;END writing files for modification


;link the necessary met files to the temp met directory and write the
;AVAILABLE file

met_dir = '/local/reh/rcowen/group_flex_idl/ECMWF'+index+'/'

met_source = '/local/reh/ecmwf/'

```

```

start_met = julday(start_run(1), start_run(2), start_run(0), start_run(3))
end_met = julday(end_run(1), end_run(2), end_run(0), end_run(3))

this_met = start_met-1.

command_met1 = '\rm '+met_dir+'EN*'
spawn, command_met1

WHILE (this_met LE (end_met+1.)) DO BEGIN

    caldat, this_met, met_mon, met_day, met_yr, met_hr
;form is EN03032600 ENYYMMDDHH
    yr = strmid(strtrim(string(met_yr), 2), 2, 2)
    mo = strtrim(string(met_mon), 2)
    IF (mo LE 9) THEN mo = '0'+mo
    da = strtrim(string(met_day), 2)
    IF (da LE 9) THEN da = '0'+da
    hr = strtrim(string(met_hr), 2)
    IF (hr LE 9) THEN hr = '0'+hr

    met_source_file = met_source+'EN'+yr+mo+da+hr
    test_file_exist = findfile(met_source_file)
    IF (test_file_exist NE '') THEN BEGIN
        command_metloop = 'ln -s '+met_source+'EN'+yr+mo+da+hr+' '+met_dir+'.'
        spawn, command_metloop
    ENDIF ELSE print, 'missing '+met_source_file

    this_met = this_met+0.125
ENDWHILE

;write AVAILABLE file in the

write_available, met_dir

;write ageclasses file

openw, outfile, filepath+'options/AGECLASSES', /GET_LUN

printf, outfile, '*****'
printf, outfile, '*                                     *'
printf, outfile, '*Lagrangian particle dispersion model FLEXPART *'
printf, outfile, '*           Please select your options           *'
printf, outfile, '*                                     *'
printf, outfile, '*This file determines the ageclasses to be used*'
printf, outfile, '*                                     *'
printf, outfile, '*Ages are given in seconds. The first class *'
printf, outfile, '*starts at age zero and goes up to the first *'
printf, outfile, '*age specified. The last age gives the maximum *'
printf, outfile, '*time a particle is carried in the simulation. *'
printf, outfile, '*                                     *'
printf, outfile, '*****'
printf, outfile, '1               Integer           Number of age classes'

age_classes_out = age_classes*60*60*24
printf, outfile, format = '(I10)', age_classes_out

free_lun, outfile

cd_command = 'cd '+filepath

```

03/19/09

run_retro_flex.pr

```
;remove old FLEXPART executable so we dont run in case there is a
;compile problem
rm_command = cd_command+'; \rm FLEXPART'
spawn, rm_command

;compile and execute FLEXPART
compile_command = 'gmake -C '+filepath
spawn, compile_command

execute_command = cd_command+' ; ./FLEXPART'
spawn, execute_command

return
end
```

B.4.6 write_available.pro



03/19/09

write_available.pr ↗

```
PRO write_available, working_dir

files = working_dir+'EN*'

fnl_files = file_search(working_dir+'EN*', COUNT = n_fnl_files)
filebreak, fnl_files, FILE = file_names
fnl_files = STRTRIM(fnl_files, 2)

out_file = working_dir+'AVAILABLE'

openw, outfile, out_file, /GET_LUN
printf, outfile, 'DATE      TIME      FILENAME      SPECIFICATIONS'
printf, outfile, 'YYYYMMDD HHMISS'
printf, outfile, '_____ _____ _____ _____'

FOR loop = 0, (n_fnl_files-1) DO BEGIN

    year = STRMID(file_names(loop), 2, 2)
    IF (year LE 15) THEN year = '20'+year ELSE year = '19'+year
    month = STRMID(file_names(loop), 4, 2)
    day = STRMID(file_names(loop), 6, 2)
    hour = STRMID(file_names(loop), 8, 2)

    outline =year+month+day+' '+hour+'0000      '+file_names(loop)+' ON ↗
DISC      '
    printf, outfile, outline

ENDFOR

free_lun, outfile

RETURN
END
```


Appendix C

Copyright permissions and information

This section contains copyright information for Chapters 2 and 3, which are based on the articles *Owen et al.* [2006] and *Owen and Honrath* [2009].

C.1 Documentation for Chapter 2

Chapter 2 is based entirely on the *Owen et al.* [2006], the copyright of which is held by the American Geophysical Union. The two following two subsections contain the email sent to request permission and information regarding reproduction of the paper in this dissertation and the response granting that right.

C.1.1 Email requesting for reproduction permission

Subject: Use of papers in dissertations and thesis

From: "R. Chris Owen" <rcowen@mtu.edu>

Date: Fri, 17 Apr 2009 00:22:42 -0400

To: jgr-atmospheres@agu.org

Hello,

I am completing a doctoral dissertation at Michigan Technological University entitled "Long-range pollution transport: Trans-Atlantic mechanisms and Lagrangian modeling methods." I am seeking permission to reprint as a chapter in my dissertation the contents from my JGR Atmospheres article:

Owen, R. C., O. R. Cooper, A. Stohl, and R. E. Honrath, An analysis of the mechanisms of North American pollutant transport to the central North Atlantic lower free troposphere, J. Geophys. Res., 111 , D23S58, 2006.

My understanding, based on communications with JGR regarding similar uses by other doctoral students at my university, is that no express permission is required to use articles in this fashion. Is this correct? Please note that my dissertation may be published by ProQuest/UMI, which I believe is for archive and inter-library sharing purposes.

Any information or assistance you can provide regarding this issue would be appreciated.

Kind regards,

Chris Owen

C.1.2 Email granting reproduction permission

permission

Subject: permission
From: Michael Connolly <MConnolly@agu.org>
Date: Fri, 17 Apr 2009 13:18:39 -0400
To: rcowen@mtu.edu

We are pleased to grant permission for the use of the material requested for inclusion in your thesis. The following non-exclusive rights are granted to AGU authors:

- All proprietary rights other than copyright (such as patent rights).
- The right to present the material orally.
- The right to reproduce figures, tables, and extracts, appropriately cited.
- The right to make hard paper copies of all or part of the paper for classroom use.
- The right to deny subsequent commercial use of the paper.

Further reproduction or distribution is not permitted beyond that stipulated. The copyright credit line should appear on the first page of the article or book chapter. The following must also be included, "Reproduced by permission of American Geophysical Union." To ensure that credit is given to the original source(s) and that authors receive full credit through appropriate citation to their papers, we recommend that the full bibliographic reference be cited in the reference list. The standard credit line for journal articles is: "Author(s), title of work, publication title, volume number, issue number, citation number (or page number(s) prior to 2002), year. Copyright [year] American Geophysical Union."

If an article was placed in the public domain, in which case the words "Not subject to U.S. copyright" appear on the bottom of the first page or screen of the article, please substitute "published" for the word "copyright" in the credit line mentioned above.

Copyright information is provided on the inside cover of our journals. For permission for any other use, please contact the AGU Publications Office at AGU, 2000 Florida Ave., N.W., Washington, DC 20009.

Michael Connolly
Journals Publications Specialist

C.2 Documentation for Chapter 3

Chapter 3 is based entirely on the *Owen and Honrath* [2009]. The author retains copyright of Atmospheric Chemistry and Physics (ACP) articles, and thus no permissions is needed to reproduce the article in this dissertation. The following section contains a copy of the webpage detailing the ACP copyright policy, which was accessed from http://www.atmospheric-chemistry-and-physics.net/general_information/license_and_copyright.html on April 17, 2009.

C.2.1 ACP copyright policy

License and Copyright Agreement


The following License and Copyright Agreement is valid for any article published by Copernicus Publications on behalf of the European Geosciences Union (EGU) in the journal Atmospheric Chemistry and Physics and its discussion forum Atmospheric Chemistry and Physics Discussions whose original manuscript was received from **10 December 2007** on. The License and Copyright Agreement for articles based on manuscripts received before 10 December 2007 can be found [here](#).

Author's Certification

In submitting the manuscript, the authors certify that:

- They are authorized by their co-authors to enter into these arrangements.
- The work described has not been published before (except in the form of an abstract or as part of a published lecture, review or thesis), that it is not under consideration for publication elsewhere, that its publication has been approved by all the author(s) and by the responsible authorities – tacitly or explicitly – of the institutes where the work has been carried out.
- They secure the right to reproduce any material that has already been published or copyrighted elsewhere.
- They agree to the following license and copyright agreement:

Copyright

- Copyright on any article is retained by the author(s).
- Authors grant Copernicus Publications a license to publish the article and identify itself as the original publisher.
- Authors grant Copernicus Publications and the European Geosciences Union commercial rights to produce hardcopy volumes of the journal for sale to libraries and individuals.
- Authors grant any third party the right to use the article freely as long as its original authors and citation details are identified.
- The article and any associated published material is distributed under the [Creative Commons Attribution 3.0](#)  License:

Creative Commons Attribution 3.0 License

Anyone is free:



to Share — to copy, distribute and transmit the work



to Remix — to adapt the work

Under the following conditions:



Attribution. The original authors must be given credit.

- For any reuse or distribution, it must be made clear to others what the license terms of this work are.
- Any of these conditions can be waived if the copyright holders give permission.
- Nothing in this license impairs or restricts the author's moral rights.

The full [legal code](#)  of this license.