2012

Automated optimization of polymer extrusion dies using finite element analysis

James Allen Peitzmeier
Michigan Technological University

AUTOMATED OPTIMIZATION OF POLYMER EXTRUSION DIES USING FINITE ELEMENT ANALYSIS

By

James Allen Peitzmeier

A REPORT

Submitted in partial fulfillment of the requirements for the degree of

MASTER OF SCIENCE

(Mechanical Engineering)

MICHIGAN TECHNOLOGICAL UNIVERSITY

2012

This report, "Automated Optimization of Polymer Extrusion Dies using Finite Element Analysis," is hereby approved in partial fulfillment of the requirements for the Degree of MASTER OF SCIENCE IN MECHANICAL ENGINEERING

Mechanical Engineering-Engineering Mechanics

Signatures:

Thesis Advisor _____
Prof. Mahesh Gupta

Committee Member _____
Prof. Allan Struthers

Committee Member _____
Prof. Song-Lin Yang

Department Chair _____
Prof. William Predebon

Date _____

Abstract

An extrusion die is used to continuously produce parts with a constant cross section; such as sheets, pipes, tire components and more complex shapes such as window seals. The die is fed by a screw extruder when polymers are used. The extruder melts, mixes and pressures the material by the rotation of either a single or double screw. The polymer can then be continuously forced through the die producing a long part in the shape of the die outlet. The extruded section is then cut to the desired length. Generally, the primary target of a well designed die is to produce a uniform outlet velocity without excessively raising the pressure required to extrude the polymer through the die [1]. Other properties such as temperature uniformity and residence time are also important but are not directly considered in this work. Designing dies for optimal outlet velocity variation using simple analytical equations are feasible for basic die geometries or simple channels. Due to the complexity of die geometry and of polymer material properties design of complex dies by analytical methods is difficult. For complex dies iterative methods must be used to optimize dies. An automated iterative method is desired for die optimization.

To automate the design and optimization of an extrusion die two issues must be dealt with. The first is how to generate a new mesh for each iteration. In this work, this is approached by modifying a Parasolid file that describes a CAD part. This file is then used in a commercial meshing software. Skewing the initial mesh to produce a new geometry was also employed as a second option. The second issue is an optimization problem with the presence of noise stemming from variations in the mesh and cumulative truncation errors. In this work a simplex method and a modified trust region method were employed for automated optimization of die geometries. For the trust region a discreet derivative and a BFGS Hessian approximation were used. To deal with the noise in the function the trust region method was modified to automatically adjust the discreet derivative step size and the trust region based on changes in noise and function contour.

Generally uniformity of velocity at exit of the extrusion die can be improved by increasing resistance across the die but this is limited by the pressure capabilities of the extruder. In optimization, a penalty factor that increases exponentially from the pressure limit is applied. This penalty can be applied in two different ways; the first only to the designs which exceed the pressure limit, the second to both designs above and below the pressure limit. Both of these methods were tested and compared in this work.

Table of Contents

Abstract 3
List of Figures 6
List of Tables 8
Introduction 9
Literature Review 12
Numerical Optimization 17
  Nelder Mead 17
  Trust Region 23
Flat Die 28
  Algorithm Adaptation for the Flat Die 28
    Nelder Mead 28
    Trust Region 29
  Geometry Description 30
  Parasolid 33
  Meshing 34
  Results 36
    Flat Die with Nelder Mead 36
    Flat Die with Trust Region Optimization 37
Spiral Mandrel Die 46
  Algorithm Adaptation for the Spiral Mandrel Die 46
    Nelder Mead 46
    Trust Region 47
  Geometry Description 47
  Parasolid 48
  Mesh skewing 49
  Results 53
Conclusions 59
Future Work 60
Bibliography 62
Appendix 63
  Generating Parasolid models for the Spiral Mandrel Die 63
  Results flat die, first optimization 66
  Additional figures for the flat die version of the trust region algorithm 70

List of Figures

Figure 1: Left: Fishtail die Right: Coat hanger die; A: Primary Manifold B: Secondary Manifold 10
Figure 2: Flow channel geometry of the coat-hanger used in this study 11
Figure 3: Flow channel geometry of the spiral mandrel die used in this work 11
Figure 4: Nelder Mead on Rosembrock function 19
Figure 5: Weighted Nelder Mead with value weighting on Rosembrock function 19
Figure 6: Nelder Mead on Equation 2 20
Figure 7: Weighted Nelder Mead with value weighting on Equation 2 20
Figure 8: Nelder Mead with multiple minima on Equation 2 21
Figure 9: Gradient method on Rosenbrock 24
Figure 10: Gradient method with multiple minima 24
Figure 11: Drawing flat die 33
Figure 12: Noise Study for 4, 6 and 8 layered meshes 35
Figure 13: Convergence for the Nelder Mead Algorithm on the flat die 37
Figure 14: Improvement of objective function throughout Equality Penalty Trust Region Optimization 39
Figure 15: Improvement of objective function throughout In-Equality Penalty Trust Region Optimization 40
Figure 16: Dimensions throughout Equality Penalty Trust Region Optimization 41
Figure 17: Dimensions throughout In-Equality Penalty Trust Region Optimization 42
Figure 18: Original Design Velocity Vector plot at Exit and Original Die Model 44
Figure 19: Second Solution set with Equality Penalty 44
Figure 20: Second Solution set with In-Equality Penalty 44
Figure 21: Velocity Distribution across Die Exit, from the second solution set 45
Figure 22: Temperature Distribution across Die Exit, from the second solution set 46
Figure 23: Spiral Mandrel Die Cross Section Veiw 48
Figure 24: Distortion effects on Objective function (a), minimal noise near undistorted die design (b) and significant noise further from undistorted die design (c) 51
Figure 25: Noise near Optimum found using trust region method, with the correction to node 8589 noted in meters. 53
Figure 26: Velocity Distribution across Die Exit for Trust Region Solutions to the Spiral Mandrel Die. 54
Figure 27: Velocity Distribution across Die Exit for Nelder Mead Solution to the Spiral Mandrel Die. 54
Figure 28: Comparison of the Original Design and the Solution from the Trust Region Method using the Equality Penalty. 55
Figure 29: Solutions for Spiral Mandrel die using Nelder Mead and Trust Region methods. a) Lobe radius (mm) by iteration number; b) Length after helix section (mm) by iteration number; c)Length of helix section (mm) by iteration number; d)Objective function (mm/s) with penalty factor by iteration number; e)Pressure by iteration number 57
Figure 30: Distortion near convergence for Spiral Mandrel die. a) Comparison of objective function values for mesh D and mesh E; b) Detail view of objective function values for mesh D 58
Figure 31: Nodes used to in Helix B-spline 64
Figure 32: One set of 12 nodes from the Helix B-spline, for three different lobe radiuses 65
Figure 33: Radius of the helix as compared to linear, helix radius measured in NX 5.0 65

Figure 34: Objective function progression for the flat die, 2nd version trust region algorithm, first solution, and equality penalty 67
Figure 35: Pressure progression for the flat die, 2nd version trust region algorithm, first solution, and equality penalty 67
Figure 36: Optimized Design Velocity Vector plot at Exit and Optimized Die Model from the first solution set using an equality penalty 68
Figure 37: Optimized Design Velocity Vector plot at Exit and Optimized Die Model from the first solution set using an equality penalty 68
Figure 38: Velocity Distribution across Die Exit, from the first solution set 69
Figure 39: Temperature Distribution across Die Exit, from the first solution set 69
Figure 40: Development of objective function throughout Equality Penalty Trust Region Optimization 70
Figure 41 : Development of objective function throughout In-Equality Penalty Trust Region Optimization 70
Figure 42: Dimensions throughout Equality Penalty Trust Region Optimization (full data set) 71
Figure 43: Dimensions throughout In-Equality Penalty Trust Region Optimization (full data set) 71

List of Tables

Table 1: Variables for the flat die 32
Table 2: Flat die constant parameters 32
Table 3: Initial and Final Dimensions 43
Table 4: Variable Parameters for undistorted meshes 50

Introduction

The focus of this work was to develop a method to automatically optimize the design of polymer extrusion dies. A finite element code was used for evaluation of designs and a numerical optimization algorithm was used to develop the new designs. In this work a fishtail die or flat die was considered for forming flat sheets or films and a spiral mandrel die to produce pipe or ring shapes. A well designed die minimizes velocity variation at the exit. This is constrained by a limit on the allowable pressure differential across the die. Other parameters such as residence time and temperature variation at exit are important but not directly considered in this work. For each die the project involved two distinct steps. The first was to evaluate a proposed design and return an objective function value as a measure of performance. This involves building the model, meshing, running a finite element simulation and evaluating the result. Commercial software packages were used for meshing and finite element simulation and will not be covered in detail. [2] [3] The second step was to optimize the die design. The optimization was complicated because of long computation times to define the die and simulation. A noisy objective function also complicated optimization.

Production of an extruded polymer part starts with a screw extruder. A screw extruder melts and pressurizes the polymer. Thermoplastic pellets can be fed to the screw by a hopper. The screw then rotates pushing the pellets forward; melting, mixing and pressurizing the material in the process. Energy to melt the polymer is primarily supplied by viscous heating from the rotation of the screw. Initially and when necessary, energy can be supplied by electric heating elements around the barrel. A breaker plate is in front of the screw to prevent any solids from being extruded and to reduce the rotational motion of the polymer. Screw extruders can be used to produce plastic products such as pipes, rods, sheets, and films. More complex cross sectional shapes can be produced with profile dies.

Flat dies and spiral mandrel dies are considered in this project for extrusion of sheets and pipes, respectively. A flat die or fish tail die is used to produce sheets and films. The polymer enters the die through a centered low aspect ratio inlet

channel and exits through a high aspect ratio rectangular shape. The inlet leads to the manifold which is a large traverse channel that is the width of the final sheet. The manifold is angled towards the die outlet from the center to reduce flow resistance to the edges while increasing flow resistance to the center. Following the manifold is a relatively narrow channel, the secondary manifold or island. Two variations of the flat die are distinguished by the shape of the primary and secondary manifolds, see Figure 1. In a fishtail die the length of the secondary manifold decreases linearly from center to edge creating a strong primary manifold. In a coat hanger die the length decreases non-linearly based on flow resistance of the die creating a curved primary manifold.

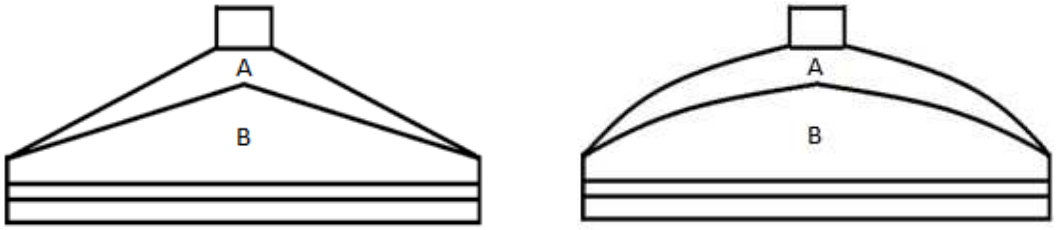

Figure 1: Left: Fishtail die Right: Coat hanger die; A: Primary Manifold B: Secondary Manifold

The fishtail die is simpler to produce but the coat hanger die has been shown to have better performance. In this study a fishtail die will be developed, the original design can be seen in Figure 2.

The secondary manifold improves flow due to its relatively high resistance. This resistance combined with the low resistance in the manifold produce a flow that is relatively even across the secondary manifold. At the end of the secondary manifold is the short narrow land. The land is normally the thickness of the desired sheet or film. For some materials and geometries the polymer will expand after the exit of the die producing a thicker part than the land. This is known as die swell and is caused by the strain rate sensitive elongational viscosity of the polymer. To counteract this a slightly thinner land may be produced to account for die swell. Sheets can be produced that can either be used as sheets as in the boards and windows in an ice rink or may later be reformed in other processes such as vacuum forming to produce a refrigerator liner [4]. [1]

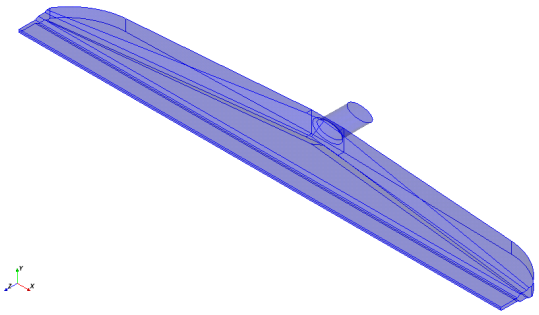

Figure 2: Flow channel geometry of the coat-hanger used in this study

The spiral mandrel die produces a pipe or hollow cylinder shape. In the variation of spiral mandrel die, considered in this work, four inlet channels lead to the mandrel, which is the portion of the die that will form the inside diameter (ID) of the pipe. The four channels follow an expanding helical path. The channels would be cut from the mandrel using a hemispherical milling bit following the path of this expanding helix, creating a channel that has at any given point a circular cross section in the direction tangent to the helix. The mandrel is a conical section with a base at the die inlet where the diameter is equal to the part OD. At the base the entire flow is through the channels. The mandrel's diameter then reduces to the part ID at the outlet. As the mandrel's diameter reduces the flow shifts from fully in the channels to fully in the gap between mandrel and OD of the die. The spiral mandrel die can be seen below in Figure 3.

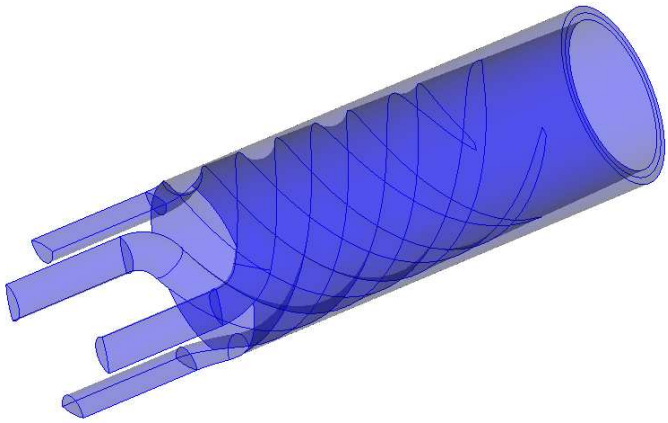

Figure 3: Flow channel geometry of the spiral mandrel die used in this work

Another similar variation that is not considered in this work supports the mandrel with legs that radiate out to the OD. This is known as a spider mandrel and is prone to weld lines due to the legs briefly separating the flow. Smearing devices have been applied downstream of the spider legs. These can be effective in reducing the effect of the separated flow but are often insufficient.

Profile dies can produce complex parts which have a constant cross section in one dimension. These dies can produce window seals, door trim and the components used to build a tire. Profile dies are not well suited for optimization programs due to the wide variety of designs that may be produced. Each new feature added to the topology would require the model generation and optimization code to be modified. Further, profile dies are complex. The number of parameters that should be considered would be cumbersome in optimization.

Current techniques for designing these dies require the time of experienced engineers and may need to be repeated for changes in die scale, shape, processing temperature, materials and flow rate. The goal of this project is to develop a numerical optimization program that will autonomously find the best design. This program would allow even an inexperienced engineer to quickly develop the die design that is considered optimal by simulation.

Literature Review

A well designed die should have uniform exit velocity and temperature distribution. A higher velocity in a given section will produce increased thickness in that section of the final part due to increased volumetric flow and higher elongational strain rate. [5] Increased temperature will create a thinner section due to increased shrinkage. A well designed die will also avoid excessive residence times and concentrated hot spots as overly high temperatures or prolonged time at high temperatures will degrade the polymer chains and change the properties of the material [4]. Non uniform residence times will lengthen the purging process that is required when materials or colors are changed and create variations in foaming or cross linking due to temperature history effects between the additives and the

polymer [6]. The most important of these concerns for a die designer is the uniform velocity at the exit with the constraint of a limited pressure source in the screw extruder.

Typically a die would be designed by an experienced engineer. To determine the shape of the die, a system of equations can be used to calculate the length of the secondary manifold on fishtail dies and to calculate both the secondary manifold length and primary manifold curvature on coat hanger dies. The latter requiring an iterative solution. These equations are for mean pressure loss or residence time for the die as a summation of a flow in the primary or secondary manifolds treated as separate fully developed flows. A boundary condition is set for uniformity at the exit. [1]. This method is limited when applied to more complex dies. For example, the flow in the channels of the spiral mandrel die are distinctly angled away from the channel direction and multiple stream lines will lead to the same point at the outlet. This contradicts the assumptions used and complicates calculating flow rates needed to develop the system of equations.

It has also been shown that designs can be solved numerically. The die design can be divided into a finite number of sub regions. Each region is then solved as a short channel section dependent on material properties, channel geometry, material temperature and boundary conditions; volumetric flow rate and pressure at the section boundaries. From this either the curvature of the primary manifold, length of the secondary manifold or height of the secondary manifold can be calculated. Further approximations for inlet losses between different segments can be included. This method allows more flexibility in die geometry and outlet boundary conditions. [1].

To allow for fine tuning of the die an adjustable land and choker bar may be added to the die when it is produced. An adjustable land allows for a couple millimeters adjustment at the end of the land by deflecting the die wall inwards to adjust for die swell. A choker bar is an adjustable section in the secondary manifold that can be used to increase flow resistance. Either tuning tool can be used to make fine adjustments and allow a wider range of materials and conditions to be used in a

given die. The process of physically adjusting these devices is complicated and requires an experienced operator or an automated system [1].

This process can be improved by using computational fluid dynamics (CFD) software. Using CFD code, 3D flow behavior can be modeled. The effects of viscous heating due to shear at the walls and viscous heating due to elongation can be captured. Applying typical CFD methods to polymers requires some modification to accurately account for non-Newtonian material properties. The flow is initially simulated as Newtonian, then the shear rates from the initial solution are used to calculate the shear and elongational viscosity for a given element. This process repeats using the previous solution until convergence. An engineer must then review the results, improve the design, modify the CAD model and restart the simulation. Similar to older methods this approach is time consuming and relies on the experience of the engineer to approach an optimal solution. An approach to replace the engineers efforts with an optimization code would reduce the time the engineer would have to invest and reduce the time and cost of designing extrusion molds. In practice this code has been separated in two parts; model and mesh generation in the first half and numerical optimization in the second [4].

An early study by Smith [6] on a flat die used 18 parameters that were varied to minimize pressure and residence time. Velocity variance and volumetric flow rate were held as defined constants. The problem was optimized using a Galerkin finite element method where a 2D mesh is defined to describe the in-plane cavity shape while 3D effects are modeled by treating height as a node property and applying a shape function to approximate the flow. This study demonstrated that including uniform residence time distribution term into the objective function will affect the velocity variance at the optimum point.

Flow approximation methods have been attempted by Michaeli [7] to reduce to computational load of successive FEA solutions. Michaeli used a flow-analysis network (FAN) coupled with a finite element model to optimize a simple die. The initial FEA solution was used to determine resistance values for sub regions. These sub regions are then quickly optimized individually and these individual solutions

are used in the next iteration to be solved by FEA. This method worked well when applied on a simple die with minimal lateral movement. This method may become cumbersome when applied to more complex dies with significant lateral movement.

In the work by Sun [5] both the flow simulation and die optimization processes were studied. Sun used the Carreau model for shear viscosity and the Sarkar-Gupta model for elongational viscosity. This was compared to the typical method of using a Newtonian model with shear thinning viscosity. The latter model is considered sufficient in shear dominated flows but extrusion dies have elongational dominated regions. Both models accounted for temperature dependence by the Arrhenius model. Both a spiral mandrel die and flat die were considered in this portion of the study. The first part of this study was to examine the improvement by this more complex model. Pressure across the flat die varied by up to 17% between the models and it was determined simulation of extrusion dies should include modeling the strain rate dependence of elongational viscosity.

The second portion of the Sun study focuses on a variation of the flat die, called a coat hanger die. The optimization used was a nine dimensional line search algorithm which uses adjoint sensitivity analysis to measure the gradient and a BFGS estimation of the Hessian. A constraint was added to limit the pressure to minimal increases from the original design. This was accomplished by including an equality penalty factor on the pressure of the original design. Since the goal is only to maintain a similar pressure to the original the weight of the penalty was not increased during optimization but instead maintained as a constant. Compared to a finite difference gradient, the adjoint sensitivity analysis had limited improvement in accuracy. This is due to the basis of the method, the stiffness matrix and the force vector, are found by a finite difference calculation. This leads to the same errors as a finite difference gradient when the step size is either too large or too small. The benefit is that this method does work well to reduce computational load. With comparable accuracy it reduces computational time by a factor of nearly n+1, where n is the number of design parameters.

A flat die similar to the design used in this report was optimized by Lebaal [8] using a response surface method (RSM). A Kriging interpolation_is used to approximate the model after a number of FEA evaluations are performed. The Kriging model is then optimized using multiple initial conditions to avoid local minima. Using this system the velocity variation optimum was found and temperatures were shown to be more uniform due to the improved uniformity of viscous heating. The FEA solver was REM3D ® and model changes were done by a MatLab® script.

Optimization methods developed for other applications can also be applied to extrusion dies. The Nelder-Mean simplex has been improved in a study by Kelley [9] that reduced the risk of stagnation by developing criteria similar to line search criteria to identify higher stagnation potential. The solver would then restart by contracting the set of evaluation points used to determine the next iteration step. Then it would rearrange to an orthogonal set where one evaluation is in the direction of estimated steepest decent based on the previous set. When predicting steepest decent, only the sign of the gradient is used since the stagnated set is a poor predictor of slope. This study has shown a Nelder-Mead method can be improved by applying success criteria and restarting when necessary.

When using any slope based method a discreet gradient would need to be calculated. Unfortunately sufficient noise can degrade or destroy the accuracy of a discrete gradient measurement. Okano [10] used a stochastic noise reaction method successfully in approximating a gradient in the presence of noise. Although this method was successful it requires too many function calls per gradient measure to be practical for this project.

Davis [11] made a study that has investigated three proposed methods for noisy optimization problems that lack direct evaluation of gradients. A simplex method modified to convert to steepest decent when close to solution worked best for the given problem. A modified SQP (sequential quadratic programming) and a Nelder-Mead method also performed well. The gradient and Hessian were calculated from a set of points around the current best point using a least squared approximation. The points are within the trust region of the model and the number of points

increases as the optimum is approached. Simultaneously the trust region is constricted when the optimum is estimated to lie within the trust region so that the set of points will be more concentrated around the expected optimum.

Numerical Optimization

Nelder Mead

In this study two different optimization algorithms were investigated. These algorithms were initially developed as two dimensional test cases in matlab. The first, a Nelder Mead simplex is the simpler of the two and can make a step forward with as little as 1 evaluation. This algorithm has been shown to work in noisy environments with some modification [9]. The Nelder Mead requires n+1 points organized in an orthogonal set, n was the number of dimensions in the optimization problem. Orthogonality must be maintained or the algorithm would lose the ability to detect a gradient in a given direction. A vector was drawn from the worst point to a mean point taken from the remaining points. The next evaluation point was taken by multiplying this vector by a factor then adding it to the mean point. At first the factor was taken to be one, that is the original vector was added to the mean point. This new point was then evaluated. If this point was better than the any other point in the group, a factor of 2 would be immediately evaluated. If the new point was not better than any other point in the group, a half step from the mean would be taken either in the same direction or the opposite direction and evaluated. When the step with the factor 1 was an improvement from the previous worst point, the second step would be taken in the same direction. When the first step was worse than the previous worst point, the second step would be taken in the opposite direction. Eventually the best value was taken and included in the new group of points. If all values fail to improve on the worse point the simplex will converge by moving all points in towards the best point of the group. The algorithm was tested with the Rosembrock function (Equation 1) and another function with multiple minima, (Equation 2). Figure 4 shows the algorithm on the Rosembrock function with a normal mean and Figure 5 shows the same function with a weighted mean that

favors points with lower and better values. In these plots the simplex triangle connects the three points required for two dimensional optimization. Only the function value from these three points were used to determine the next step. For both cases the algorithm initially focused on the large improvements moving to the center of the canyon. Then both cases had to adjust and move through the canyon. Near the optimum many evaluations were used to as the algorithm approaches convergence. Figure 6 and Figure 7, again with the normal mean and weighted mean respectively, use the multiple minima function but start around a single minimum for comparison. Here it is more clear the advantage of the weighted mean. In Figure 7 the evaluation points are more closely centered around the optimum and fewer evaluations are needed. In Figure 8 the algorithm uses the normal mean and demonstrates the algorithms inability to find a global minimum in a function with multiple minima. Additional processes would be necessary to handle multiple minima.

$$f(x,y) = 100 * (y - x^2) + (1 - x)^2 \tag{1}$$

$$g(x,y) = e^{sin(50*x)} + sin(60 * e^{(y)}) + sin(70 * sin(x)) + sin(sin(80 * y)) - sin(10 * (x + y)) + \frac{1}{4} * (x^2 + y^2) \tag{2}$$

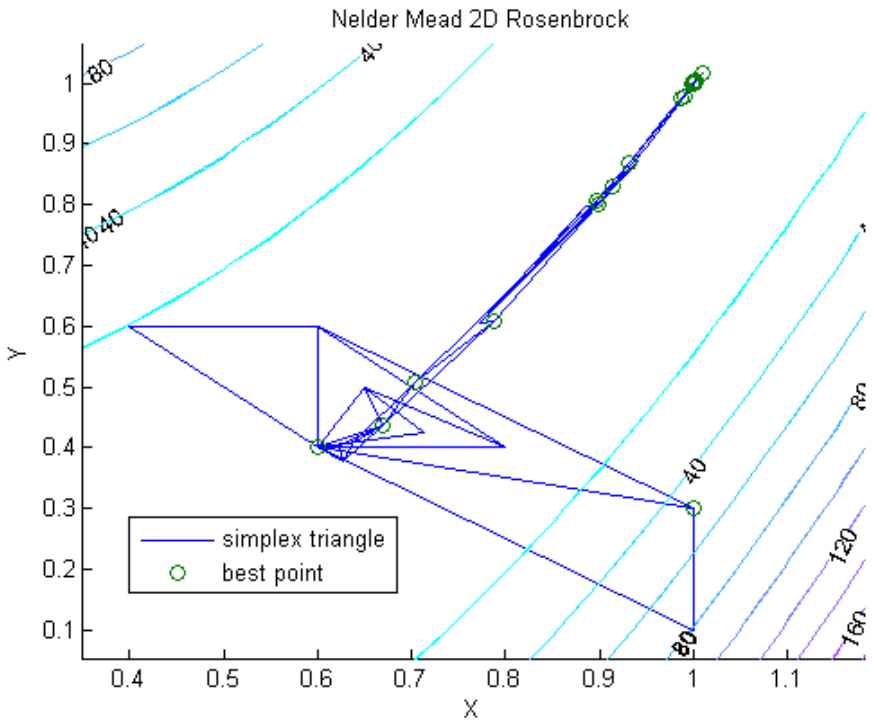

Figure 4: Nelder Mead on Rosembrock function

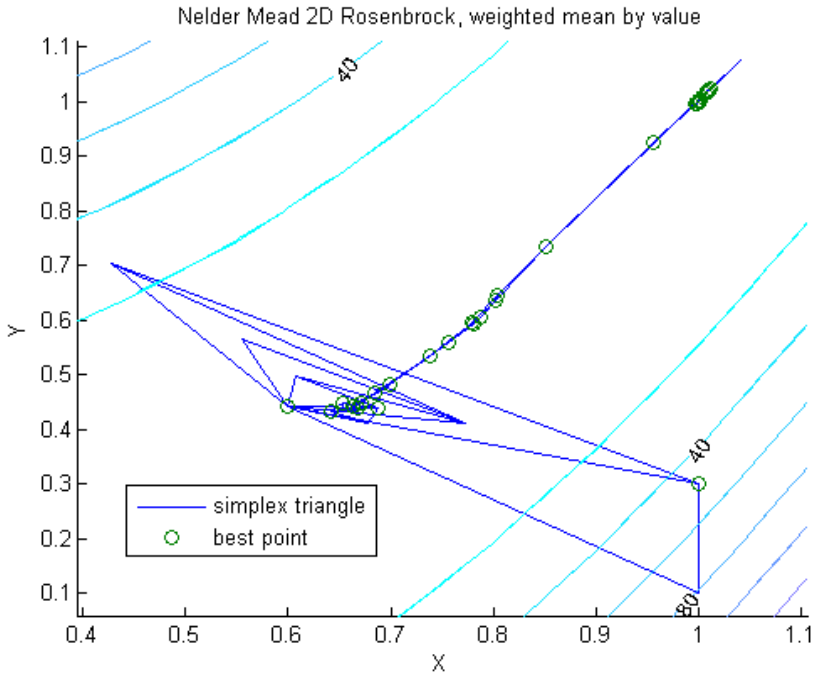

Figure 5: Weighted Nelder Mead with value weighting on Rosembrock function

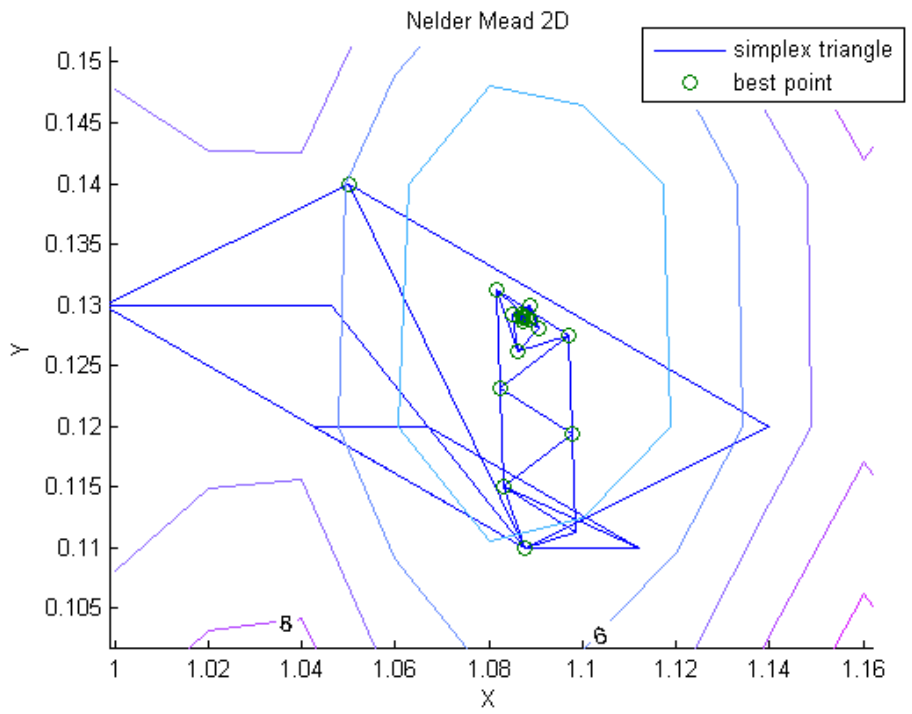

Figure 6: Nelder Mead on Equation 2

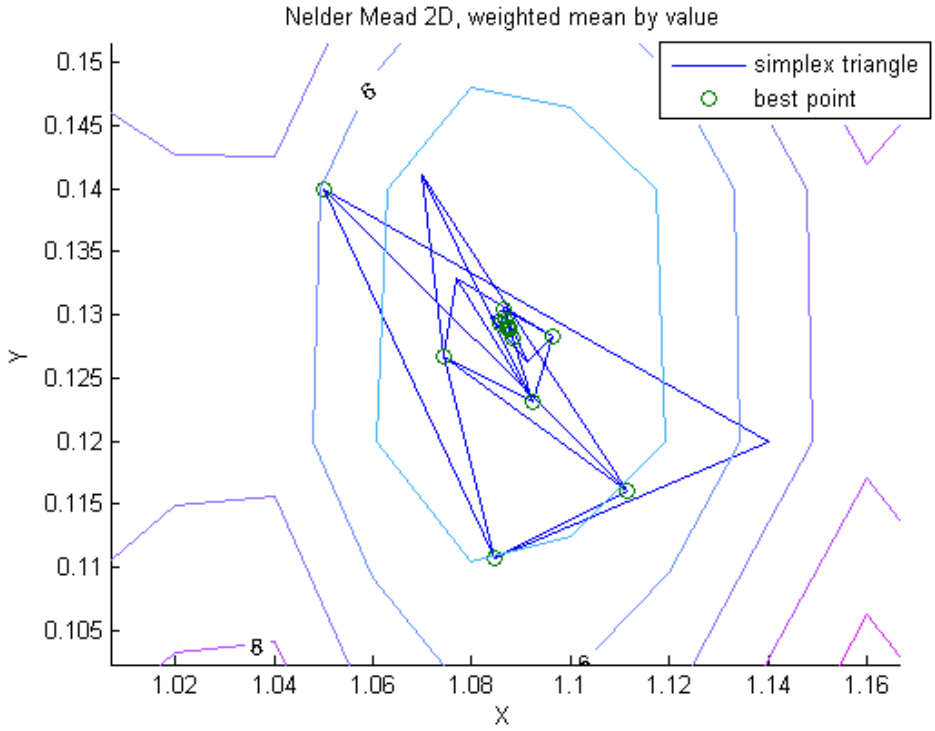

Figure 7: Weighted Nelder Mead with value weighting on Equation 2

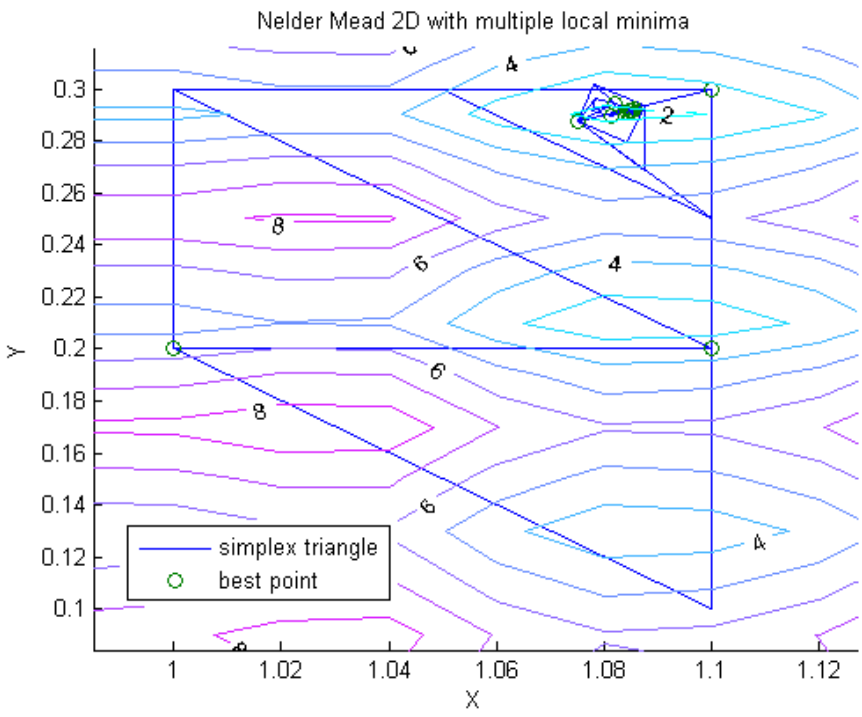

Figure 8: Nelder Mead with multiple minima on Equation 2

Comparing the Nelder Mead algorithm using a normal mean and a weighted mean revealed little practical difference. Occasionally the weighted version would take a step that was more productive than the non-weighted would have but the overall number of function calls were comparable. Despite this, the weighted version was investigated with the dies. The exact rates for these sample problems was of minimal importance. The actual problems studied were three and six dimensional and had to contend with a degree of noise.

When optimizing the design of a die, the objective function which needed to be minimized is given in Equation 3. The objective function measured the velocity variation at the exit. A penalty was added to prevent the pressure across the die from increasing beyond a specified value. When the first point was evaluated the pressure from the initial evaluation, or a user entered max pressure, was recorded as the pressure limit, P_0. The objective function, G, was then evaluated.

$$G(x) = V(x) + \frac{1}{\mu} * (P(x) - P_0)^2$$

(3)

x is the evaluation number and P is pressure. μ is defined below and V, the velocity variation is

$$V(x) = 100 * \sqrt{\frac{1}{n} * \sum_{i=1}^{n} |v_i - \bar{v}|^2}$$

(4)

And the penalty factor μ is defined by

$$\frac{1}{\mu} = 100 * \left(\frac{V(1)}{P_0^2}\right)$$

(5)

In this case the objective function used an equality penalty factor because the term $(P(x) - P_0)^2$ in Equation 3 increases when the pressure was either above or below the pressure limit. This typically is reasonable because it was expected that the best design will be at the pressure limit. The equality penalty focused the search in a smaller area. The other option was to only penalize a design when it was over the pressure limit. Here $P(x)$ is set to be equal to P_0 if $P(x) < P_0$. In this case the algorithm was free to explore lower pressure designs and had the ability to move away from the pressure limit in cases of a local minima. There was increased risk of becoming stuck in a local minima in the larger design space. Both equality and in-equality penalties have been tested in this study.

Every design called for by the algorithm was checked for geometric validity. For example, an ID must be less than an OD. A function would correct these values and recheck the other dimensions with the corrected values. If an original design value was invalid the user would be prompted to change the value. The program would suggest the value with minimal change that would correct the problem.

Trust Region

The trust region method was also first tested in matlab on two dimensional problems. The same functions (Equation 1 and Equation 2) were also used to test the trust region method.

A gradient based method using a BFGS Hessian approximation to take a dog leg step inside a trust region was used. This method requires function evaluation and gradient evaluation at the current point. The initial step was in the steepest decent direction. The new point and its gradient were then evaluated. The information about the change in slope was then used to estimate the Hessian matrix. Now using the gradient and Hessian information, a model of the function was estimated and the minimum point of the estimated model was found. If this point was further than a set distance from the current point, that is outside the trust region, then the step would have been to the edge of the trust region using a dog leg step. A dog leg step initially would find the minimum point in the steepest decent direction. If this was inside the trust region the new point would be located at the trust region limit linearly between this steepest decent minimum and the model estimated minimum, somewhere outside the trust region. If the minimum in the steepest decent direction was outside the trust region, the step would be simply taken in the steepest decent direction to the trust region limit. When the new value matched well or was better than the estimated value based on the model function the trust region was allowed to expand. When the new value was worse than the current or shows much less improvement than expected the trust region would contract. In the matlab test case the gradient was available analytically. In the project the gradient had to be found using discrete derivatives. This added evaluation calls and additional noise to the system when the discrete step become too small. Figure 9 and Figure 10 show the gradient method solving the test cases. The trust region algorithm was generally quicker than the Nelder Mead method on the test functions. Also the trust region algorithm tended to test a narrower variety of points in searching. It was expected that this would be a disadvantage when used on a noisy function where diversity of evaluation points may smooth noisy errors.

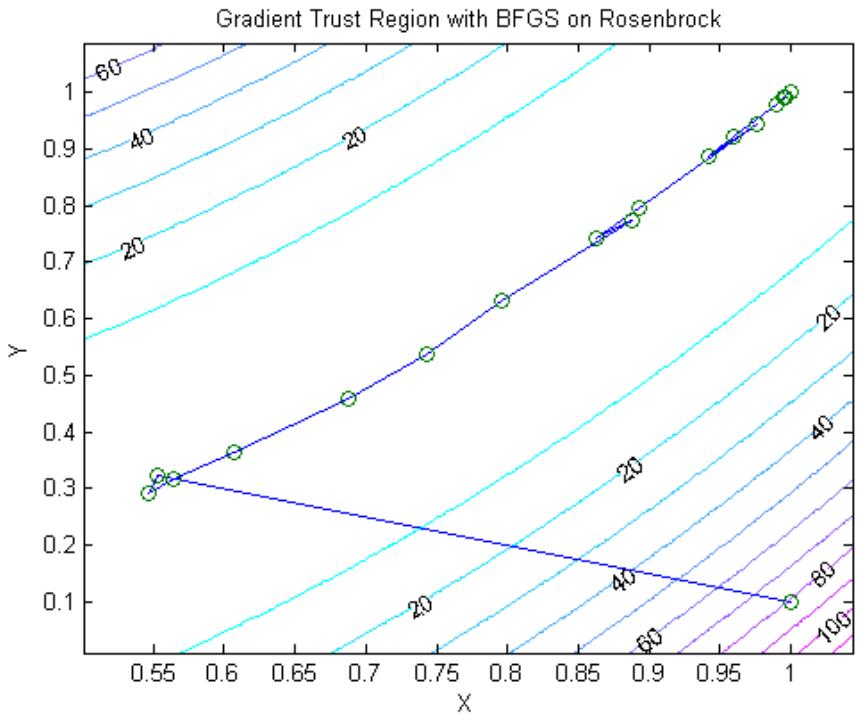

Figure 9: Gradient method on Rosenbrock

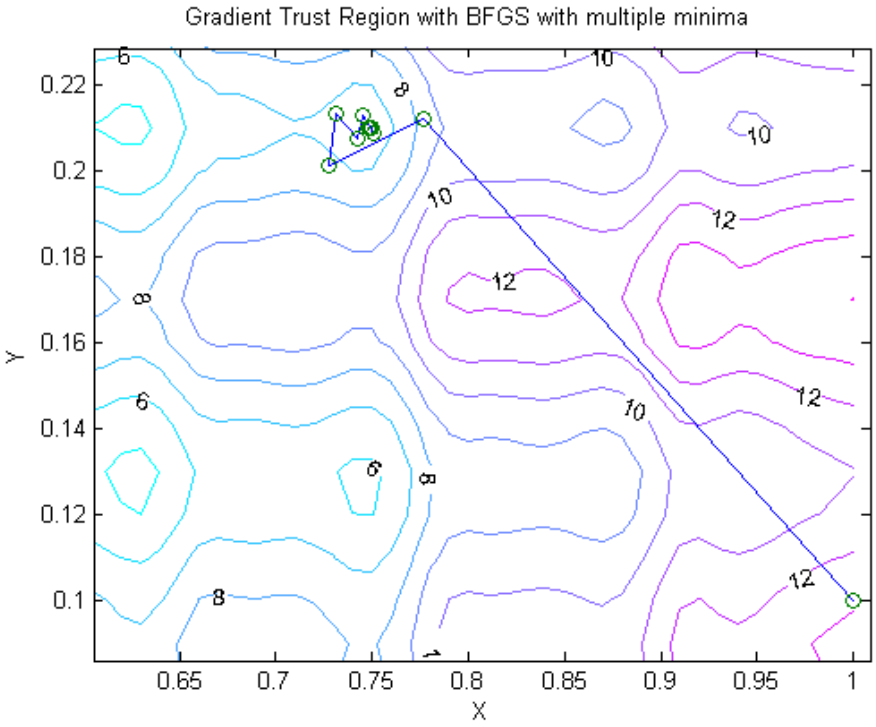

Figure 10: Gradient method with multiple minima

Considering Figure 9 and Figure 10 it is clear neither algorithm was useful in finding a global minimum. As presented here these algorithms had no checks for local verses global minima. These algorithms can be included in larger structures that do better with ignoring local minima but these require more complexity and more function evaluations than was initially deemed necessary for this project. This fault was found to limit both algorithms when applied to the dies and was addressed in later versions of the programs. At the trial state described above these algorithms show a clear ability to locate at least a local minimum

After the initial point was evaluated a forward difference discrete derivative would then be calculated. The discrete step size was set as a fixed percentage of the trust region with a lower limit enforced. When the calculated discrete step was smaller than a lower limit, a central difference derivative would be calculated with both forward and backward steps at the lower limit value. When the discrete gradient was being calculated the best objective function value was stored and compared to the central value. This best value was typically a very small improvement and would be stored for possible later use. The benefit of this small improvement was to add diversity in the derivative calculation which helped prevent stagnation. The model estimated the optimum, a vector Pb away from the current point where

$$\boldsymbol{Pb} * \boldsymbol{\beta} = -\boldsymbol{\nabla} \tag{6}$$

ß is the Hessian matrix and $\boldsymbol{\nabla}$ is the gradient. For the first iteration and when the Hessian was reset, ß would be set to the identity matrix and the steepest decent direction would be used. For subsequent iterations

$$\boldsymbol{\beta} = \boldsymbol{\beta}_{previous} - \frac{\boldsymbol{\beta}_{previous} * \boldsymbol{sk} * \boldsymbol{sk}^{T} * \boldsymbol{\beta}_{previous}}{\boldsymbol{sk}^{T} * \boldsymbol{\beta}_{previous} * \boldsymbol{sk}} + \frac{\boldsymbol{yk} * \boldsymbol{yk}^{T}}{\boldsymbol{yk}^{T} * \boldsymbol{sk}} \tag{7}$$

Where yk is the change in the gradient and sk is the change in the parameters.

If Pb was inside the trust region then it would be added to the current point and evaluated. If Pb lay outside the trust region the vector to the minimum along the direction of steepest decent, Pu, is calculated.

$$\boldsymbol{pu} = -\frac{\nabla * \nabla^T}{\nabla^T * \beta * \nabla} * \nabla \tag{8}$$

If Pu were outside the trust region, the direction of Pu would be taken to the edge of the trust region and evaluated. When Pu was inside the trust region a dog leg method was applied to find a point on the edge of the trust region which would then be evaluated. To find this step the parameter τ in Equation 9 needed to be determined. [12]

$$\boldsymbol{\Delta^2} = \sum_{i=1}^{n} \left| |\boldsymbol{Pu} + (\boldsymbol{\tau} - \mathbf{1}) * (\boldsymbol{Pb} - \boldsymbol{Pu})| \right|^2 \tag{9}$$

Where Δ is the radius of the trust region. This can be converted to a quadratic of the form

$$\boldsymbol{A\tau^2} + \boldsymbol{B\tau} + \boldsymbol{C} = \mathbf{0} \tag{10}$$

$$\boldsymbol{A} = \sum_{i=1}^{n} (\boldsymbol{P_i b} - \boldsymbol{P_i u})^2 \tag{11}$$

$$\boldsymbol{B} = \sum_{i=1}^{n} (\boldsymbol{P_i b} - \boldsymbol{P_i u}) * \boldsymbol{P_i u} * \mathbf{2} \tag{12}$$

$$\boldsymbol{C} = \sum_{i=1}^{n} \boldsymbol{P_i u^2} - \boldsymbol{\Delta^2} \tag{13}$$

And it can be shown that $\tau > 0$ when $\sum_{i=1}^{n} P_i u^2 < \Delta^2$ which was the requirement to perform the dog leg step. The step taken is then found to be

$$\boldsymbol{P = P_u + \tau * (P_b - Pu)}$$

(14)

The change in the objective function for this step in the modeled approximation was found by

$$\boldsymbol{\Delta_{model\ objective\ function} = \nabla^{T} * P + .5 * P^{T} * \beta * P}$$

(15)

After the step was taken, the new point would be evaluated. The resulting objective function value change is divided by $\Delta_{model\ objective\ function}$ to obtain ρ.

$$\rho = \frac{G(x)}{\Delta_{model\ objective\ function}}$$

(16)

This gives a measure of how well the model represents the actual function. When ρ is less than .25 or $\Delta_{model\ objective\ function}$ is positive, Δ would be reduced to a quarter its value. $\Delta_{model\ objective\ function}$ can become positive when ß is non-positive definite, so when this was found ß was returned to the identity matrix. The approximation if the Hessian is no longer accurate when ß is non-positive definite. If ρ was greater than .75 then Δ would be doubled or raised to the upper limit, whichever were the smallest. If ρ were less than η, the minimum acceptable value for ρ in optimization, the smaller of either the previous point or the best point from the last derivative would be returned and a new derivative would be calculated. In this case η is 1/32. If the new point from the step is an improvement the new gradient will be taken and the process repeated until a minimum difference between consecutive points was found.

If the new point was worse than the previous, a one dimensional model of the function in the direction of the step would be formulated. This model assumed the second derivative was a constant and would calculate the step size to the inflection point. This model used the value and slope at the current center, the distance of the

step and the value of the new point to formulate the model. If this point is an improvement it is accepted. This method has proven useful in dealing with steps that have significant penalty factors and reducing the step to the edge of the penalty. If this line search point was larger the program would either return to the previous point or the best point from the last gradient calculation. Then the gradient would be recalculated, normally with new discrete step sizes. At this stage there is a potential for an infinite loop when the discrete derivative step size is at the lower limit and all points used to calculate the discrete derivative are worse than the center point. This problem and issues with calculating an accurate derivative are addressed in the adaption for the flat die, where the trust region method was further refined.

Flat Die

Algorithm Adaptation for the Flat Die

Nelder Mead

A simple check was done on the flat die to avoid non-orthogonal problems. A design parameter in the worst point was offset a small amount when all the other points shared the same value for that design parameter. This would not detect diagonal planes in the design space but prevented the points from becoming planer due to a single design parameter's geometric limits. In this way it was not a complete orthaogonality test. The requirement for being orthogonal is a limit of the Nelder Mead algorithm. If all points are in a plane the algorithm loses the ability to leave this plane.

The Nelder Mead optimization algorithm used a weighting function for each design variable to reduce skewness in the design space. This was needed since a fixed change in different variables result in different scales of changes in the objective function. In the case of the flat die, a 1mm change in the land length was a much more significant change than a 1mm change in manifold depth. To correct this the

initial set of points were selected with less sensitive design parameters further apart and more sensitive design parameters closer together.

Trust Region

The trust region method was further developed for the flat die. These adaptations were not carried out with the spiral mandrel die. Initially there were no weighting on the design parameters. Unfortunately this leaves the trust region and discreet derivative equidistant for all parameters. The problems this caused for the discreet derivative as well as other issues with the original algorithm were addressed in the following adaptations.

To improve the accuracy of the discrete derivative, in the adaptation for the flat die, the uniform step size was replaced with an independent and adaptable scheme. A target value for the change in objective function value controlled the discreet derivative step size for each parameter. If the actual change were greater than 110% of the target, the step size for that parameter was reduced to three quarters of the current value. If less than 90% the step size was doubled. Further, every evaluation of the derivative changed between forward difference and backward difference to increase the diversity of points evaluated. The choice of central or single sided discrete derivatives was no longer determined by the trust region size. In this new version a single sided derivative was attempted first and if it failed the other half of the central derivative was evaluated and included. A central derivative was done immediately when it was preceded by a failed step based on another central derivative.

A final adaptation was used to reduce the effect of a value in a central derivative that was expected to be inaccurate. When forward and backward values are both points worse than the central point, the derivative value is reduced by a factor of the smallest single design parameter derivative divided by the largest. This did not affect the results when all points were worse than the center and focused the movement in directions of known improvement.

The infinite loop problem mentioned above could happen when the current iteration was in a local minima, often created by noise in the system. This was addressed by increasing the discrete derivative step size when a central difference approximation was used and all values were worse than the center value. To balance this, whenever a central difference discrete derivative is used and values are found that are better than at the center, the discrete step sizes are reduced, with a lower limit enforced. This resulted in new points being evaluated each time until an improved point was found or the step size reached the upper limit, where the program would end. While this did not guarantee avoidance of local minima it has been very useful for distinguishing local minima due to noise. To further limit noise based local minima an evaluation was made after both the normal step and the short line search style evaluation failed and the trust region was smaller than the average discrete derivative step. Here a step is taken in the same direction as both previous steps but at the length of the average discrete derivative step. When this larger step is successful the trust region is reset to this length. This prevents the trust region from spending too much time making steps on the scale of the noise as opposed to the true model contour.

Geometry Description

The flat die produces a rectangular sheet or film. The polymer enters the die through a centered inlet channel, in this case with an elliptical cross section with major and minor axis's listed in Table 2 as major and minor. This allows circular and elliptical inlets to be used with the same code. The inlet channel length is defined by IL (inlet length) and blends into the manifold with a fillet, where the radius is listed as a parameter. The manifold is a large transverse channel that distributes the flow across the die. It is tapered towards the outlet in the x z plane and narrows in the y direction towards the edges. The angle towards the outlet is controlled by MBAXZ (manifold base angle in the x z plane) on the back wall of the die and PCL (pre-land center length) in the front. The manifold depth is at the maximum at MCD (manifold center depth) and decreases towards the edges at an angle of MBAXY (manifold base angle in the x y plane). At the end of the manifold

the back edge curves forward to prevent long residence times in a corner. The length and radius of this bend is controlled by MESL (manifold end sweep length). The initial manifold thickness is defined by MCFL (manifold center flat length). After the manifold, there is a relatively narrow channel, called the secondary manifold. It is connected to the manifold by a sloped surface at an angle equal to half MA (manifold angle) and is meant to stop the flow until the entire manifold is filled. The secondary manifold and the rest of the die down stream are the width of the final shape, DW (die width). After initial filling, the secondary manifold improves flow due to its relatively high resistance. It forces the flow down the manifold to the edges of the die. The dimensions SMD (secondary manifold depth) and SMCL (secondary manifold side length) will control this resistance. At the end of the secondary manifold is the land. The opening thickness in the land (LG) is normally the thickness of the desired sheet or film, although this may be slightly smaller to account for die swell. The land also forces the flow to be more even through resistance. This resistance is adjusted by the land length (dimension LL). The parameters listed are all modifiable at the start of the program but only the six parameters listed in Table 1 are considered in optimization. These parameters were found in previous studies to have larger influence on the velocity distribution and pressure [Sun Yong].

Table 1: Variables for the flat die

Parameter	Initial value
PCL (mm)	80
SMCL (mm)	20
SMD (mm)	20
LL (mm)	50
MBAXY (radians)	.492 π
MCD (mm)	64.6

Table 2: Flat die constant parameters

Parameter	Value	Parameter	Value
Major (mm)	40	Minor (mm)	20
Radius, inlet to manifold (mm)	7	IL (mm)	100
DW (mm)	1500	IW (mm)	100
MCD (mm)	60	MCFL (mm)	12
MSFL (mm)	10	LG (mm)	9
MBAXZ (radians)	0.4831 π	MA (radians)	0.3332 π
SMA (radians)	0.3332 π	Volumetric flow rate (m^3/s)	.00335
Inlet Temperature (K)	500	Wall Temperature (K)	450

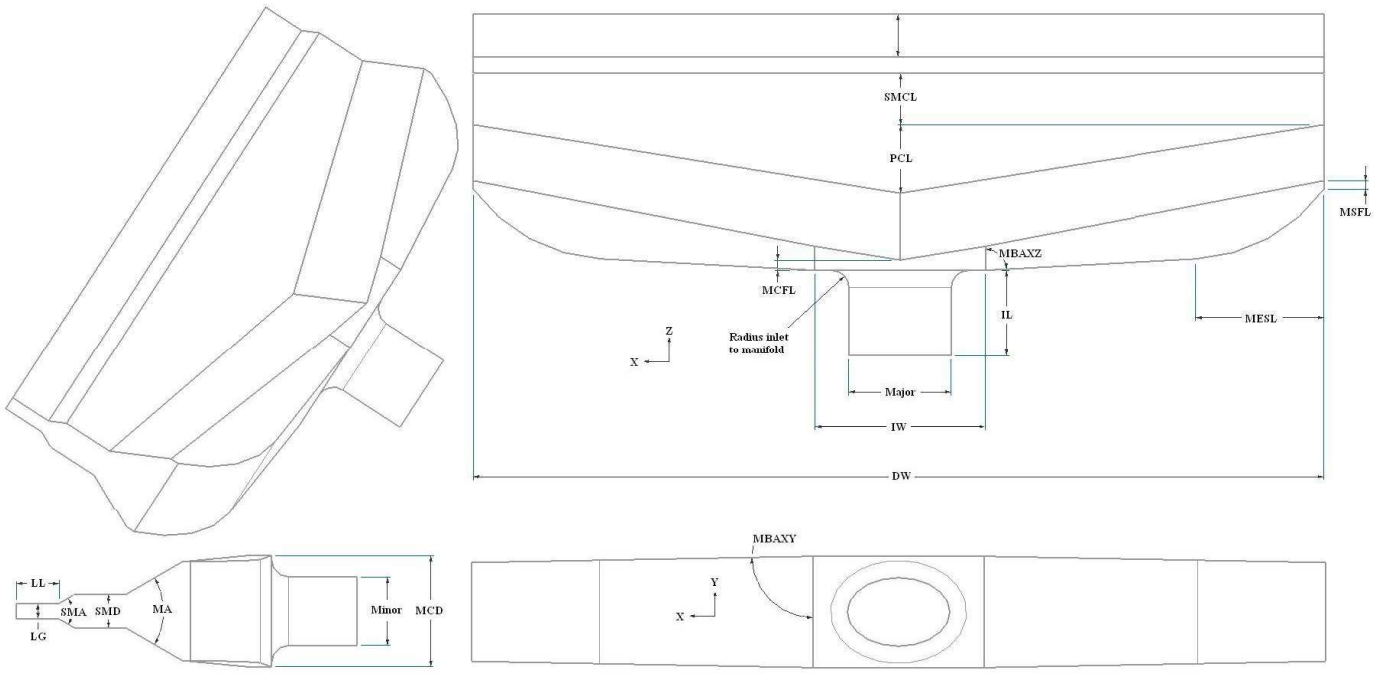

Figure 11: Drawing flat die

Parasolid

The Parasolid file format is a text based universal CAD file format used to describe a given geometry. Parasolid files are commonly used to transfer models between different geometric modeling programs. In this work, a parasolid file describing the die was re-written to describe each new design. Then it was used by Simmetrix software to generate a mesh. A parasolid file starts with a header section that logs information such as modeling program used, authors' user name, date and time. Information could be added to the header without affecting later parsing functions. In this case the design parameters are added to the header file so that a user could open a previously used parasolid file and retrieve the dimensions. Following the header is a long set of numbers and letters that contain the information describing the part. In this data set geometric information defining points, edges and surfaces, as well as topological relations between these features are listed. An entry will start with a number identifying what will be described next. Following this number will be the number 255, if this is the first entry of this type, followed by topological

information. At the end of the entry, if the entry describes geometric information, values relating dimensions will be listed.

For example an entry to describe a vertex from the flat die in the code used in this project was "29 43 249 0 45 46 40 0 .5*MCD MCFL". The 29 identified this entry as a vertex. The lack of a 255 means this was not the first vertex in the list. The values 43, 249, 0, 45, 46, and 40 relate the topological information for this point, that is which edges and surfaces it was connected to. The last three numbers described the coordinate; x=0, y=.5*MCD and z=MCFL. This vertex was on the top of the die, in the center of the manifold. Straight and circular edges were easily described in a similar manor. Surfaces that were flat, cylindrical and conical were also easily described. More complex edges and surfaces use B-splines and other geometric entities.

The relatively simple flat die used around 150 entries for geometric information and many more for topological relations. Topological relations were not changed, so new surfaces or edges could not be created or removed. This method's ability to consistently generate results for the flat die will be discussed next.

Meshing

The parasolid file could be generated for any valid set of parameters and using Simmetrix, a new mesh could automatically be generated. Every mesh with new dimensions would be slightly different than the previous. These differences would have some effect on the final results. This randomness will be referred to as noise. To understand the magnitude of the noise in the system a test group of evaluations were run where each new point was modified by a small and equal step. In a system without any noise very small changes made in a linear fashion would produce a smooth change in results. Figure 12 is a portion of the study that clearly shows the noise effects. The X axis in this plot is the total change in the six parameters from the original point as measured in six dimensional space. In this plot the whole span is 1 mm, this would correspond to 0.2 mm change in each dimension. The tight cluster of points are grouped ten times closer and span 0.1 mm across the group or

0.02 mm in each dimension. The Y axis is the objective function used to measure the performance of the die. Values tend to vary by up to 0.2 in the objective function due to noise. Later in the study this information on the noise was required to set a target value for the change in the objective function when calculating a discreet derivative. This degree of noise is considered manageable and was able to be ignored with relatively few added evaluations.

Also in Figure 12, there are three data sets. These were arranged to compare the effects of mesh density on the solution. The number of layers refers to the number of elements across the depth direction of the die, that is in the direction of MCD. While increasing mesh density does not produce a converging pattern the trends are very consistent. It is expected that the optimum point for all three meshes would be approximately equal, as the slopes were well matched. The six layer mesh was selected as it was expected to have the least amount of noise. The four layer mesh was also tested in optimization for comparison.

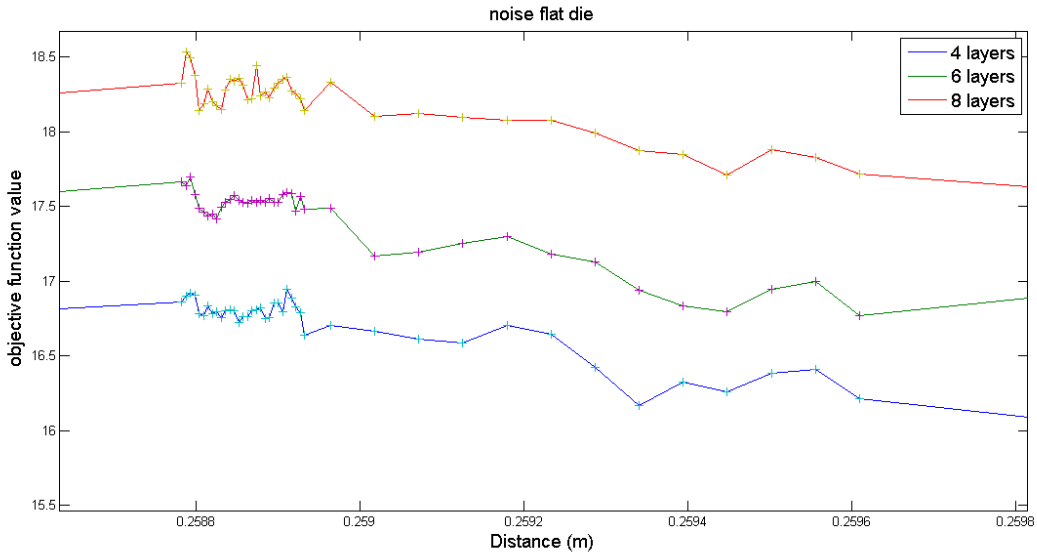

Figure 12: Noise Study for 4, 6 and 8 layered meshes

Results

Flat Die with Nelder Mead

Both the four and six layer meshes were tested with the Nelder Mead algorithm. Both meshes were unsuccessful at preventing stagnation, see Figure 13. The six layer mesh was completely stagnated by the 50th evaluation. The four layer mesh progressed further but stagnated at an objective function value of 13.5 mm/s. In earlier versions of the program, which were less stable, the four and six layer meshes performed comparably. It is expected the difference in performance on this version is coincidental and may turn out differently with a different original design. It is clear the Nelder Mead algorithm as used here was ineffective at preventing stagnation.

As expected the algorithm was successful in maintaining the pressure near the limit. Both versions of the Nelder Mead algorithm were tested using the in-equality pressure penalty.

Earlier versions were developed with weighted and non-weighted means and shown relatively little difference in performance. This algorithm did not include a function for re-expanding once inside a local minima, either created by noise or a true contour of the objective function. With such a function this algorithm may have been able to find a solution. Further development was not attempted as the trust region algorithm was identified as the faster method.

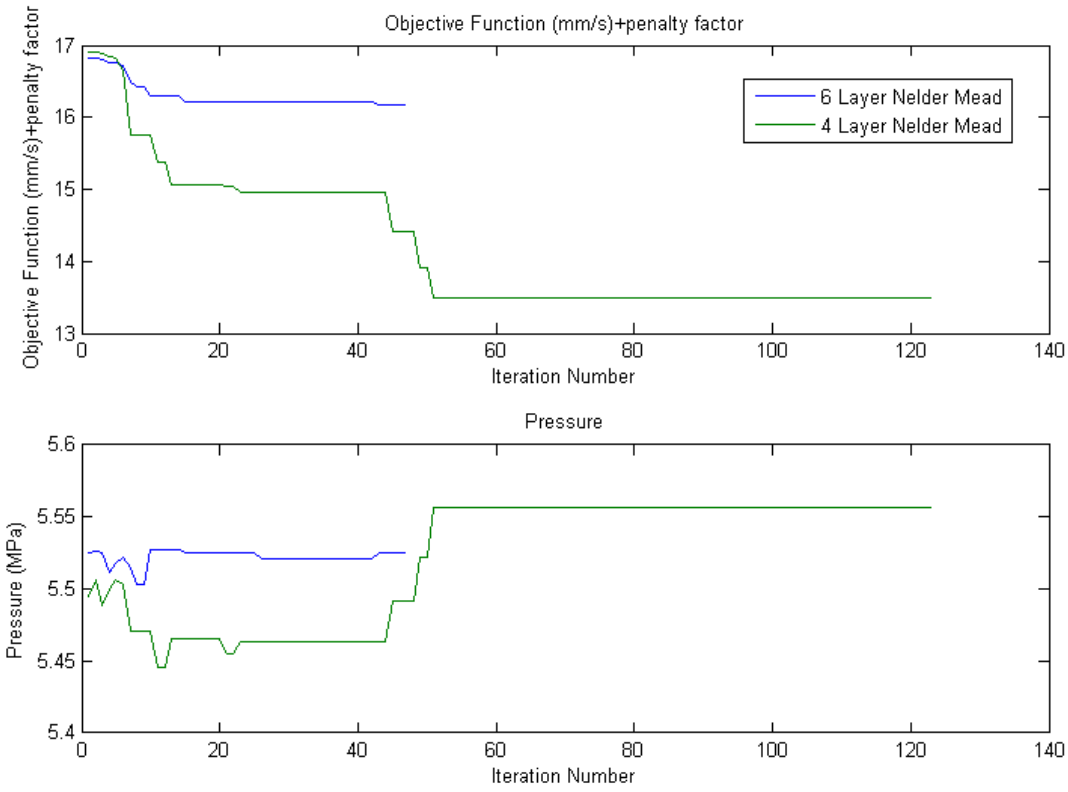

Figure 13: Convergence for the Nelder Mead Algorithm on the flat die

Flat Die with Trust Region Optimization

Program Inputs

The parameters that define the die geometry were listed at the start of the program. The user supplied the program with the relevant geometric dimensions to describe the initial design of the die. CFD boundary conditions and options were supplied at the same time. These included the inlet and wall temperatures, volumetric flow rate, global mesh size and number of element layers across the thickness of the die. The current code allowed the option to reuse the dimensions and settings from the previous iteration to save time. After the relevant parameters were selected, the user was prompted to input a maximum allowable pressure or allow the program to use the pressure from the initial evaluation as the limit. A material data file was read separately and was generated using PolyXtrue software. [3] This material file

was simply saved in the same directory as the optimization code and read during operation.

Processing Time

Processing time is highly dependent on initial design and at this point in development may still be improved by tuning several optimization parameters. Generally a very good initial design would process for 2 days (48 hours) . The program would reach the optimum after a day and spend the second day confirming the design is not located in a local minimum. Less accurate initial designs, which are discussed here, took around 2 weeks.

Program Outputs

The program script directly reports the parameters being optimized and objective function value but more useful to the user are the parasolid model and CFD simulation for the optimum design. Also available during processing was a log of all designs evaluated, log of optimization activities and separate log of derivative function calls.

Convergence Rate

The flat die was able to generate new meshes and was selected for further investigation with the second version of the trust region method. The first version took 505 and 550 evaluations for the in-equality and equality penalty factors respectively. In an attempt to improve performance, the discreet derivative step size was reduced by 25% rather than 55%, when the function contour required reductions of the discreet derivative step size. Also the minimum discreet derivative target value and initial value were both set to .25 rather than .3 and .2 respectively. The result were reductions to 344 evaluations for the in-equality penalty and 378 evaluations for the equality penalty. It can be seen in Figure 14 and Figure 15, the original design is quickly improved at the early stages of the program. This quick pace was due to the contour of the objective function being steep. The flatter sections were then where the code had more difficulty and needed to adjust the controlling parameters. For example, between the 150th and the 200th evaluation of

the first version, the trust region was reduced due to poor steps. These relatively short steps had difficulty with noise and the trust region was further reduced. To prevent this from ending the program the algorithm took a step much larger than the trust region, based on the step size being used for the discreet derivative. When this returned an improved objective function value the trust region was redefined to this larger size. The code was successful at avoiding stagnating in local minima. The second version, both equality and in-equality penalties managed to move from an objective function value of about three, where there is an expected local minima, to the final values near one. When all step types failed to find an improved design the code completed the other half of a central difference discreet derivative. When all these discreet derivative points were evaluated as worse than the central point, it was expected that the current point was inside a local optimum. The target change in objective function value for the discreet derivative was increased until an improved objective function value was found. Naturally this method repeats at the global optimum where about 100 evaluations are used to check the surrounding design space. When the change in objective function target value increases beyond a limit the program ends.

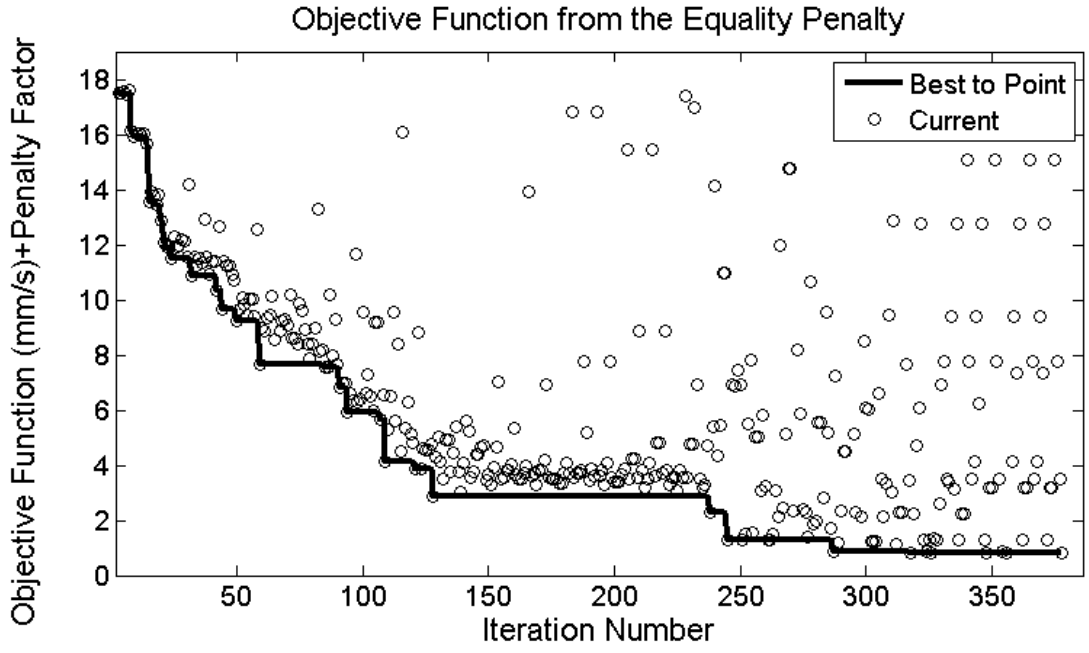

Figure 14: Improvement of objective function throughout Equality Penalty Trust Region Optimization

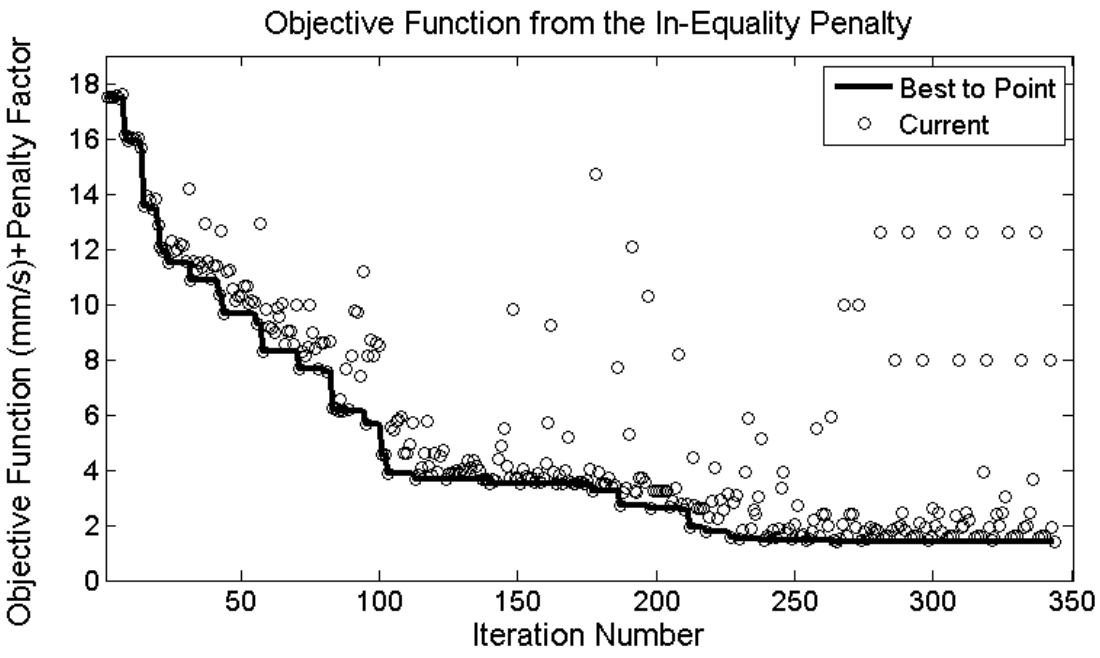

Figure 15: Improvement of objective function throughout In-Equality Penalty Trust Region Optimization

Throughout the optimization the equality penalty held the pressure to within +/- 3% of the target value, 5.5 MPa. Details on the development of the pressure term during optimizations can be found in Figure 40 and Figure 41 in the appendix. A potential for improved processing time is evident from Figure 14 and Figure 15. Between the 275th and 350th evaluation 2 rows of solutions are found at objective function values of 8 and 13. These values represent identical designs that were re-evaluated as the algorithm searched the area around the optimum for better solutions. These were the result of the system calculating the discreet derivative while a given parameter was at the upper limit value. Processing time could be saved by logging all evaluations and taking the objective function from the first evaluation in cases of repeat design calls. In Figure 16 and Figure 17, the repeated evaluations are also seen between the 275th and 350th evaluation as rows of constant values. Because these plots separate the evaluations by design parameter the repetition can be seen more clearly. Considering Figure 16 and Figure 17 around 8% or 30 evaluations could be avoided.

Optimized Design

Figure 16 and Figure 17 illustrate the changes in the dimension parameters. The angle MBAXY is not considered but it's complement is used for this plot. This is done to make the percentage change legible. Similar to Yong Sun's work, the land length (LL) was decreased when generally designers would increase this dimension to improve velocity distribution. In this work we see that the LL initially increases before dropping below the original value. Other reversing trends are seen in SMCL in Figure 16 and in SMD Figure 17. These trends would be difficult for a designer to predict. The increases in MCD and PCL are expected in poorly performing dies, as these increase the manifolds ability to move polymer to the edges of the die. Additional plots expanding the data in Figure 16 and Figure 17 can be found in the appendix in Figure 42 and Figure 43. These figures show MBAXY and MCD from the inequality solution to continue to 5.1 and 3.5 times their original value and for both solutions values for SMCL below 0. The negative values for SMCL were not considered in the algorithm but were attempted due to a programming mistake.

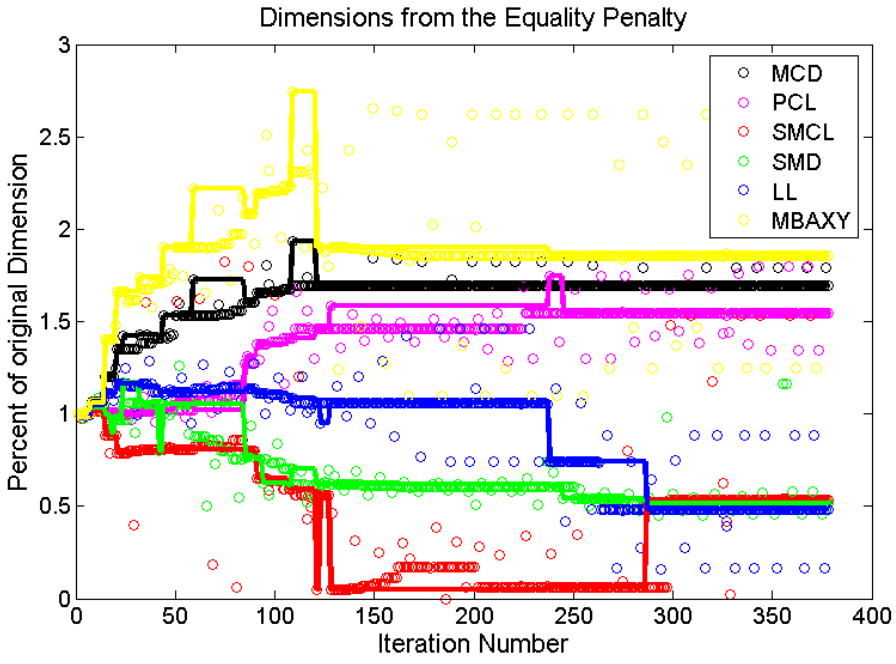

Figure 16: Dimensions throughout Equality Penalty Trust Region Optimization

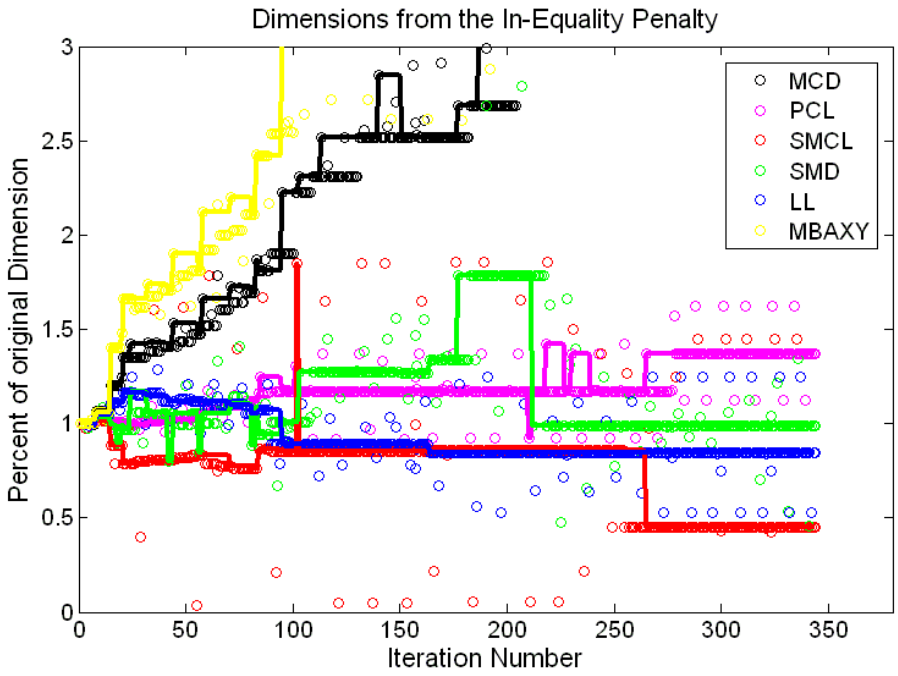

Figure 17: Dimensions throughout In-Equality Penalty Trust Region Optimization

Considering Table 3, two sets of equality and inequality optimizations were executed. The second version improved convergence rates by adjusting several optimization parameters. The first version solutions were near the same optimum. They differ primarily by SMCL and LL but have a very similar objective function values, 0.720 and 0.723 . The second set of solutions did not converge to the same optimum. The equality penalty seems to have approached the optimum located by the first set while the inequality, which was free to explore the designs with lower pressures, located a design optimum with a very large primary manifold volume. In this case the inequality solution would have a higher potential for residence time problems. A visual comparison the original die and the second version solutions are shown in Figure 18 through Figure 20.

Table 3: Initial and Final Dimensions

Parameter	Initial value	1st Version Equality Penalty	1st Version In-Equality Penalty	2nd Version Equality Penalty	2nd Version In-Equality Penalty
PCL (mm)	80	136	131	124	110
SMCL (mm)	20	13.5	23.7	10.7	8.97
SMD (mm)	20	12.2	12.1	10.3	19.7
LL (mm)	50	34.4	26.9	24.2	42.4
MBAXY (radians)	.492 **π**	.486 **π**	.486 **π**	.485 **π**	.457 **π**
MCD (mm)	64.6	112	116	102	211
Evaluations	0	547	505	378	344
Objective Function Value	17	.720	.724	.811	1.41

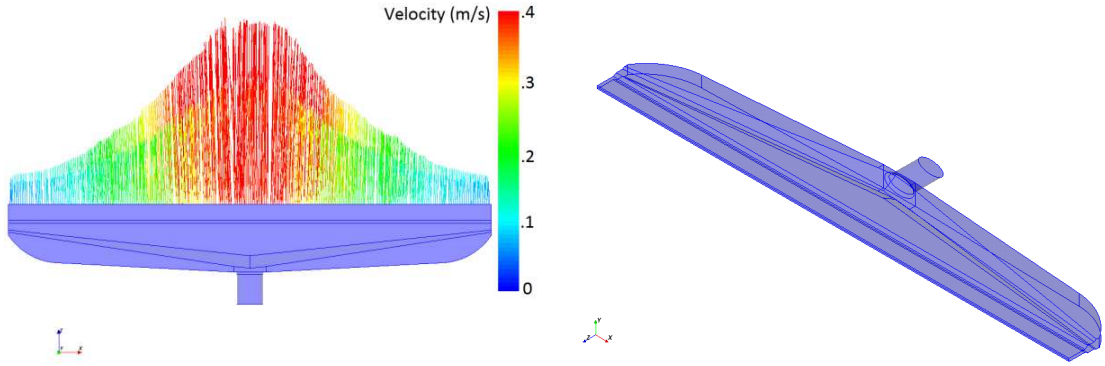

Figure 18: Original Design Velocity Vector plot at Exit and Original Die Model

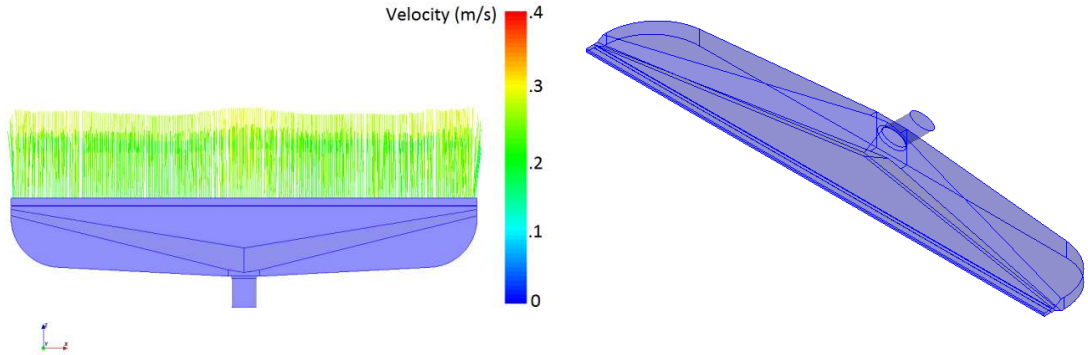

Figure 19: Second Solution set with Equality Penalty

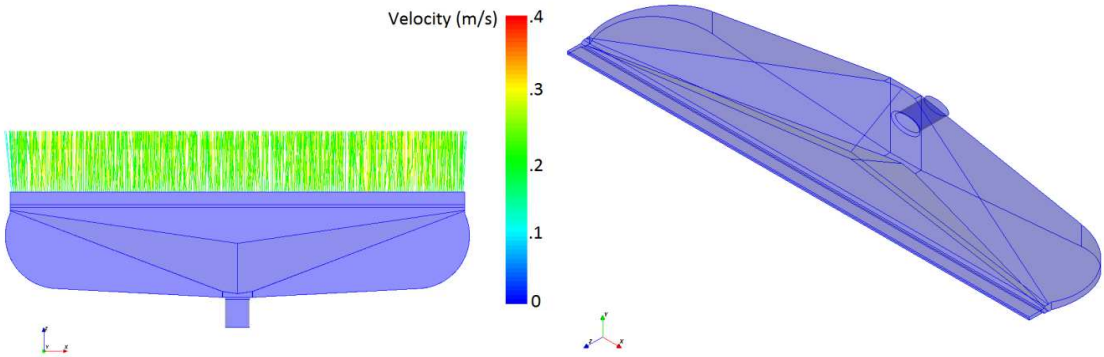

Figure 20: Second Solution set with In-Equality Penalty

Comparing the original design and the optimized illustrates how much of a difference is possible with two similar dies, Figure 18 and Figure 19. Figure 19 and Figure 20 are relatively different dies but perform similarly. The original die, seen in Figure 21 was six times faster in the center than the sides and due to the un-even viscous heating has an un-even temperature profile. The peaks at the edges of the original dies temperature profile are due to increased shear stress from the sides of the die. The optimized dies on the other hand have a very flat velocity distribution and from this a reasonably flat temperature distribution. Comparing the die models in Figure 18, Figure 19 and Figure 20, it is seen that the optimized dies are longer overall and the manifolds have greater volume. These larger manifolds may negatively affect the residence time distribution as Smith had predicted in an earlier study. Additional plots may be found in the appendix, Figure 38 and Figure 39, detailing the first solution set.

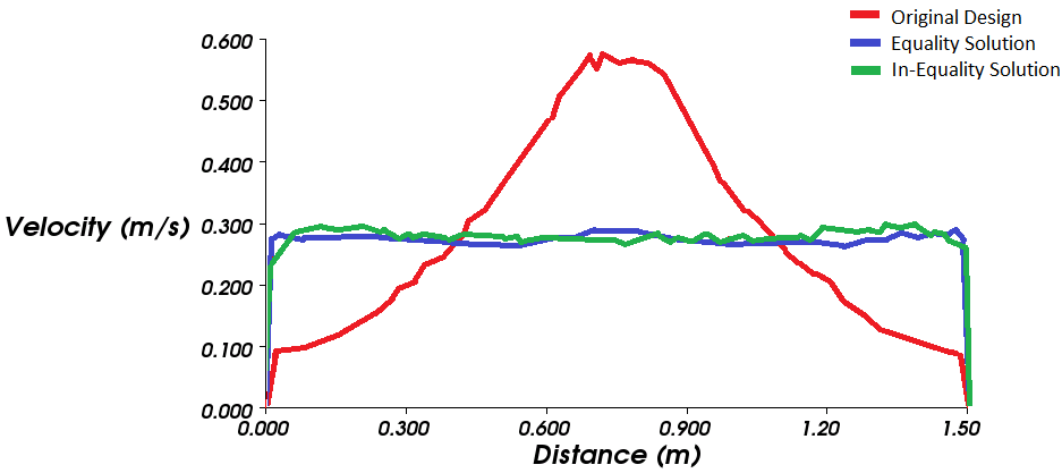

Figure 21: Velocity Distribution across Die Exit, from the second solution set

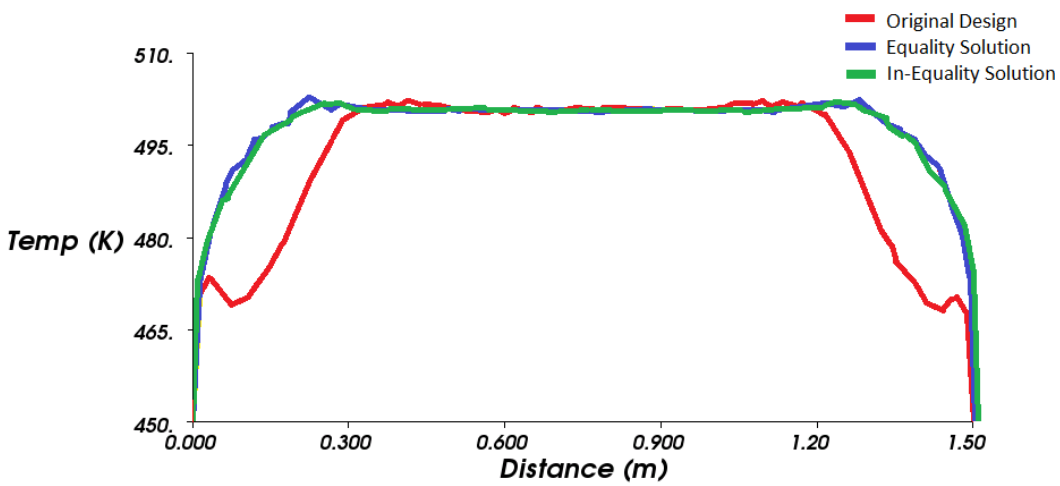

Figure 22: Temperature Distribution across Die Exit, from the second solution set

Spiral Mandrel Die

Algorithm Adaptation for the Spiral Mandrel Die

Nelder Mead

In the constriction step of the spiral mandrel die, a check was done to see if all points were co-planner and non-orthogonal. If it is found that all the points were co-planer the second best point would be offset orthogonally from the plane by a small amount. This was necessary because the limit between the helix section length and the lobe radius formed a diagonal planer limit in the design space. This is a limit of the Nelder Mead algorithm. If all points are in a plane the algorithm loses the ability to leave this plane.

The Nelder Mead optimization algorithm useed a weighting function for each design variable to reduce skewness in the design space. This was needed since a fixed change in different variables result in different scales of changes in the objective function. In the spiral mandrel die the lobe radius was increased by ten times when in the design space. The initial three off-set points were a fixed value from the center point in the design space.

Trust Region

The spiral mandrel die continued to increase the lobe radius by a factor of ten in the design space. Otherwise the trust region code runs as described in the numerical optimization section.

Geometry Description

The spiral mandrel die (Figure 23) produces a pipe or hollow cylinder shape. In NX the die was defined by the outside diameter at the exit (OD), the inside diameter at the exit (ID), the lobe radius, the number of turns in the helix, the length of spiral section and the length after spiral section. In this variation, the cross section of the four inlet channels were defined by an inside and an outside arc. The inside arch was the lobe radius and the outside was the outside radius of the die, one half the OD. The length of the inlet channels increased proportionally with changes in the OD. The inlet channel then arcs to connect to the helix section with the bend radius defined below.

$$\boldsymbol{spiral_{angle}} = \mathbf{arctan}\left(\frac{\boldsymbol{\pi * OD * (number_{turns})}}{\boldsymbol{length_{helix}}}\right) \quad \textbf{(17)}$$

$$\boldsymbol{radius\ of\ bend} = \frac{\boldsymbol{(offset) * \left(\frac{OD}{OD_{original}}\right)}}{\mathbf{sin}(\boldsymbol{spiral_{angle}})} \quad \textbf{(18)}$$

Where $offset$ is the original offset between the end of the inlet channel and the beginning of the helix. This was then scaled with changes in the OD. The four channels followed a helical path around the mandrel starting with a helical radius at the part OD and linearly increased towards the end of the die. The final diameter was a function of lobe radius, OD, ID and length of lobes. The angle of expansion for the helix radius was defined by $\boldsymbol{\theta_1}$ and $\boldsymbol{\theta_2}$.

$$\boldsymbol{\theta_1} = \boldsymbol{arcsin}\left(\frac{\boldsymbol{lobe_{radius} * cos(\theta_2)}}{\boldsymbol{length_{helix}}}\right) \quad \textbf{(19)}$$

$$\theta_2 = arctan\left(\frac{wall\ thickness}{length_{after\ helix}+length_{helix}}\right)$$

(20)

$$final\ diameter\ of\ helix = \left(\left(\frac{OD}{2}\right) + tan(\theta_1 - \theta_2) * (length_{helix})\right)$$

(21)

The mandrel forms the inside wall of the cavity and was initially the diameter of the part OD so that the entire flow was in the channels. It then linearly reduced to the part ID.

To aid in development a template was made in Unigraphics so that the spiral mandrel die model was fully defined by 5 parameters. All other dimensions in the model were automatically updated relative to these 5 parameters.

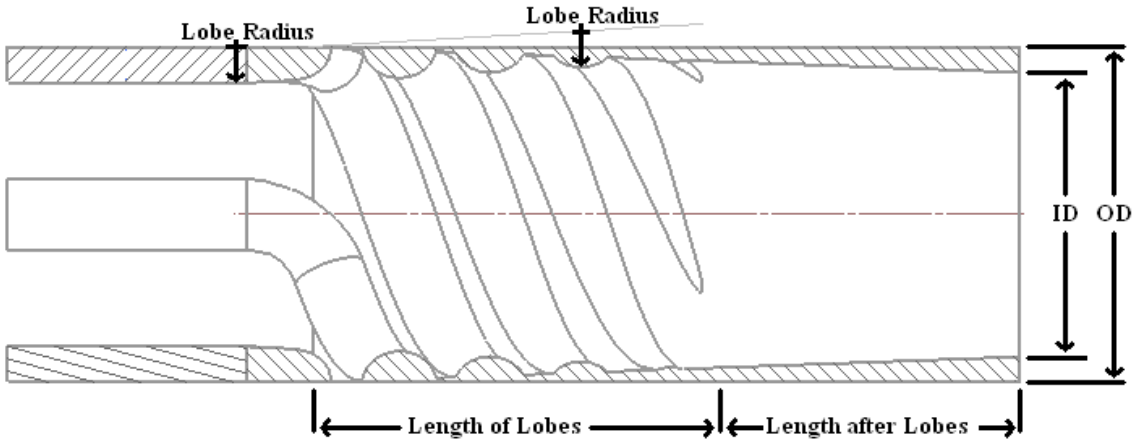

Figure 23: Spiral Mandrel Die Cross Section Veiw

Parasolid

Development of a code for automatic generation of a parasolid model for a spiral mandrel die was attempted but was not successful. The limiting factor was in the definition of the spiral sections. Parasolid uses B-splines to define the surface and edges of the spirals. The documentation that was available details the general form of how the points were weighted together to form the surface but does not include the B-spline basis functions. In an attempt to replicate the changes in a B-spline

caused by a change in lobe radius, the movements of the points in the B-spline were investigated by creating three different models with varying lobe radiuses. The points which define the surface of the spiral did not follow the surface but instead used the weighting factors included in the parasolid data. The movements of these nodes were easily tracked and could be replicated. Next, the intersection edge between the lobe surface and the ID conical surface would need to be updated. This was unsuccessful due to the surface defined by the B-spline being an approximation of the intended surface. Details on these difficulties are covered in the appendix. This inability to regenerate a parasolid file with new dimensions prompted another approach, which is discussed in the next section.

Mesh skewing

Since automatic generation of the parasolid files for spiral mandrel dies was not successful a mesh skewing approach was attempted for minor modifications of the die geometry. By changing the coordinates of the nodes in a previously generated finite element mesh a new die geometry could be created. This new mesh would have elements with higher aspect ratios and less uniform growth rates but these effects were expected to be small for minor changes in the die geometry. When moving nodes in the finite element mesh it was important to accurately represent the die geometry. The main cylindrical channel needed to expand linearly to maintain a conical inside wall. The spiral channel also was required to remain as a circular cross section in the direction of the helix. The primary limitation to how far a mesh could be skewed with this program was the movement of the elements between the channels. The requirement that the distribution channels had to maintain an aspect ratio forced significant element skewing in between the channels.

In practice, skewing of the mesh to produce geometric change did not work in a standalone optimization program. To illustrate the difficulties in this approach three different meshes are compared in their ability to simulate flow in dies with the same dimensions. In Figure 24 (a) multiple points were evaluated between mesh A, the original mesh, and mesh C, a similar mesh generated later. Mesh B was also

generated with an undistorted state half way between the two meshes to be used for comparison. The undistorted dimensions for the meshes are listed in Table 4. All dies used an inlet diameter of 23.75 mm and an outside diameter of 28 mm. The meshes all had minor differences but were generally able to predict the same slope of the objective function. In Figure 24 (b) the skewed mesh calculated the objective function with minimal noise when very close to the undistorted mesh but as seen in Figure 24 (c), too far from the undistorted mesh the noise would corrupt most discrete gradient calculations.

Table 4: Variable Parameters for undistorted meshes

	Lobe Radius	Helix Section Length	Length after Helix Section
Mesh A	3 mm	56 mm	16 mm
Mesh B	3.25 mm	48 mm	22 mm
Mesh C	3.5 mm	40 mm	28 mm
Mesh D	3 mm	33 mm	26 mm
Mesh E	5.5 mm	70 mm	40 mm

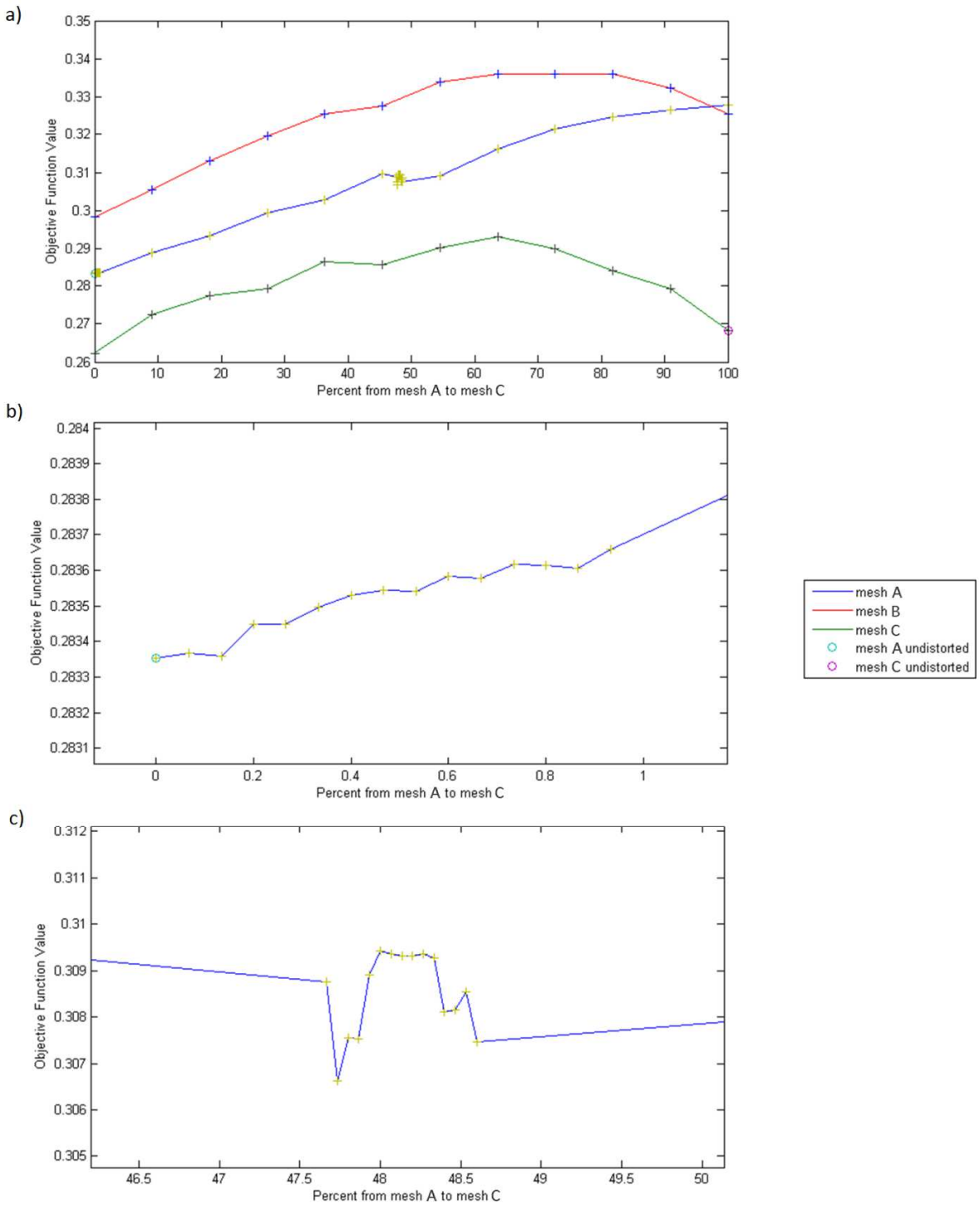

Figure 24: Distortion effects on Objective function (a), minimal noise near undistorted die design (b) and significant noise further from undistorted die design (c)

Based on this, optimization programs were run on a given mesh until it was expected that the mesh was out of the region that it would work to an acceptable degree of accuracy. At this point a new mesh would be manually generated and the optimization would continue. An attempt was made to create a data base of meshes so that the program could automatically load an appropriate mesh. Using this method and starting with mesh A, a set of meshes were worked though until reaching mesh D, listed in Table 4. The noise near the point of convergence can be seen in Figure 25. Here four off-set levels seem to be present. This noise prevented different optimization schemes to converge to the same point. It was expected that the differences would be caused by a non-continuous change between the meshes.

As the node coordinates are modified to generate new die geometries, the finite elements can be skewed in a way to create an inverted element. The program will check each element and correct the inversion by moving a node. This correction creates non-continuous change as the die geometry is continuously changed. For instance, in a die geometry two nodes were corrected, nodes 1196 and 8589. Node 1196 was changed the same amount for all points but node 8589 was change at four different levels. However, it was not expected that this was the cause of the noise. When the amount of change was color coded by degree of correction and plotted it was seen that amount of correction was not correlated to the off-set level (Figure 25). This un-identified noise could not be resolved in this work. Possible sources of noise include; the objective function calculation and variation in the node group used in objective function calculation, cumulative truncation error or non convergent solutions which are forced to stop at 25 iterations.

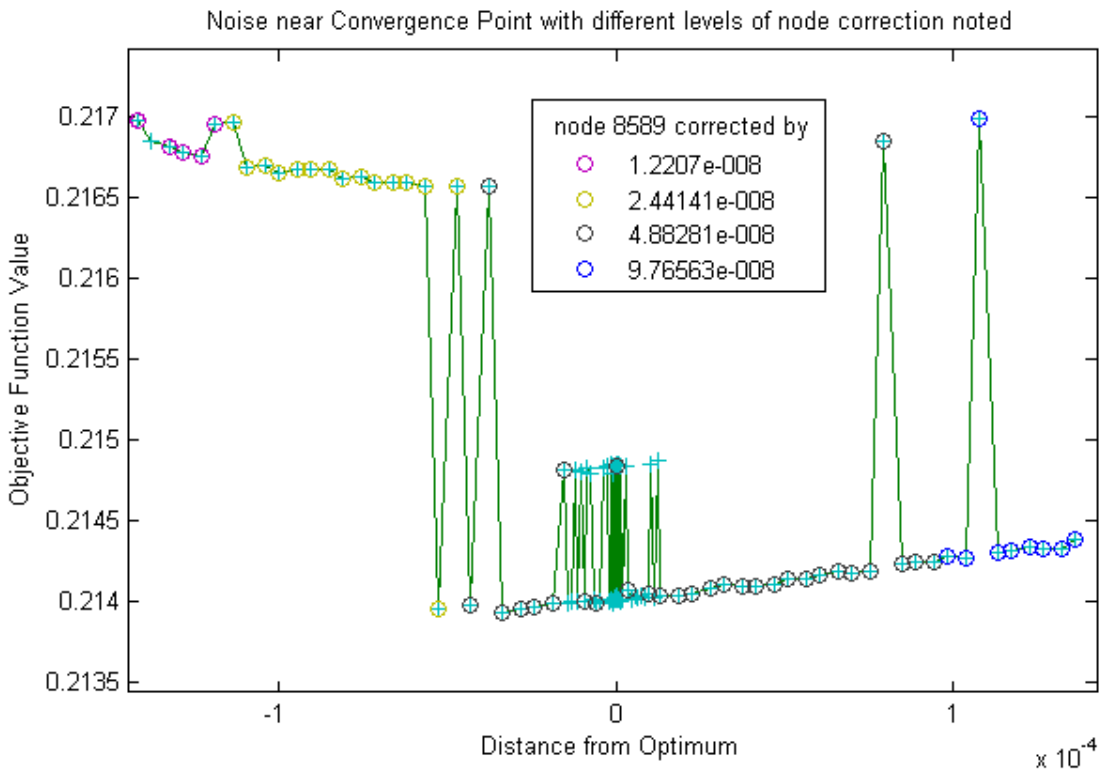

Figure 25: Noise near Optimum found using trust region method, with the correction to node 8589 noted in meters.

Results

Using the programs as described above optimization of the spiral mandrel die was attempted. The initial condition was the undistorted state of mesh A and progressed through a series of meshes to mesh D, detailed in Table 4 above. The velocity distribution measured halfway between the OD and ID around the exit can be seen in Figure 26 and Figure 27. The original design and the solutions are similar and relatively good. The original varies by about +/- 5% and the new solutions slightly less. It can also be noted that the average center line velocity is higher in all optimized designs. This was not desired but since it was not accounted for in the objective function the solver had no means of detecting it.

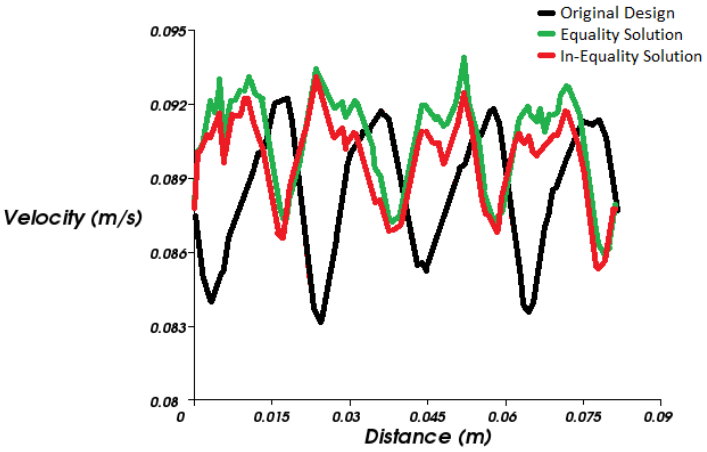

Figure 26: Velocity Distribution across Die Exit for Trust Region Solutions to the Spiral Mandrel Die.

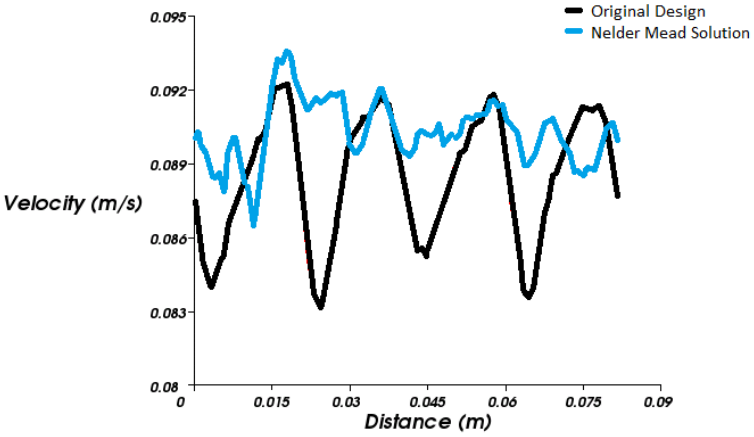

Figure 27: Velocity Distribution across Die Exit for Nelder Mead Solution to the Spiral Mandrel Die.

The initial design, mesh A was selected as the initial design because it was expected to be near an optimum and would require limited distortion to reach the optimal design. An initial design with expected poor performance may have shown greater improvement but would become cumbersome when using the mesh skewing method, as greater distortions would be required. In Figure 28 the original design and modified die geometry are compared. Even this relatively small change required ten meshes for the algorithm to search for the solution.

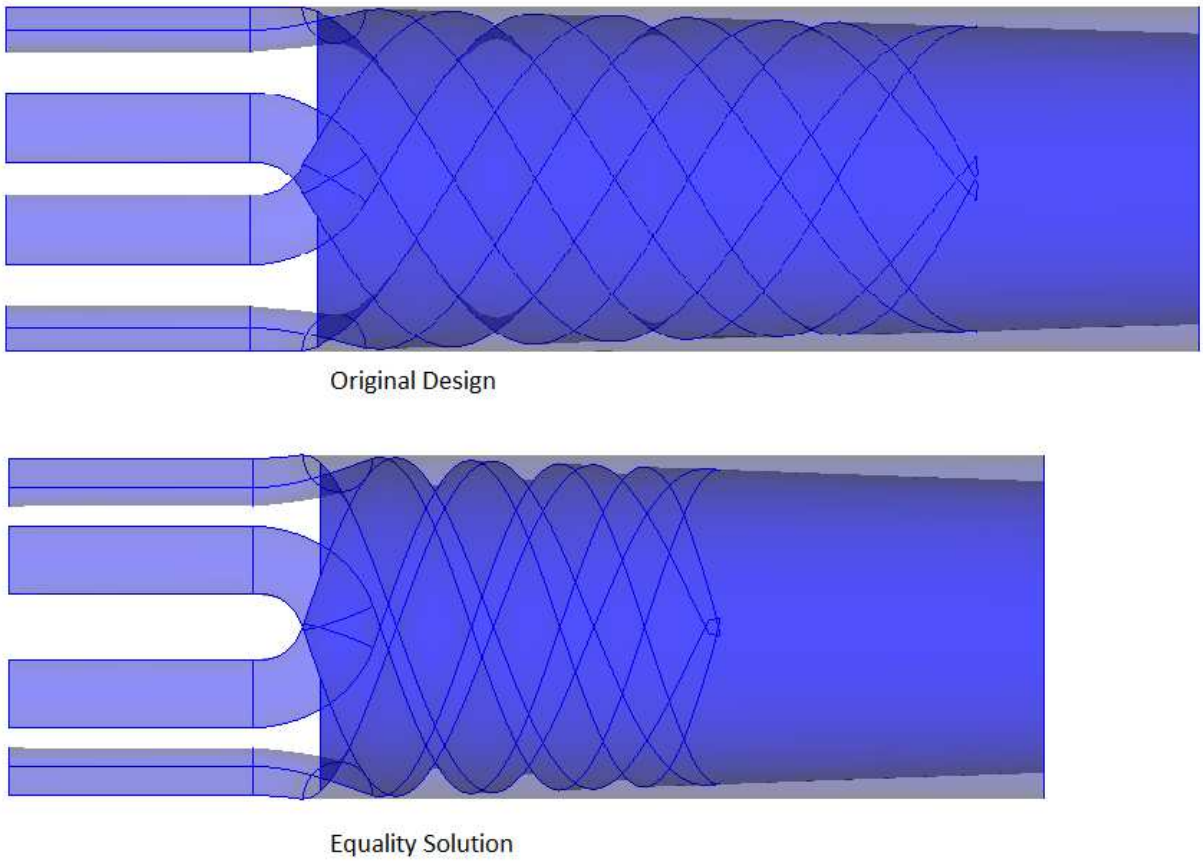

Figure 28: Comparison of the Original Design and the Solution from the Trust Region Method using the Equality Penalty.

Figure 29 shows the progress of three optimization schemes when mesh D was used. The Nelder Mead method returned values much better than the trust region gradient methods but the better results from Nelder Mead method was probably due to inaccuracy in the mesh resulting from skewing effects. In Figure 30, evaluations were performed for transformations between mesh D and mesh E, where mesh E is near the expected inaccurate Nelder Mead optimum. In the figure, mesh D is undistorted on the left side, 0% between mesh D and mesh E. At the right side mesh D is skewed to match the undistorted design of mesh E. Figure 30 (a) illustrates that each mesh miscalculates the objective function when skewed, but the slope is similar. Figure 30 (b) re-plots the data from mesh D in Figure 30 (a) so that small changes can be seen more clearly. An example of the problem with the Nelder Mead solution and the general instability of the mesh skewing scheme can be seen here.

There was a small local minimum near the undistorted mesh. The more cautious gradient method happened to find this minimum but the Nelder Mead method stepped over it and continued to the artificial minimum created by the inaccuracy of the skewed mesh. As it was noted, and depicted in Figure 24 (b) and (c), the noise in evaluating a mesh was significantly smaller near the undistorted mesh than further away, so there was some confidence that the minimum near the undistorted mesh was correct while the minimum further away might be due to error. Based on the problems discussed, optimization of the spiral mandrel die using this method was considered ineffective. If a poor performing design was taken as the initial design too many meshes would have to be created manually to find the optimum. To avoid this, the program could be limited to making small improvements to good designs. If this were done, few meshes would be required to optimize but the gain would be quickly limited by the problems with noise. This method would then be limited for use as a tool in manual optimization. A program could evaluate gradient and Hessian information about a design and make recommendations to a designer who would then manually create the new model and mesh. For this process to be efficient the designer would still need some experience to utilize the predicted gradient and Hessian for optimization of the die.

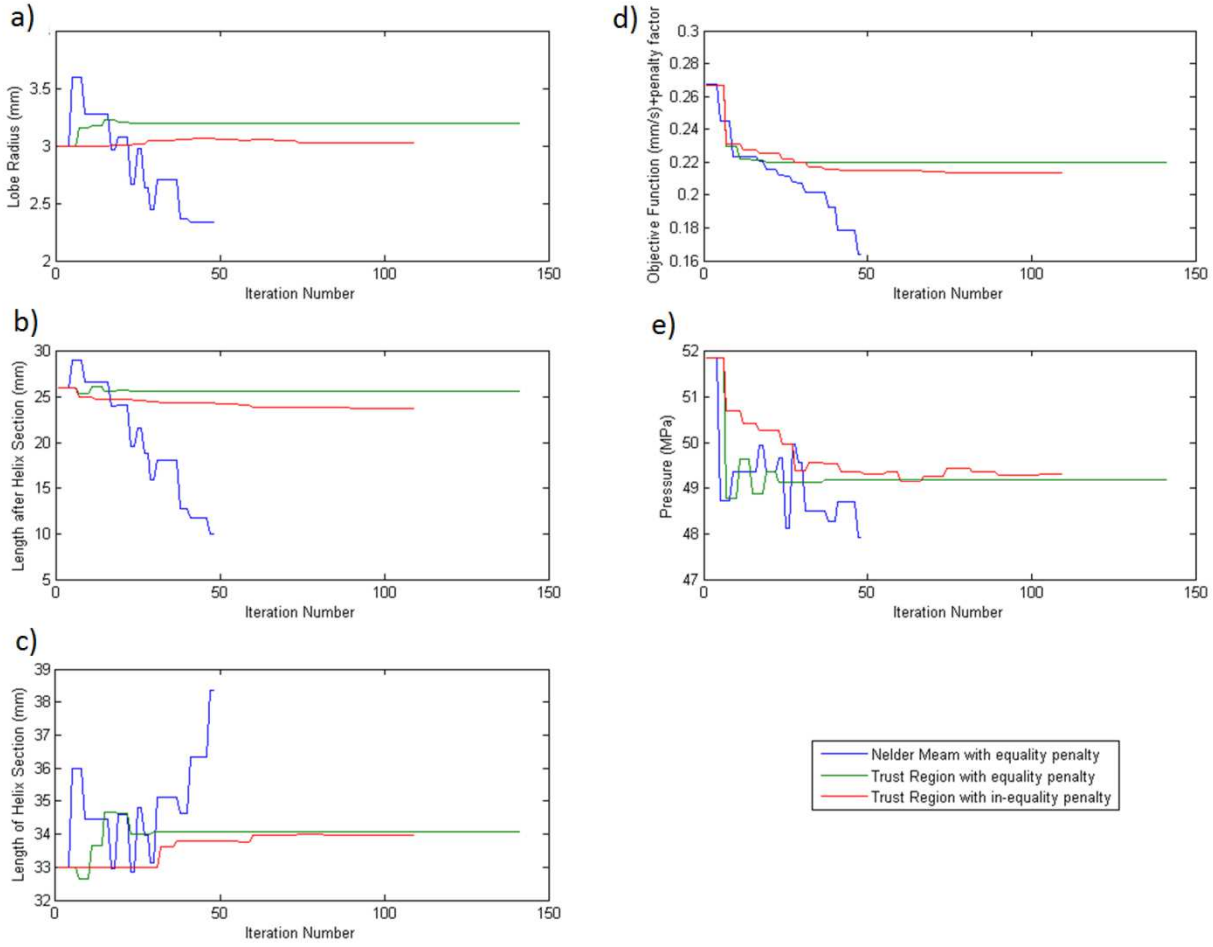

Figure 29: Solutions for Spiral Mandrel die using Nelder Mead and Trust Region methods. a) Lobe radius (mm) by iteration number; b) Length after helix section (mm) by iteration number; c)Length of helix section (mm) by iteration number; d)Objective function (mm/s) with penalty factor by iteration number; e)Pressure by iteration number

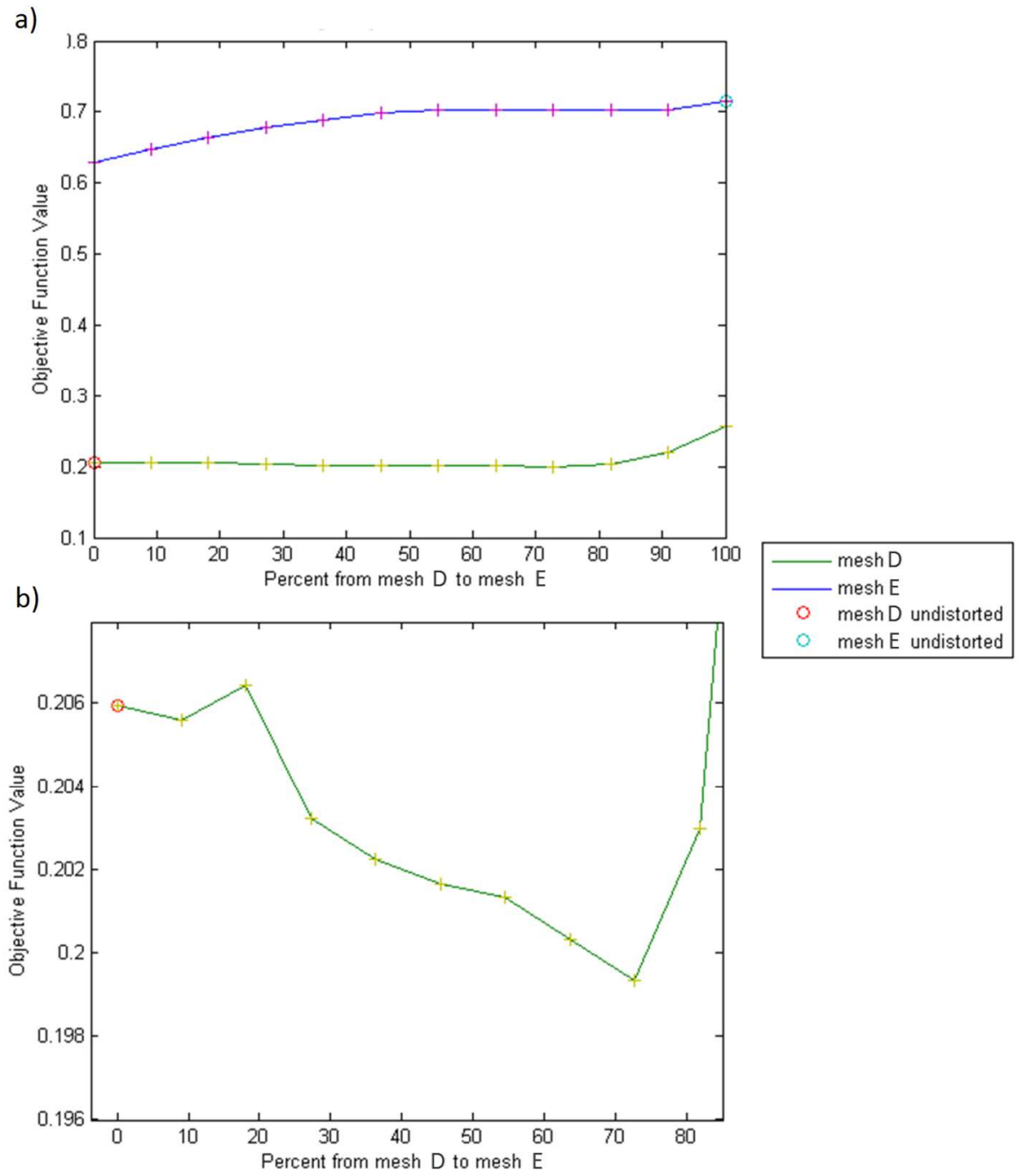

Figure 30: Distortion near convergence for Spiral Mandrel die. a) Comparison of objective function values for mesh D and mesh E; b) Detail view of objective function values for mesh D

Conclusions

The trust region method when applied to the flat die could successfully optimize the die with the presence of noise in the system. The optimization program could be directly applied to other die geometries with the adaption of a the optimization parameters. Maximum step size can be guessed a with knowledge of the typical variation in design parameters. Only determining the new minimum discreet derivative target change would require some effort. In this study the minimum discreet derivative target change was set to .25 the scale of noise in the system, which needed to be determined before running the optimization program. A discreet derivative in a noisy system requires feedback to maintain appropriate discreet step sizes. Adjusting the step size for each design parameter independently based on the resulting change in objective function was found to be effective in this work. For efficiency and diversity, the forward and backward difference approximations can be alternated and central difference approximations used only when needed. Allowing changes in the target value for the change in the discreet derivative not only allows the system to adjust between regions of steep and shallow gradient but also works to expand the search area outside a small local minimum. When a minimum, either local or global, is found it is possible that a given design may be tested repeatedly resulting in a wastage of the computing resources. Therefore, a system that logs and stores results to avoid repeat evaluations may be useful in reducing processing time. For some parameters the final optimized solution was found to be in the direction opposite to the initial improvement. This attribute would be difficult for a designer working on manual optimization to predict and supports the use of automated optimization algorithms not only as potential time savers but possibly finding better final designs than a designer would be able to find manually.

Due to the inaccuracy in the prediction with skewed meshes, the usefulness of the mesh skewing concept was limited to finding a direction for the change in geometric parameters. Knowing the direction of change the designer would then have to update the model and re-evaluate. In this sense this program could be more of a tool for manual optimization than a automatic optimizer. It was found there could be

some noise in the prediction even near the non-distorted geometry. These may be due to the discretization error or due to a non-continuous change when the mesh is skewed. But it could be shown that this is not related to the element correction scheme. Although any opportunity to reduce the noise in this system would improve the results there will always be a degree of noise to deal with. Considering the results from the flat die, it is more effective to develop methods to adapt to a noisy system than try to remove the noise altogether.

Future Work

The optimization program for the flat die developed in this work should be further tested for robustness. In the present work, only a couple of starting die geometries have been tested for only one material and boundary condition set. A sensitivity study could be done for the optimization parameters. Both initial values and limit values for the trust region size and target objective function change for the discrete derivative may need to be adjusted for new dies. Adjustment factors, such as how much the discrete derivative step size should increase when the change in objective function is too small, could be tuned to improve efficiency. Further efficiency gains could be made by tracking all evaluations and results and using this data to skip re-solving a given design.

After the robustness of the optimization algorithm has been tested and optimization parameters have been fine tuned, a study on variations of optimal designs relative to boundary conditions and material parameters would be interesting. How different are the optimum points for two different materials? Does wall temperature affect the optimal design? Can a trend be found to link changes in die width to changes in optimal design? If such trends are found, a rule of thumb model could be developed that would quickly give a near optimal design before starting the optimization process.

For the spiral mandrel die, a code could be developed using the mesh skewing scheme as a tool to aid in manual optimization. This code could find gradient information and potentially Hessian information and make recommendations to the

designer for the next geometry to be attempted. This code would be focused on minimizing the number of new die geometries that would require new meshes to be manually created. This new code could use many evaluations near the undistorted mesh to improve the prediction of the new geometry further away from the undistorted mesh. The Unigraphics NX model for the spiral mandrel die has already been developed for this purpose. The designer would only have to modify three values in the expressions table or load an already modified expressions table to generate a new die geometry. Once a die geometry is defined the new mesh could be generated easily manually.

Bibliography

[1] W. Michaeli, Extrusion Dies for Plastics and Rubber, 2nd ed., New York, NY: Hanser Publishers, 1992.

[2] Simmetrix Inc., "Simmetrix 6.3," Simmetrix Inc., Clifton Park, 2008.

[3] M. Gupta, "PolyXtrue," Plastic Flow, Houghton, 2011.

[4] M. Gupta, Interviewee, *Design with Plastics.* [Interview]. 15 12 2009.

[5] Y. Sun and M. Gupta, "Optimization of a Flat Die Including Elongational Viscosity Effects," *Intern. Polymer Processing,* 2005.

[6] D. E. Smith, D. A. Tortorelli and C. L. Tucker, "Optimal Design for Polymer Extrusion. Part I: Sensitivity Analysis for Nonlinear Steady-State Systems," *Comput. Methods Appl. Mech. Engrg.,* vol. 167, pp. 283-302, 1998.

[7] W. Michaeli and S. Kaul, "Approach of an Automatic Extrusion Die Optimization," *Journal of Polymer Engineering,* vol. 24, pp. 123-136, 2004.

[8] N. Lebaal, F. Schmidt and S. Puissant, "Design and optimization of three-diminsional extrusion dies, using constraint optimization algorithm," *Finite Elements in Analysis and Design,* vol. 45, pp. 333-340, 2009.

[9] C. T. Kelley, "Detection and Remediation of Stagnation in the Nelder-Mead Algorithm using a Sufficient Decrease Condition," *Society for Industrial and Applied Mathematics,* vol. 10, no. 1, pp. 43-55, 1999.

[10] H. Okano and M. Koda, "An Optimization Alogorithm Based on Stochastic Sensitivity Analysis for Noisy Objective Landscapes," *Reliability Engineering and System Safety,* vol. 79, pp. 245-252, 2003.

[11] E. Davis and M. Ierapetritou, "Adaptive optimisation of noisy black-box functions inherent in microscopic models," *Computers and Chemical Engineering,* vol. 31, pp. 466-476, 2007.

[12] J. Nocedal and S. J. Wright, Numerical Optimization, P. Glynn and S. M. Robinson, Eds., New York, New York: Springer-Verlag, 1999.

Appendix

Generating Parasolid models for the Spiral Mandrel Die

The limiting factor for the spiral mandrel die was in the definition of the spiral sections. Parasolid uses B-splines to define the surface and edges of the spirals. The documentation that was available details the general form of how the points were weighted together to form the surface but does not include the B-spline basis functions. In an attempt to replicate the B-spline changes due to lobe radius changes the movements of the nodes in the splines were investigated. The nodes do not follow the surface but instead use weighting factors included in the parasolid data. The helix surface is controlled by 154 nodes (Figure 31), which are in sets of twelve (Figure 32). As the radius changes ten of the nodes move in or out from a central point linearly, two remaining nodes move linearly towards or away from each other. Whereas when the twelve nodes were updated the helix B-spline would be easily replicated for the new lobe radiuses. Attempting to replicate the edge between the helix and the inner wall of the die was more complicated. Knowing the dimensions of the helix and the inner wall of the die the intersection could be calculated. However, the geometric form created by this method did not pass the consistency checks applied by the meshing program. The B-spline describing the surface is an approximation of the intended surface, see the approximation of the linearly expanding helical radius (Figure 33). Without an ability to model the B-spline defined surface, the intersection cannot be found.

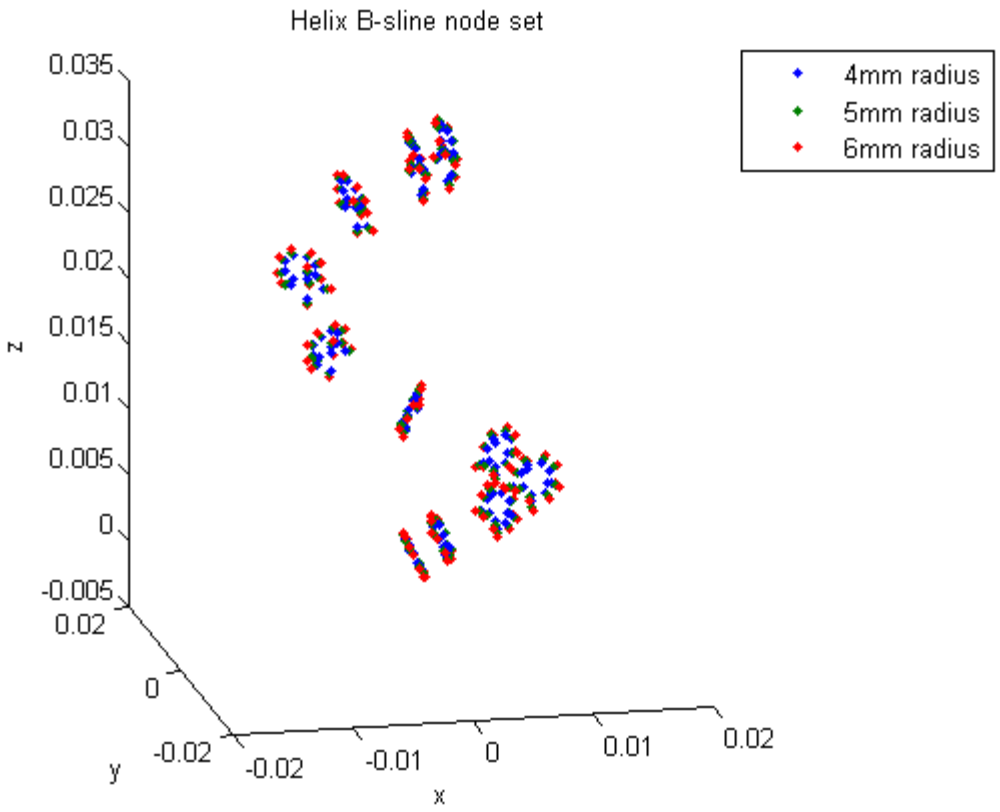

Figure 31: Nodes used to in Helix B-spline

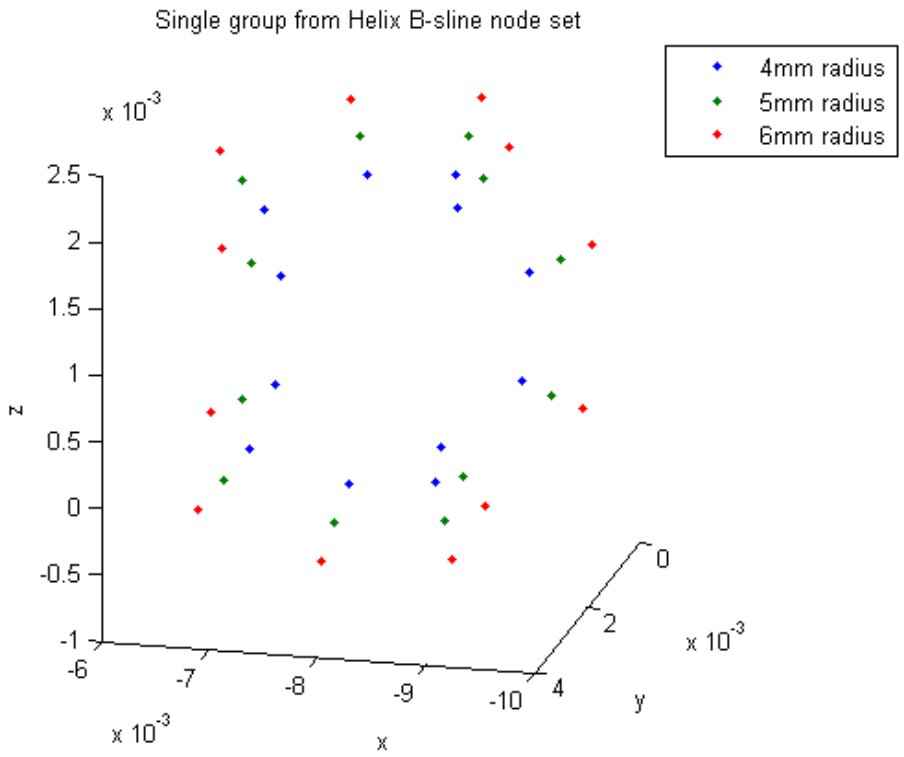

Figure 32: One set of 12 nodes from the Helix B-spline, for three different lobe radiuses

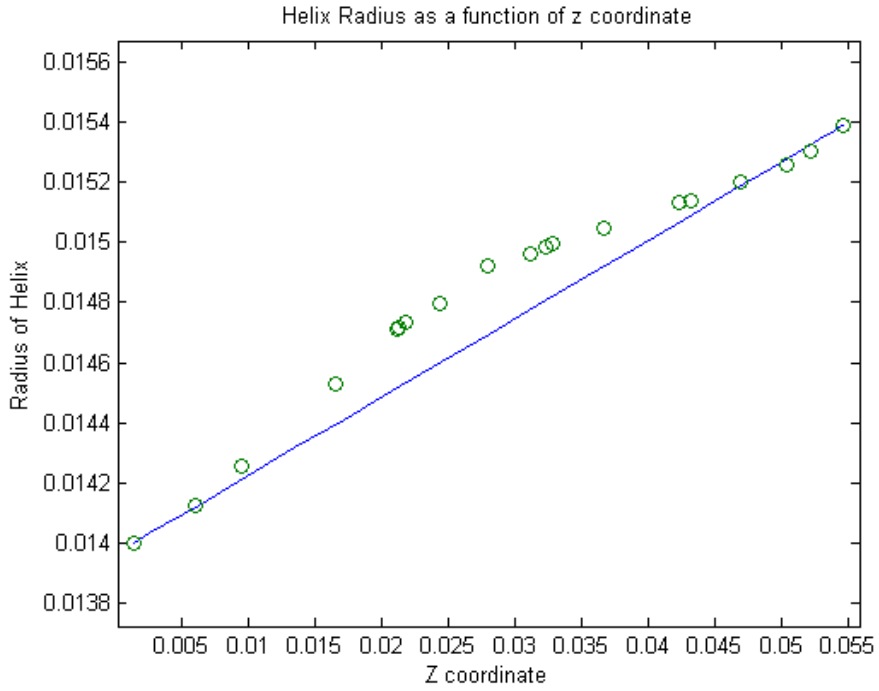

Figure 33: Radius of the helix as compared to linear, helix radius measured in NX 5.0

Results flat die, first optimization

An earlier version of the trust region code for the flat die reached an optimum but at a slower rate. Several optimization parameters were adjusted for the second optimization after reviewing the process this version of the program had taken. Following is a discussion of this first optimization. The trends and conclusions are matching what is discussed earlier in this study.

It can be seen in Figure 34, the original design is quickly improved at the early stages of the program. This quick pace is due to the contour of the objective function being steep. The flatter sections are then where the code has more difficulty and needs to adjust the controlling parameters. For example, between the 150th and the 200th evaluation, the trust region is reduced due to poor steps. These relatively short steps have difficulty with noise and the trust region is further reduced. To prevent this from ending the program the algorithm took a step much larger than the trust region. When this returned an improved objective function value the trust region was redefined to this larger size. Such a step is taken when the trust region is much smaller than the average step taken to find the gradient and both the normal step and the second shorter linear optimization step fail. At the end about 100 evaluations are used to check the area around the optimum to try to determine if the current point is a local or global optimum. This is done by allowing the central difference discreet derivative to increase the target value for change in the objective function every time all evaluated points are worse than the central point. When this target value increases beyond the limit the program ends. Using the same format Figure 35 details the pressure values and shows that throughout the optimization the equality penalty held the pressure to near the target value, 5.5 MPa.

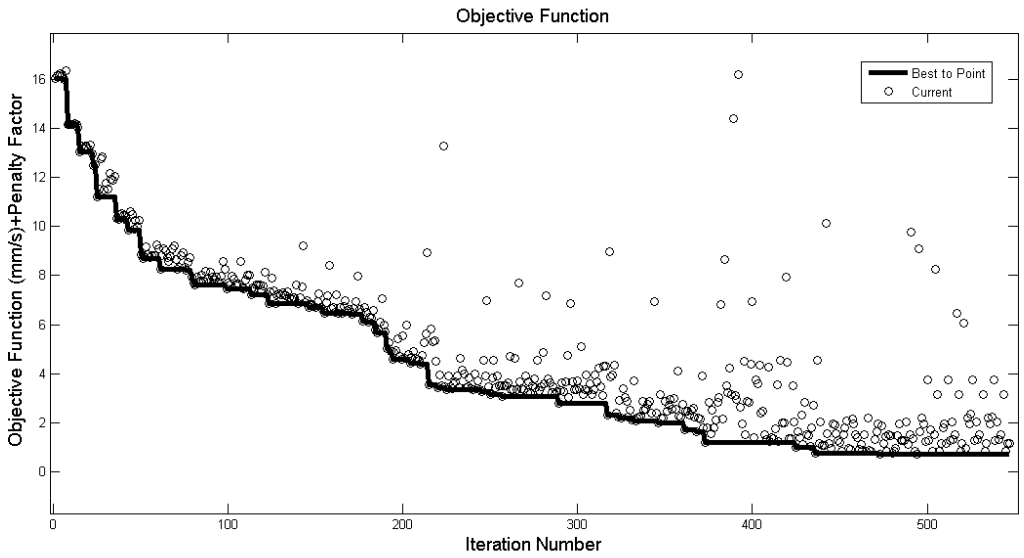

Figure 34: Objective function progression for the flat die, 2nd version trust region algorithm, first solution, and equality penalty

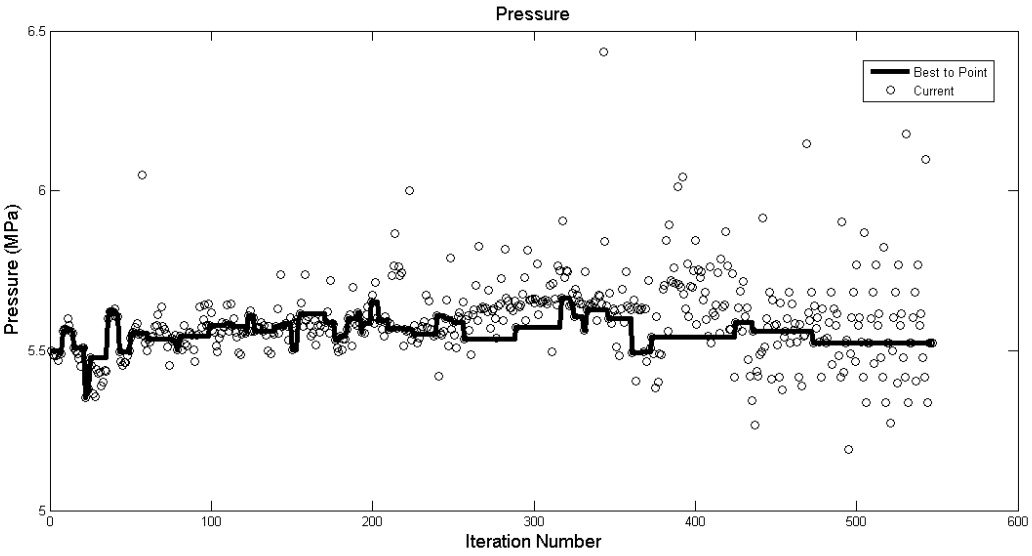

Figure 35: Pressure progression for the flat die, 2nd version trust region algorithm, first solution, and equality penalty

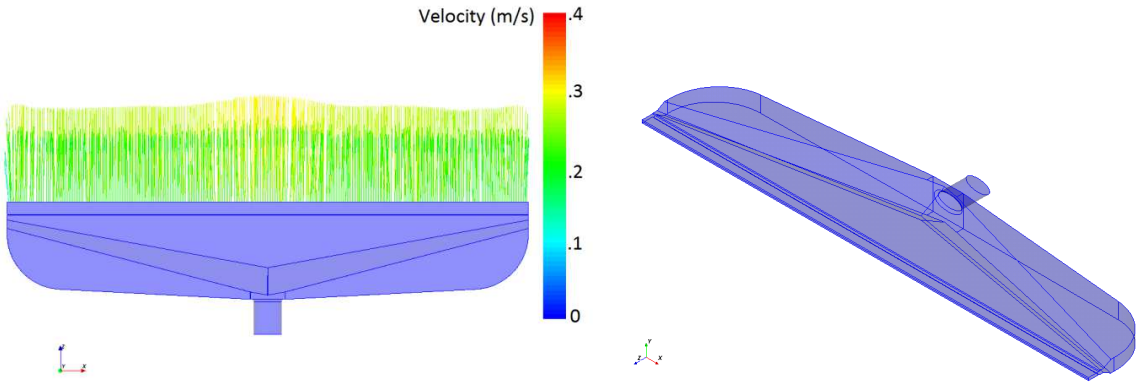

Figure 36: Optimized Design Velocity Vector plot at Exit and Optimized Die Model from the first solution set using an equality penalty

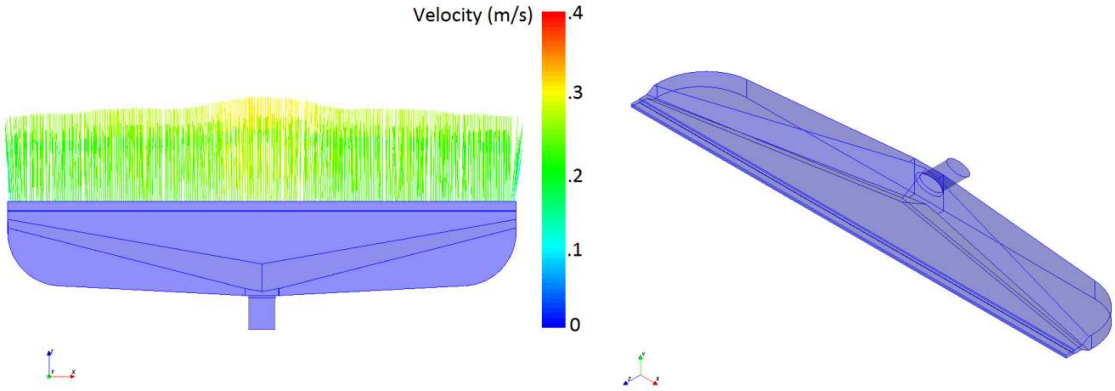

Figure 37: Optimized Design Velocity Vector plot at Exit and Optimized Die Model from the first solution set using an equality penalty

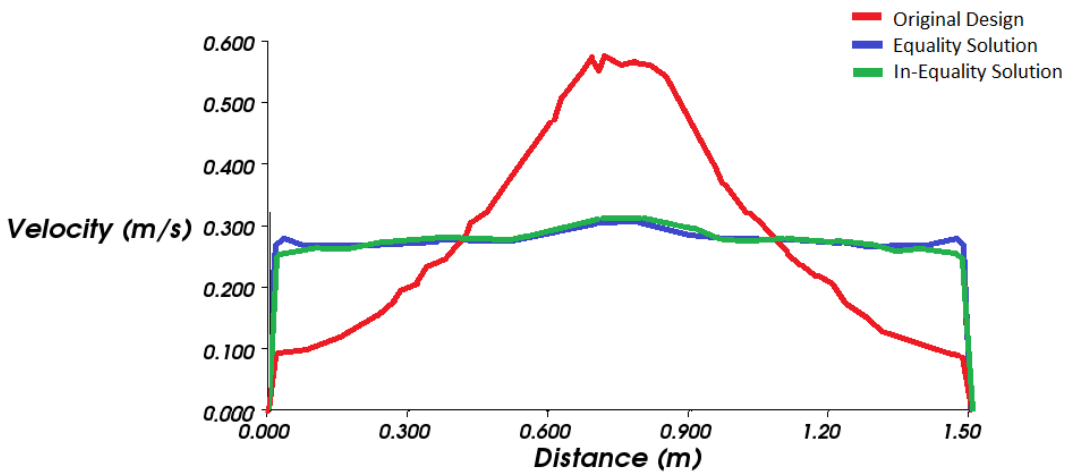

Figure 38: Velocity Distribution across Die Exit, from the first solution set

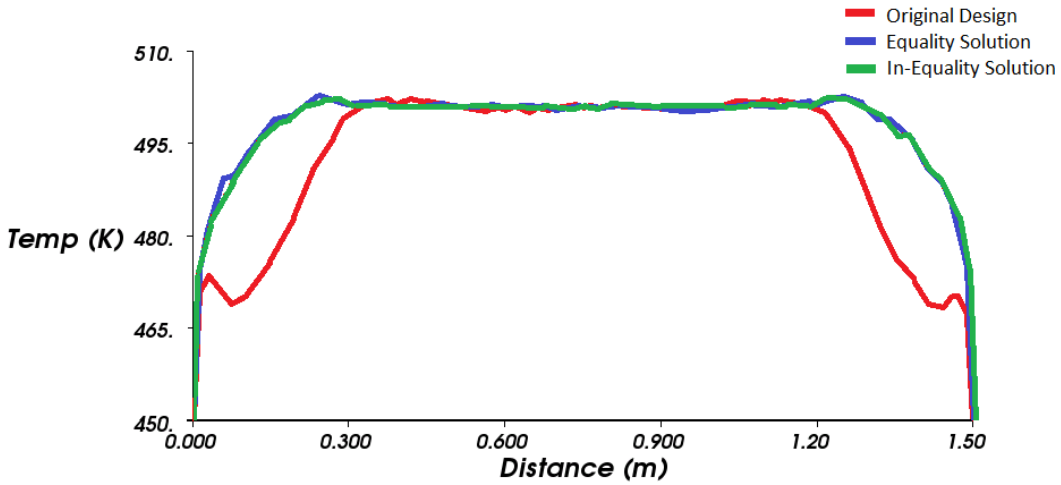

Figure 39: Temperature Distribution across Die Exit, from the first solution set

Additional figures for the flat die version of the trust region algorithm

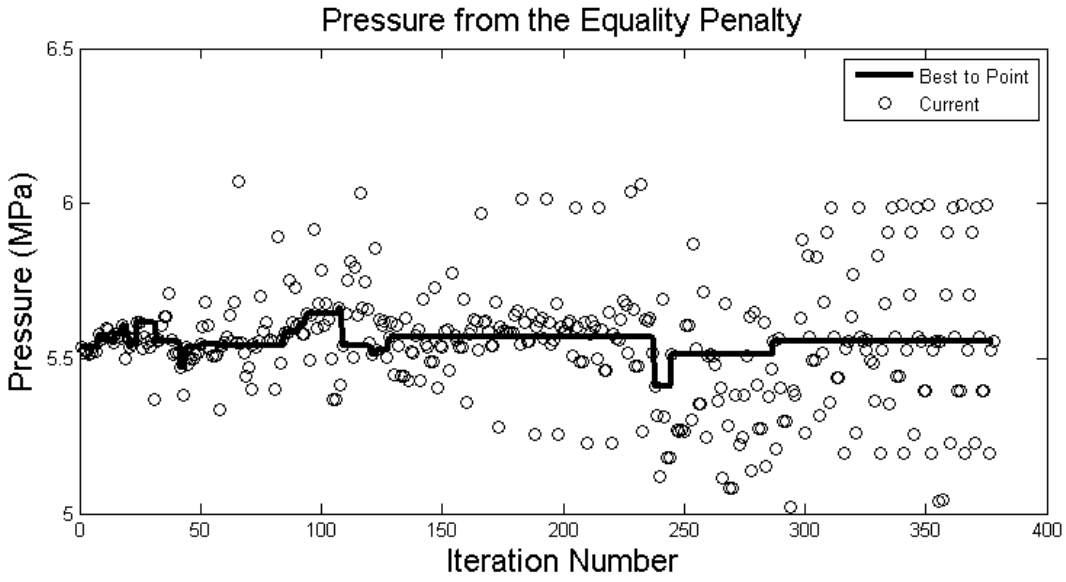

Figure 40: Development of objective function throughout Equality Penalty Trust Region Optimization

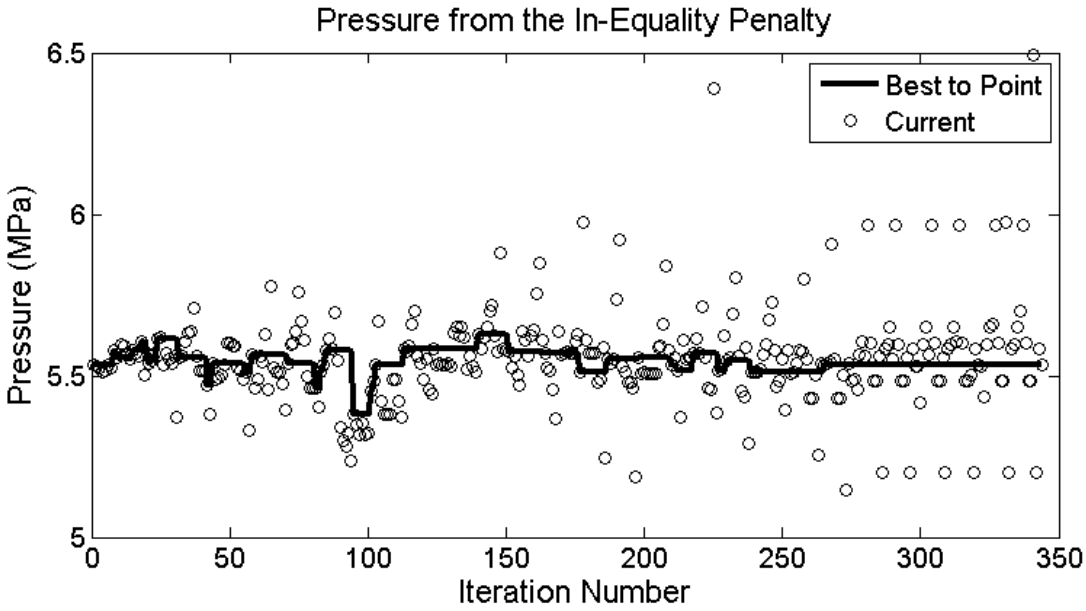

Figure 41 : Development of objective function throughout In-Equality Penalty Trust Region Optimization

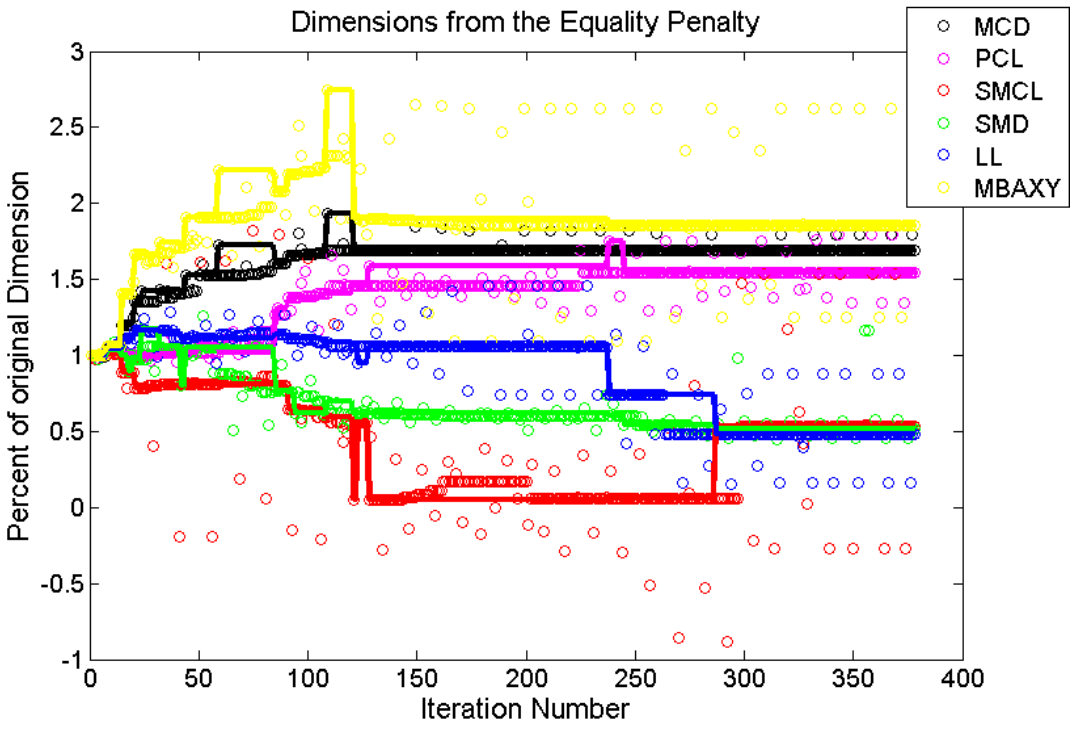

Figure 42: Dimensions throughout Equality Penalty Trust Region Optimization (full data set)

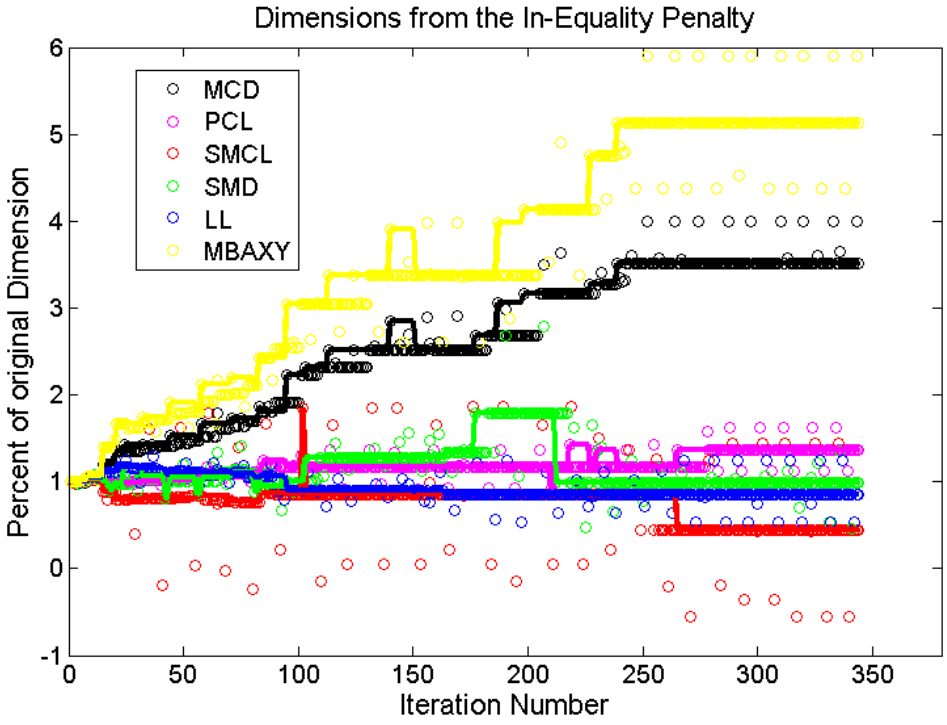

Figure 43: Dimensions throughout In-Equality Penalty Trust Region Optimization (full data set)