Numerical Optimization Project

The Brachistochrone Problem

Author:
Nagarajan

Course Advisors:
Prof. Walter Murray
Kathy Lora Jenson, CA
Nick Henderson, CA

Stanford University
October 30, 2009

Contents

List of Figures

Praise for the project...
Hi:
Thanks for sending it. I am sure had I been a student in the class I would not have done such a comprehensive job. Well done.

Walter

Chapter 1

Introduction

The *brachistochrone* curve, or curve of *shortest time* (literally in *Greek*), is the curve between two points that is covered in the least time by a body that travels under the action of constant gravity. The problem is described below pictorically...

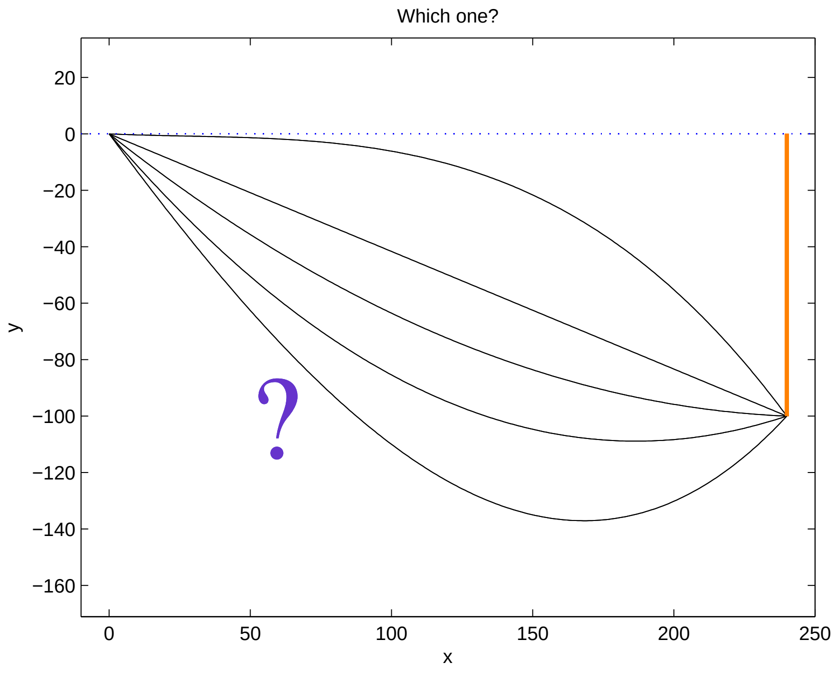

Figure 1.1: Some possible Paths

1.1 History

In the words of Johann Bernoulli, who originally posed the problem,

I, Johann Bernoulli, address the most brilliant mathematicians in the world. Nothing is more attractive to intelligent people than an honest, challenging problem, whose possible solution will bestow fame and remain as a lasting monument. Following the example set by Pascal, Fermat, etc., I hope to gain the gratitude of the whole scientific community by placing before the finest mathematicians of our time a problem which will test their methods and the strength of their intellect. If someone communicates to me the solution of the proposed problem, I shall publicly declare him worthy of praise.

Earlier, Galileo incorrectly stated in 1638 in his Discourse on two new sciences that this curve was an arc of a circle. Johann Bernoulli solved the problem before posing it to readers of Acta Eruditorum in June 1696. Five mathematicians responded with solutions: Isaac Newton, Jakob Bernoulli (Johann's brother), Gottfried Leibniz and Guillaume Francois Antoine l'Hopital. Four of the solutions (excluding l'Hopital's) were published in the May 1697 edition of the same publication.

In an attempt to outdo his brother, Jakob Bernoulli created a harder version of the brachistochrone problem. In solving it, he developed new methods that were refined by Leonhard Euler into what the latter called (in 1766) the *calculus of variations.*

1.2 Analytical Solution

The analytical solution to the brachistochrone problem is given by the equations,

$$x = a(\theta - sin\theta) \tag{1.1}$$

$$y = a(1 - cos\theta) \tag{1.2}$$

where the initial point is assumed to be at the origin. The solution curve is a *cycloid* with the cusp at the origin. The time taken to travel to the destination point (x_0, y_0) along this curve is given by,

$$t_0 = \sqrt{\frac{a_0}{g}}\theta_0$$

We solve the problem numerically using optimization techniques and compare the solution against the optimal analytical time.

1.2.1 Finding a_0 and θ_0

A good way of comparing the numerical solution against the analytical solution is to find the difference in the minimum time predicted by both methods. We require to find the value of a_0 and θ_0 in order to find the analytical time. A program performs a binary search for an appropriate a, such that the resultant θ satisfies both the equations (1.1) and (1.2) of the cycloid.

1.3 Preliminary Analysis

It is clear that the problem has no solution if the destination point is higher than the initial point. In the case when the destination point is at the same level, errors in calculation can make the problem infeasible to a numerical solution (A nearby problem does not have a solution :-)). In this case, we perturb the data and make the destination point a bit lower than the initial point to get an approximate solution, such that it is well within the limits set by numerical errors.

Chapter 2

Numerical Solution

We now describe the method of Numerical Optimization and supporting algorithms used to find the Brachistochrone curve.

2.1 Discretization Model

The problem is first discretized appropriately into n variables which correspond to the height of the path at intermediate discrete points.

2.1.1 Uniform discretization

A uniform division of the horizontal distance between the 2 points in to (n+1) equal parts leads to, what I refer to as, "Uniform Discretization". Figure 2.1 shows 3 different possible paths using uniform discretizaton.

2.1.2 Non-uniform Discretization

When the horizontal distance between the points is not divided uniformly, it is referred to as "Non-uniform Discretization". This is desirable as the error between the approximated curve and the actual curve is high in places where the curvature is high. Figure 2.2 depicts one possible non-uniformly discretized path. Notice that the division of the horizontal distance near the initial point is much finer than farther from it.

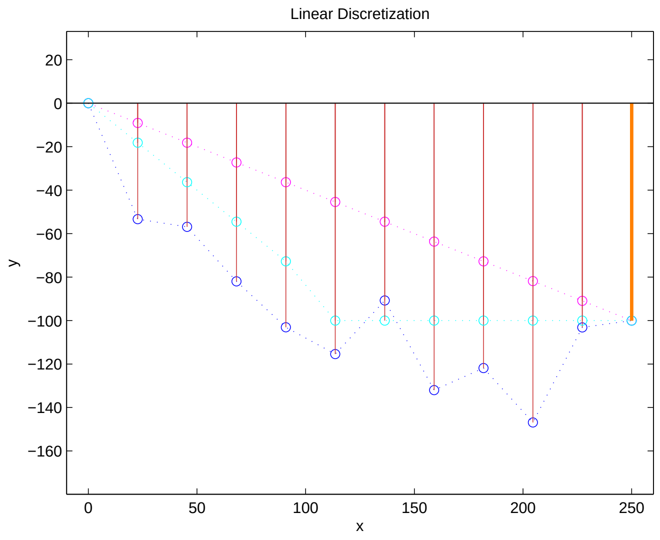

Figure 2.1: Uniform Discretization

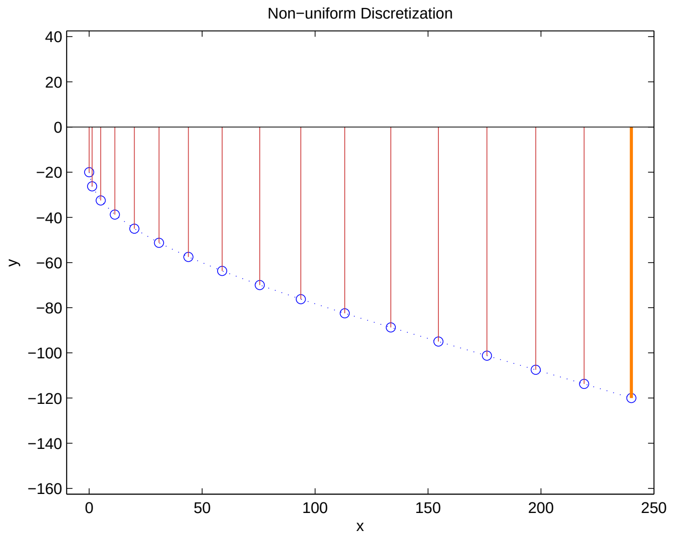

Figure 2.2: Non-uniform Discretization

The width of each interval here depends on the curvature of the approximated curve near the optimal solution.

2.1.3 Optimal Discretization

This raises an interesting question about the optimal non-uniform discretization. "What is the optimal spacing of the variables in the horizontal direction which gives the minimum error as compared to the optimal brachistochrone?". This is an optimization problem in '2n' variables. We will talk more about this topic in the appendix.

2.2 Equations for Time and Gradient

For a particular discretization and a set of values for the variables, we join the intermediate points by appropriate curves (or straight lines) and find the time for the whole path. The time and rate of change of the time with respect to the different variables (*gradient*) is then computed analytically. Let us define some useful intermediate variables which will be used to define the final formulas for time and gradient. Let l_k denote the horizontal length of each interval and h_k denote the value of the height at beginning of each interval, and $N = n + 2$ (including the end points h_1 and h_N), then define

$$d_k = h_k - h_{k+1} \qquad \forall\{1 \leq k \leq N-1\} \tag{2.1}$$

$$y_k = h_1 - h_k \qquad \forall\{2 \leq k \leq N\} \tag{2.2}$$

$$s_k = \sqrt{l_k^2 + d_k^2} \qquad \forall\{1 \leq k \leq N-1\} \tag{2.3}$$

$$r_k = \sqrt{\frac{2}{g}}\frac{s_k}{d_k} \qquad \forall\{d_k \neq 0\} \tag{2.4}$$

We impose a restriction that $y_k > 0$. It is justified as $y_k = 0$ is a problem which is close to another with no solution. We impose this restriction on the initial conditions as well as on intermediate values of the variables. The amount for the time taken by a particle to travel each interval, in a straight line path, is given by the formulas,

$$t_1 = s_1\sqrt{\frac{2}{gy_2}}; \qquad \{y_2 \neq 0\} \tag{2.5}$$

$$t_k = r_k\left\{\sqrt{y_{k+1}} - \sqrt{y_k}\right\} \qquad \forall\{k > 1, d_k \neq 0\} \tag{2.6}$$

$$t_k = \frac{s_k}{\sqrt{2gy_k}}; \qquad \forall\{k > 1, d_k = 0, y_k \neq 0\} \tag{2.7}$$

$$\tag{2.8}$$

The total time is given by

$$T = \sum_{k=1}^{N-1} t_k \tag{2.9}$$

The gradient for $\{2 \leq k \leq (N-1)\}, y_k > 0$ is computed using the formulas

$$\frac{\partial t_k}{\partial h_k} = r_k \left\{ \frac{(\sqrt{y_{k+1}} - \sqrt{y_k})(d_k^2 - s_k^2)}{s_k^2 d_k} + \frac{1}{2\sqrt{y_k}} \right\} \qquad \{d_k \neq 0\} \tag{2.10}$$

$$\frac{\partial t_k}{\partial h_k} = \frac{l_k^2}{4 t_k g y_k^2} \qquad \{d_k = 0\} \tag{2.11}$$

$$\frac{\partial t_{k-1}}{\partial h_k} = r_{k-1} \left\{ \frac{(\sqrt{y_k} - \sqrt{y_{k-1}})(s_{k-1}^2 - d_{k-1}^2)}{s_{k-1}^2 d_{k-1}} - \frac{1}{2\sqrt{y_k}} \right\} \qquad \{d_{k-1} \neq 0\} \tag{2.12}$$

$$\frac{\partial t_{k-1}}{\partial h_k} = \frac{l_{k-1}}{\sqrt{32 g y_{k-1}^3}} \qquad \{d_{k-1} = 0\} \tag{2.13}$$

The final gradient is given by

$$\frac{\partial T}{\partial h_k} = \frac{\partial t_k}{\partial h_k} + \frac{\partial t_{k-1}}{\partial h_k} \qquad \{2 \leq k \leq N-1\} \tag{2.14}$$

Gradient by central difference For the line search test functions, whenever the gradient is not available in analytical form, the gradient is found by a central difference using $\delta x = 10^{-4}$. This uses n evaluations of the objective function.

2.3 Algorithm

The Algorithm uses the Quasi newton direction using the BFGS update and a line search using gamma conditions to find a local minimum of the objective function.

2.3.1 Quasi Newton - BFGS

The equation of the quasi-newton direction is given by

$$p_k = -B_k^{-1} g_k$$

where the B_k is defined by the BFGS update. We set $B_0 = I$ and then compute

$$B_{k+1} = B_k - \frac{B_k s_k (B_k s_k)^T}{s_k^T B_k s_k} + \frac{y_k y_k^T}{y_k^T s_k}$$

where $s_k = x_{k+1} - x_k$ and $y_k = \nabla f_{k+1} - \nabla f_k$.

Another way to use the BFGS update is to directly update the inverse of the Hessian, H_k. Again, $H_0 = I$. This approach reduces the computation time required for computing the search direction. The inverse update is given by

$$H_{k+1}^{-1} = H_k^{-1} + \frac{(s_k s_k^T)(s_k^T y_k + y_k^T H_k^{-1} y_k)}{(s_k^T y_k)^2} - \frac{H_k^{-1} y_k s_k^T + s_k y_k^T H_k^{-1}}{s_k^T y_k}$$

The direction is now computed directly as

$$p_k = -H_k g_k$$

Though this technique can be faster, but it is also prone to more numerical errors, hence we will not use the method of updating the inverse of the Hessian.
A common practice is to add I to the hessian or set the computed approximate Hessian to I when the Hessian becomes very ill conditioned, typically $cond(H) > 10^6$ or when the search direction is not actually a descent direction and line search returns a step size of zero.

2.3.2 Linesearch

A line search using the gamma conditions (as described in Nocedal and Wright) is used to calculate step length in the direction given by the quasi newton. Cubic interpolation is used to find intermediate points. The line search also takes care that the step size should not take the variables into the infeasible region. The line search starts with the initial step size of 1 and tries to bracket the minimizer by increasing or decreasing the step size.

2.3.2.1 Constraints

The constraint on the problem is that any of the variables cannot be greater than the initial point. This constraint is respected by the line search through interaction with the objective function which returns a NaN when the variables are out of bound. A bisection in the current bracket is done by line search in order to decrease the step size and remain in the constrained space.

2.3.2.2 Stopping Criteria

The algorithm stops when the norm of the gradient, $||g|| < 10^{-7}$.

2.3.3 Line Search Testing

2.3.3.1 Quadratic in 2 variables

This was the first test function used to visually gauge the progress the of the Line search. Figures 2.3 and 2.4 compare the Steepest descent and Quasi Newton direction using the same gamma conditions.

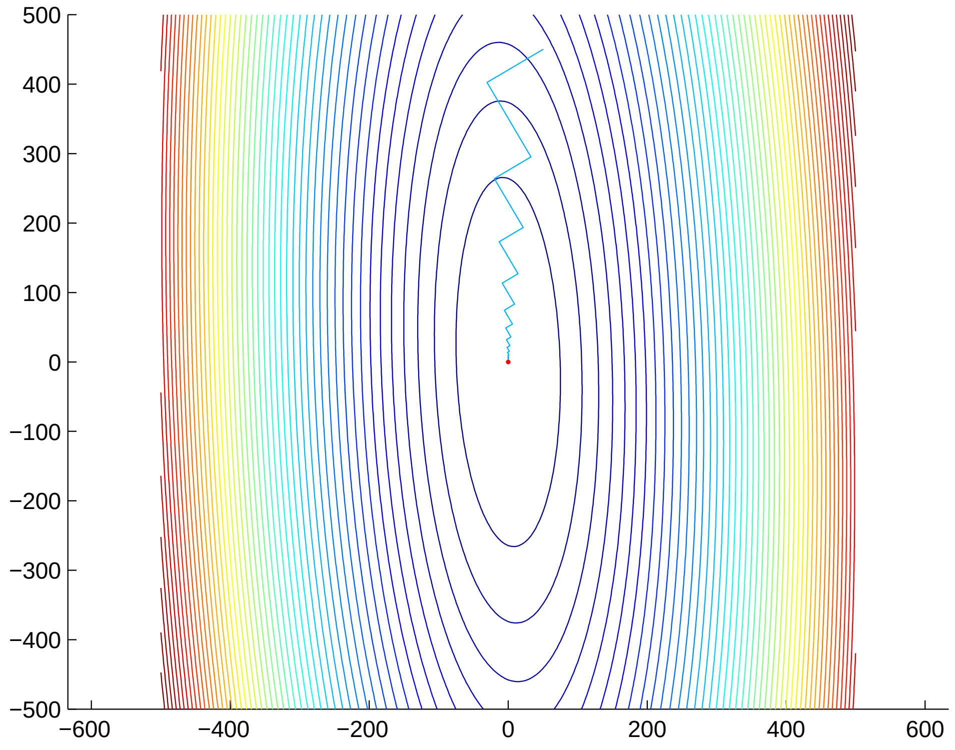

Figure 2.3: Steepest Descent

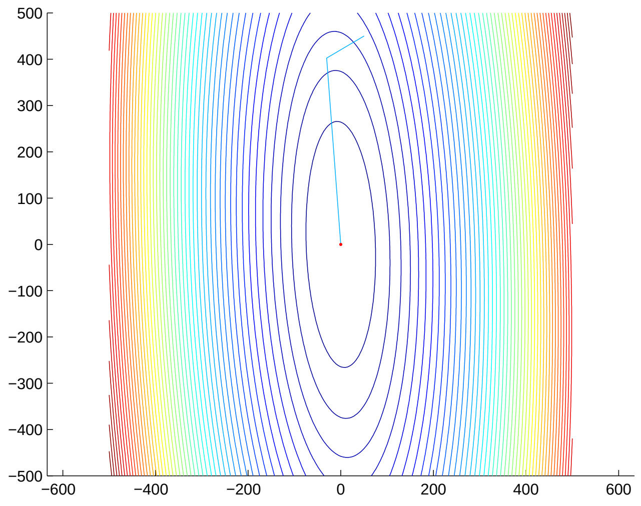

Figure 2.4: Quasi Newton Line search

2.3.3.2 Rosenbrock Function

Also called the banana function, this is a classic function used for testing optimization algorithms. The global minimum lies inside a long and thin curved valley at (1,1). The function is given by

$$f(x) = \sum_{i=1}^{n-1} 100(x(i+1) - x(i)^2)^2 + (1 - x(i))^2$$

The convergence of the line search is shown in figure 2.5.

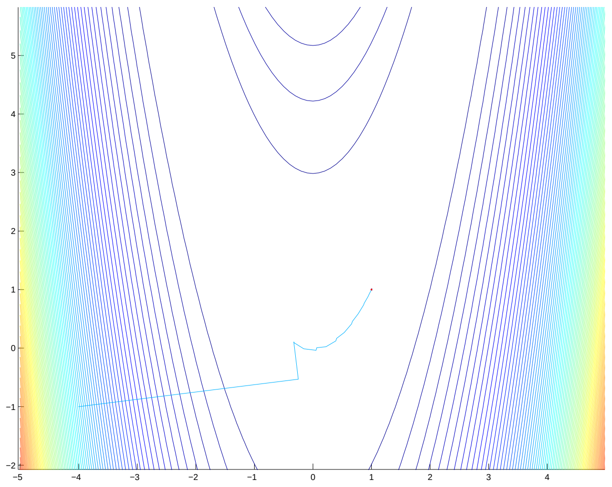

Figure 2.5: Rosenbrock Function

2.3.3.3 Branin's Funtion

This is another optimization test function.[1] It has 4 local minima near the origin. The convergence to one of the minima is shown in the figure 2.6. This example also illustrates that the line search might not converge to the nearest local minimum. It only guarantees convergence to any local minimum.

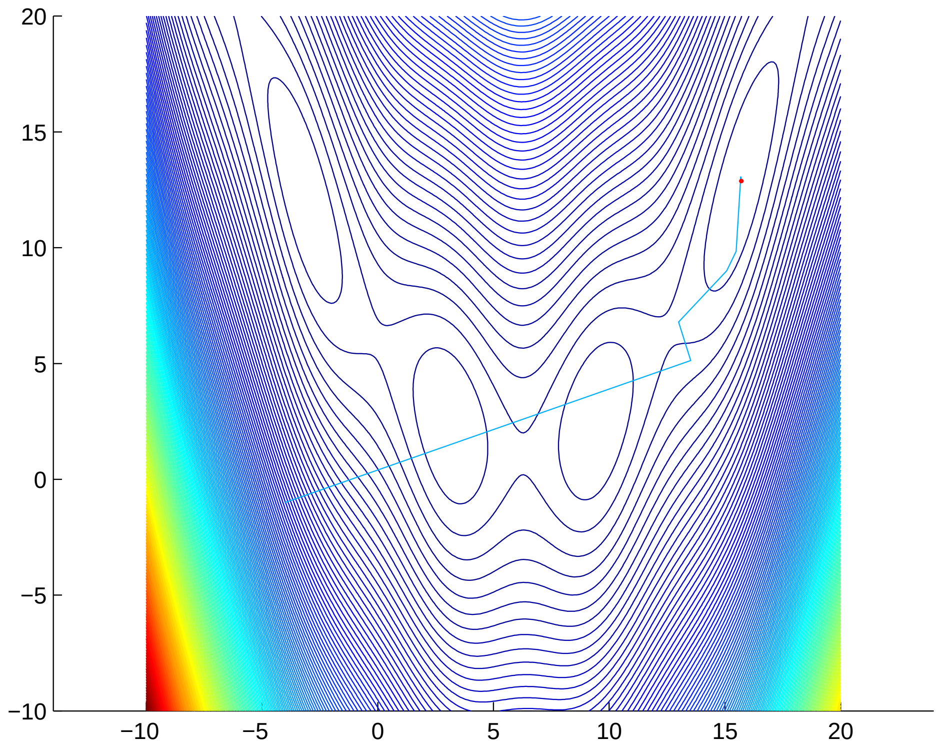

Figure 2.6: Branin's Function

2.3.3.4 Schwefel's Funtion

This is an optimization test function which has lots of variation and hence can give wrong directions of search. Also, it can be noticed that small discrete perturbations in the initial point can change the final local minima. The convergence for a particular case is shown in figure 2.5 and 2.6.

[1]Thanks to Gurjeet for suggesting Branin's and Schwefel's functions

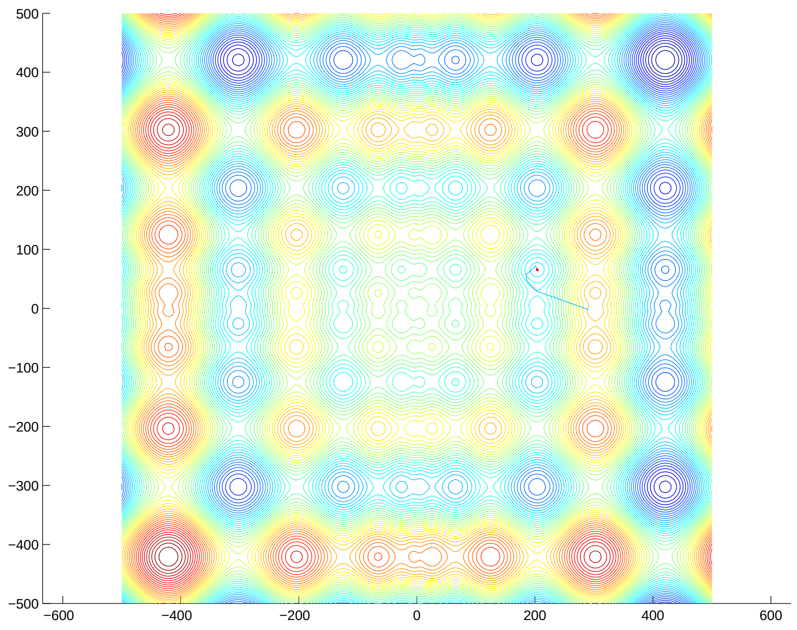

Figure 2.7: Schwefel's Function

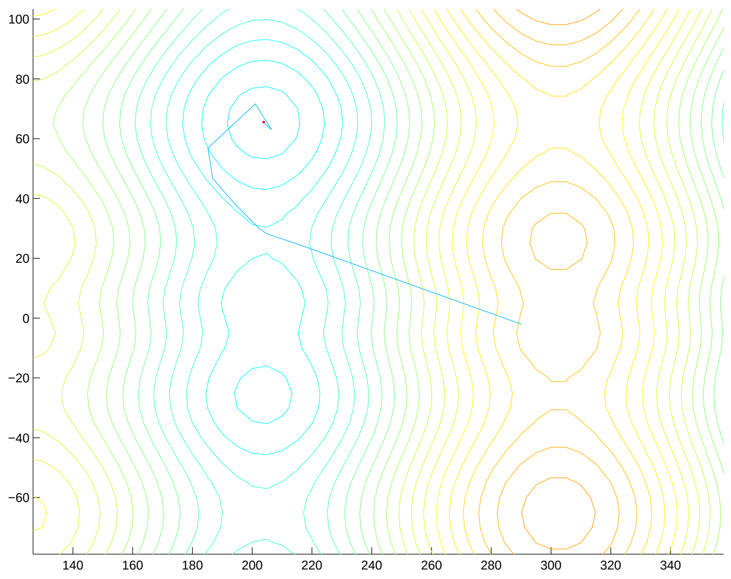

Figure 2.8: Schwefel's Function (Zoomed in)

2.3.3.5 N dimensional Quadratic Function

An N dimensional quadratic function is defined by the equation

$$f(x) = g^T x + \frac{1}{2} x^T H x$$

and the location of the global minima, x^*, is given by

$$x^* = -H^{-1} g$$

provided that H is positive definite.

The function sprandsym(N,1, eig) in matlab, generates a random symmetric matrix with eigenvalues given by the vector eig. Using positive numbers in the vector eig, we generate the desired quadratic function. The function converges within the expected number of steps.

Multiple Eigenvalue Quadratic Also, we expect that optimization converges in less than N steps when multiple eigenvalues are present. We tested this function for various values of N and a few configurations of multiple eigenvalues. Again, the search converges within the expected steps.

Chapter 3

Results

We look at the Brachistochrone using various initial and final points as well as different starting estimates of the curve. We also analyze the running time, the number of function/gradient evaluations as well as the error with respect to the optimal analytical time vs. the number of variables used for discretization.

The figure below shows a typical curve obtained using 29 variables, and convergence in 107 steps. The intermediate steps are colored in cyan and the final curve is in red.

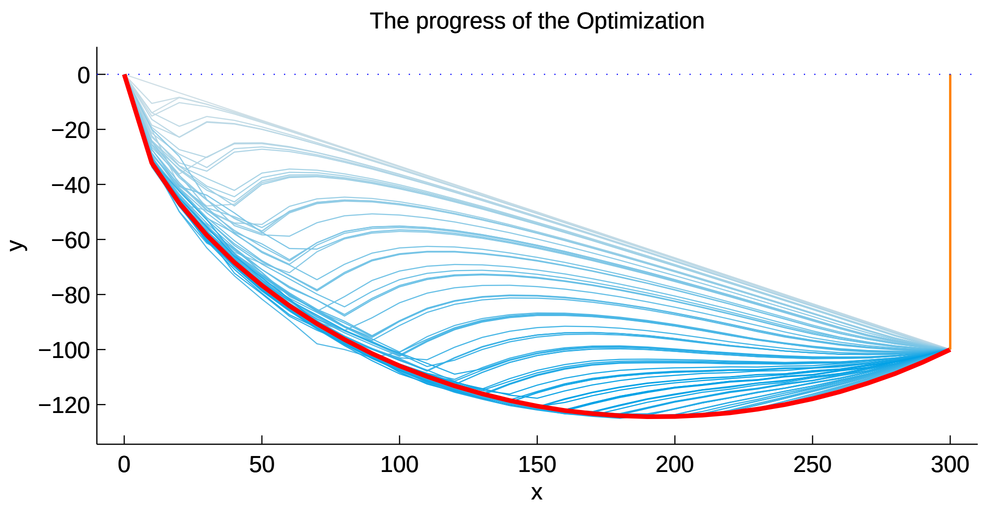

Figure 3.1: Destination (300, -100)

3.1 Curve with Linear Discretization

3.1.1 Optimal curves

We look at some brachistochrones corresponding to different final points. The origin is always the starting point (the method has been tested with other starting locations as well). In this section, we use the straight wedge as the initial curve. Each output gives comprehensive information about the final curve as well as the order and error performance of the algorithm. Each plot contains 6 subplots. Below is an example output followed by information about each plot. The plots are zoom-able as they are vector images.

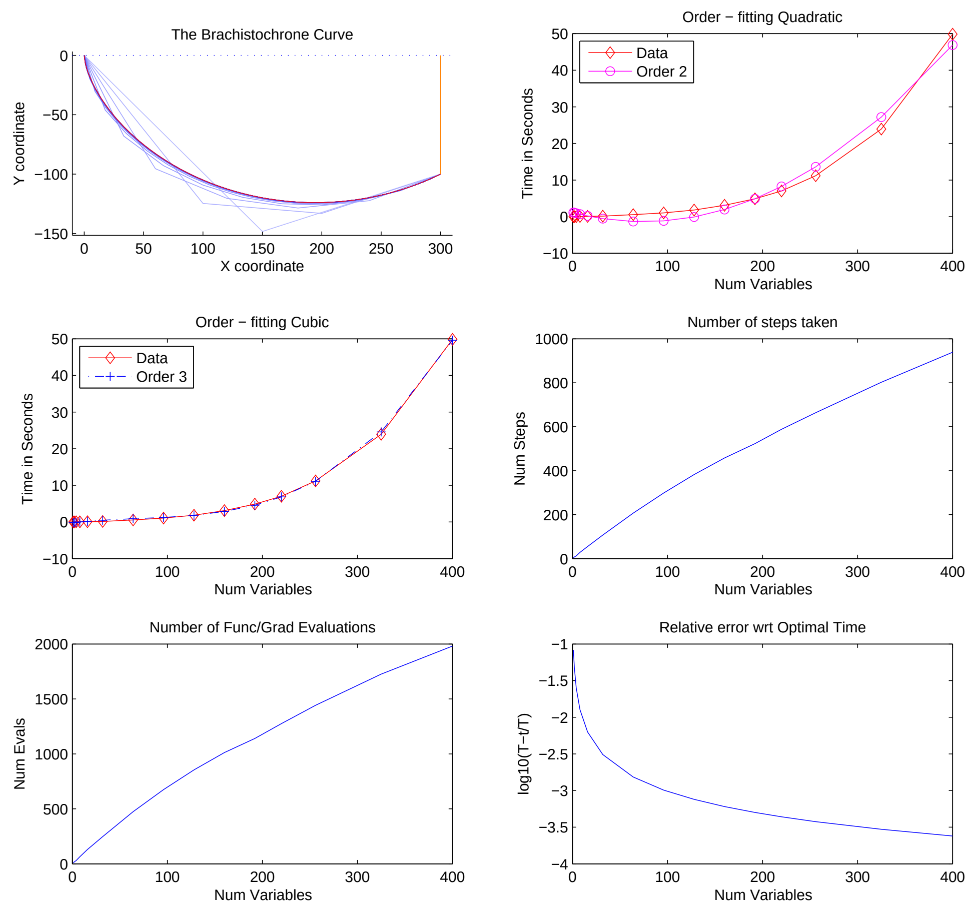

Figure 3.2: Destination (300, -100) with analysis

1. In the first plot, the curves in blue are the final optimized curves obtained for (1, 2, 4, 8, 16, 32, 64, 96, 128, 160, 196, 220, 256, 325)

variables and the curve in red has 400 variables.

2. The second plot (Order - fitting Quadratic) tries to fit a quadratic to the running time for the algorithm vs. num variables.

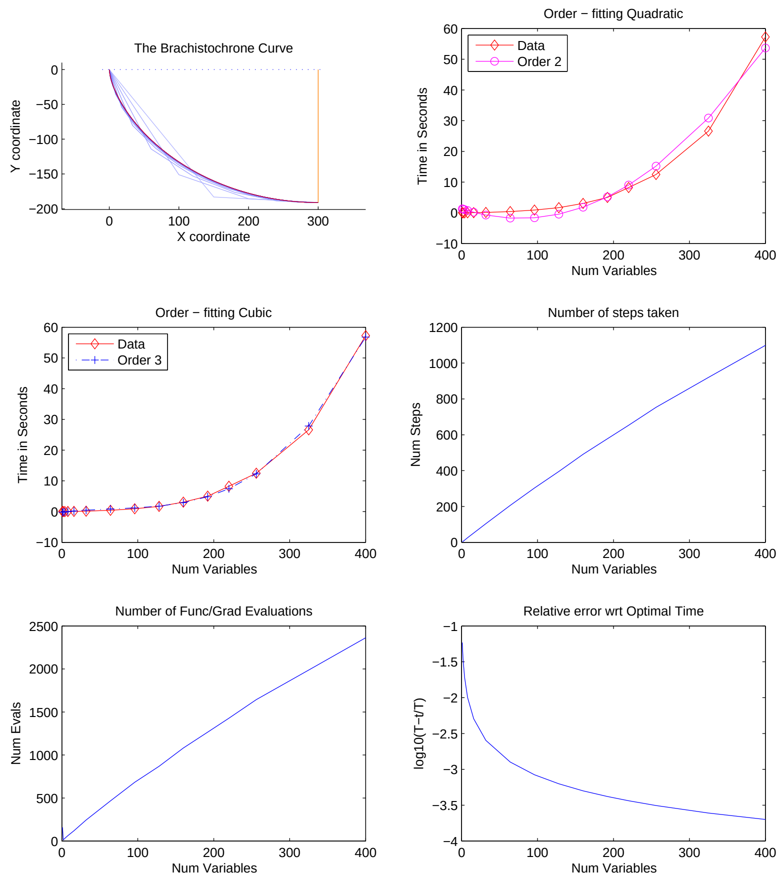

Figure 3.3: Destination (300, -191) with analysis

3. The third plot (Order - fitting Cubic) tries to fit a cubic to the running time for the algorithm vs. num variables.

4. The fourth plot (Number of steps) traces the number of steps vs. num variables.

5. The fifth plot (Number of function evaluations) gives the number of functions/gradient evaluations vs. num variables.

6. The sixth plot (Relative error) traces the relative error $(log_{10}(T-t)/T)$ as compared to the optimal time (T) vs. num variables.

Many more destination points are evaluated in the Appendix. We now show only the brachistochrones and along with the progress of the optimization for some more configurations with 99 variables.

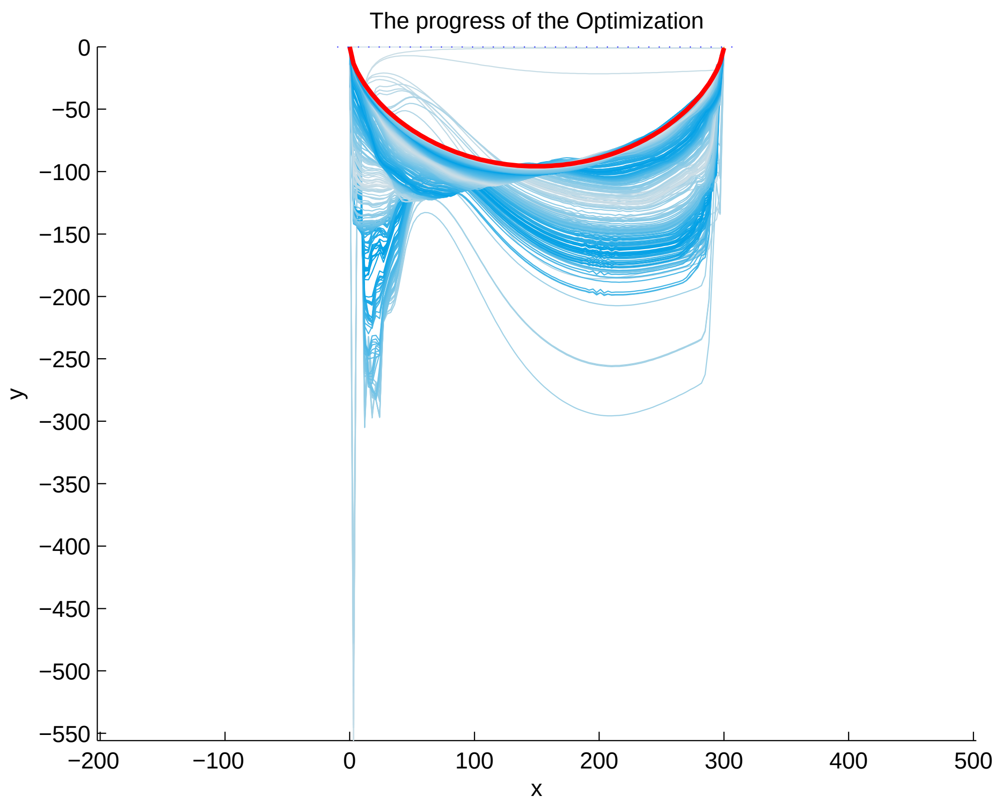

Figure 3.4: Destination (300, -1)

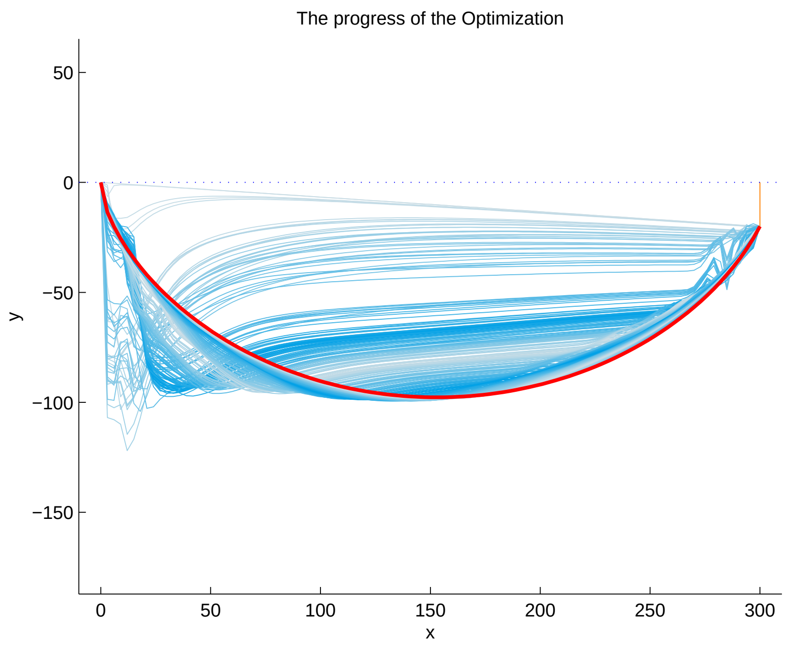

Figure 3.5: Destination (300, -20)

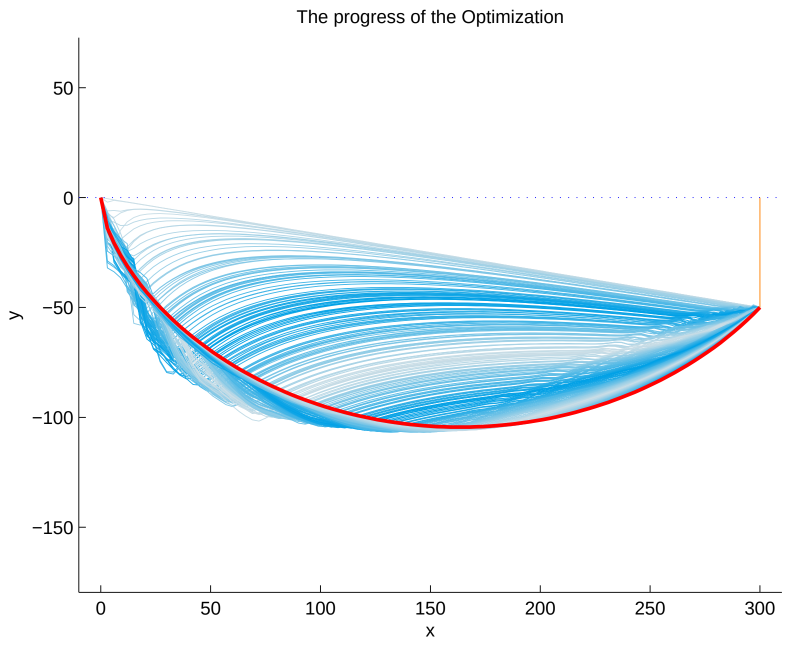

Figure 3.6: Destination (300, -50)

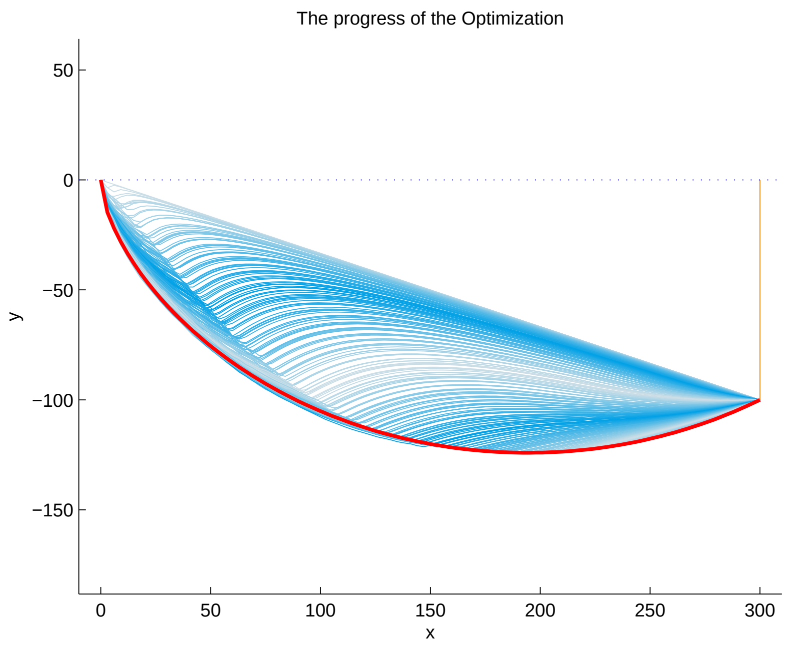

Figure 3.7: Destination (300, -100)

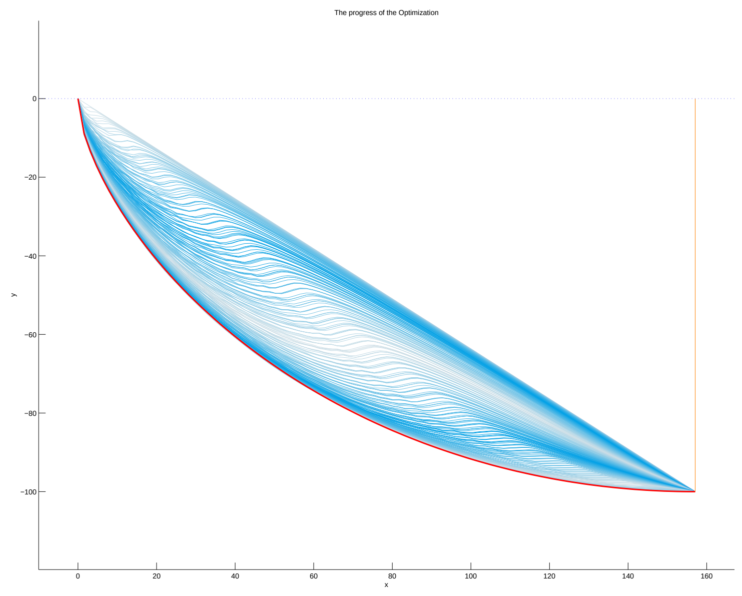

Figure 3.8: Destination (157, -100)

3.1.2 Varying initial conditions

Now we look at different starting curves and their effect on the convergence. Each of the curves have 99 variables and the starting values of the variables are in magenta.

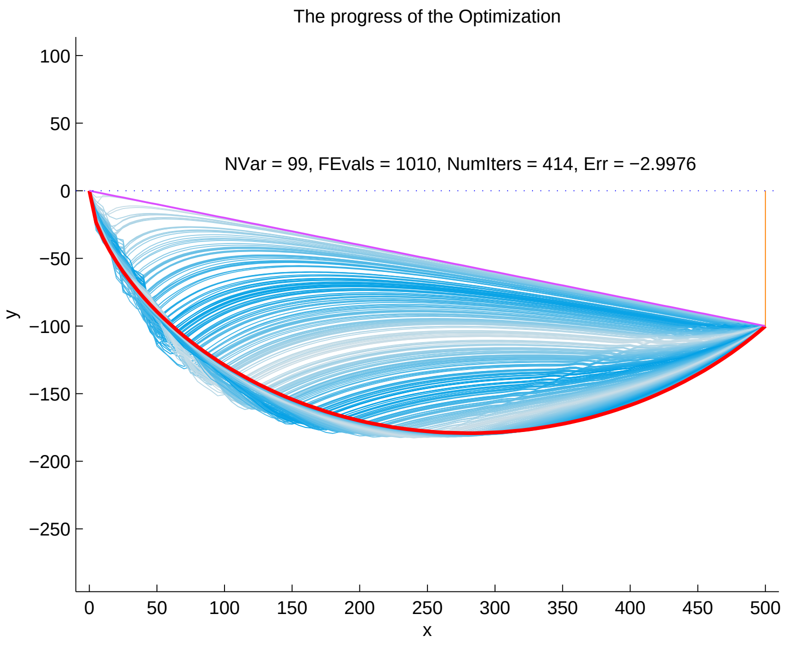

Figure 3.9: Destination (500, -100), Init - Linear curve

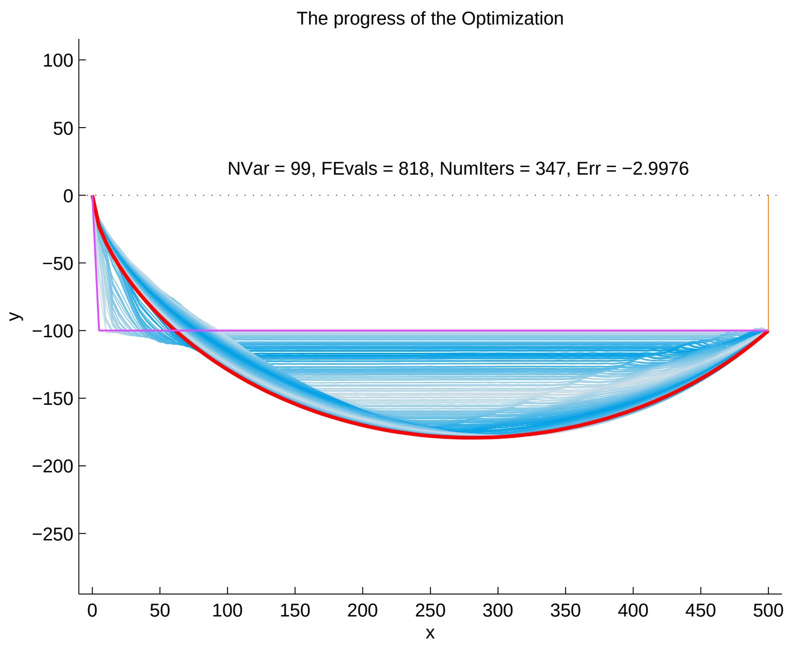

Figure 3.10: Destination (500, -100), Init - Flat curve

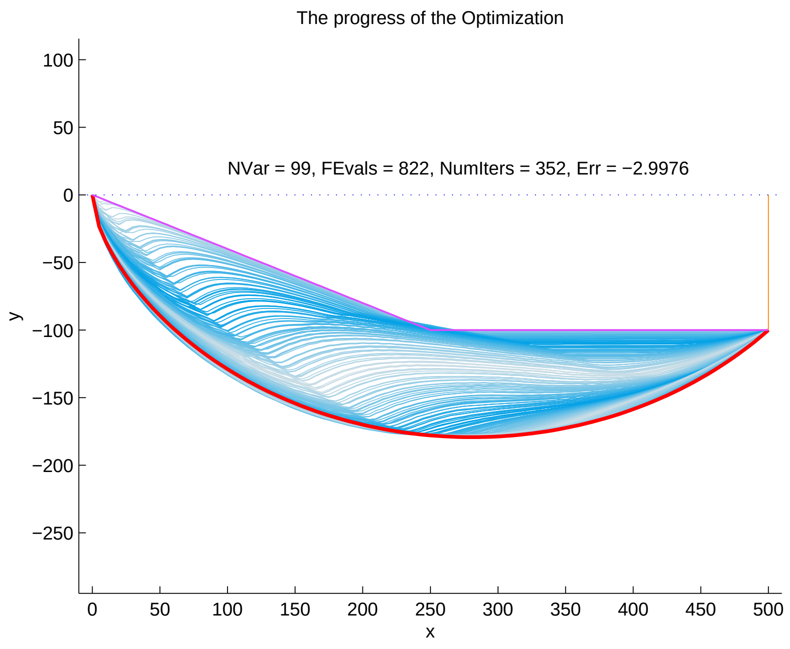

Figure 3.11: Destination (500, -100), Init - Linear/Flat curve

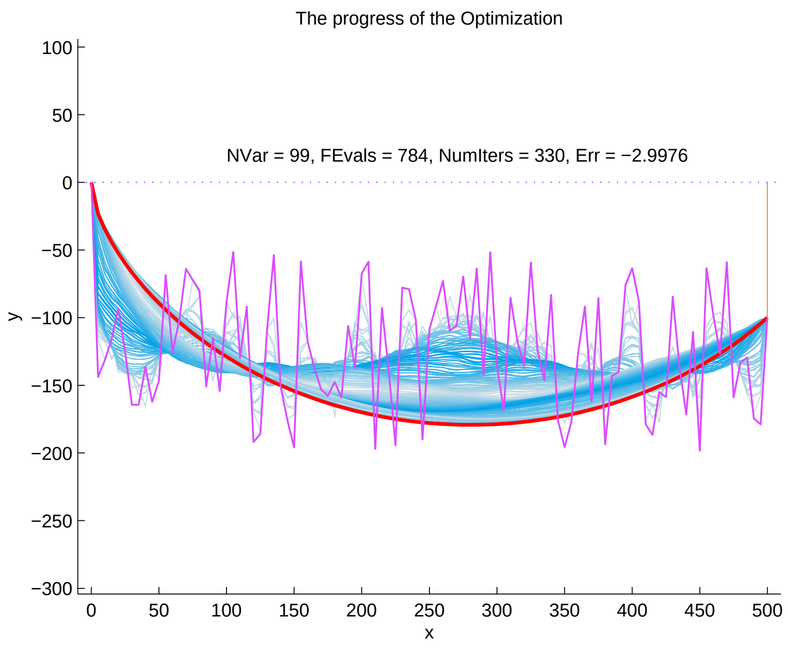

Figure 3.12: Destination (500, -100), Init - Random 1

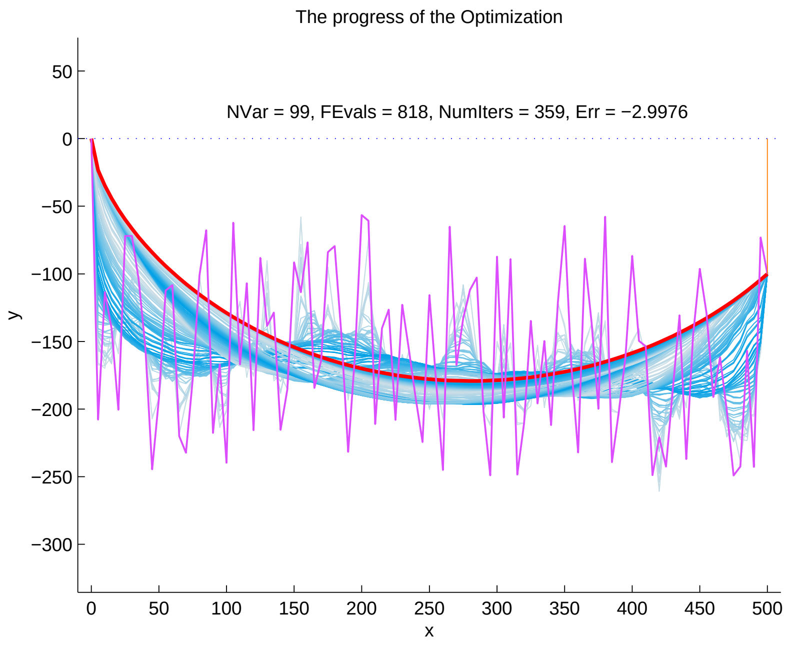

Figure 3.13: Destination (500, -100), Init - Random 2

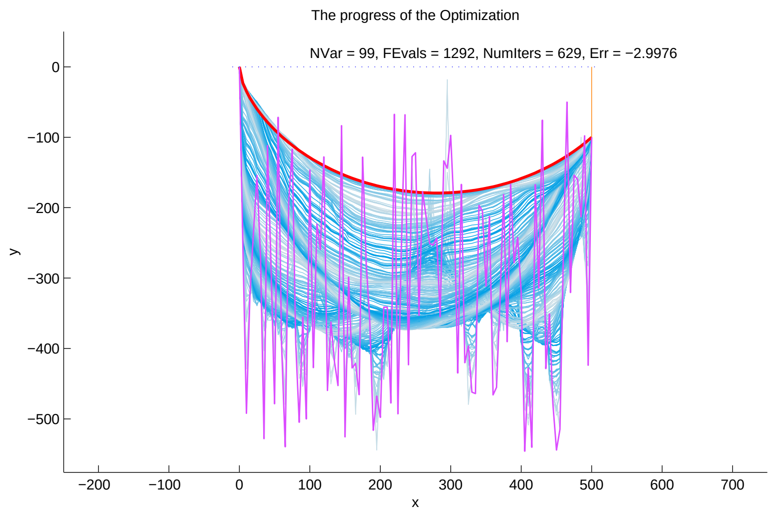

Figure 3.14: Destination (500, -100), Init - Random 3

We notice that the number of function evaluations does vary a bit with the initial conditions, but there seems to be a unique global minimum to which line search from all initial conditions invariably converges to. Random initial conditions perform reasonably well in terms of function evaluations. Also, error performance is independent of the initial condition, as expected.

3.1.3 Value of gravity

It is interesting to note that the value of the acceleration due to gravity does not have any effect on the final shape of the curve. This is expected from the analytical solution and is reassuring that the optimization solution also agrees. Of course, gravity does have an affect on the final optimal time. The value of acceleration due to gravity used is $10m/s^2$

3.2 Curves with Nonlinear Discretization

Using the knowledge about the shape of the brachistochrone from the uniform discretization and looking at the error performance of the various curves, we can easily see that we should have finer discretization when the curvature of the brachistochrone is large in order to have better performance with the same number of variables. A closer look at the brachistochrone near the origin shows the root of the problem. The linear approximation is most error prone near the origin where the curvature is largest.

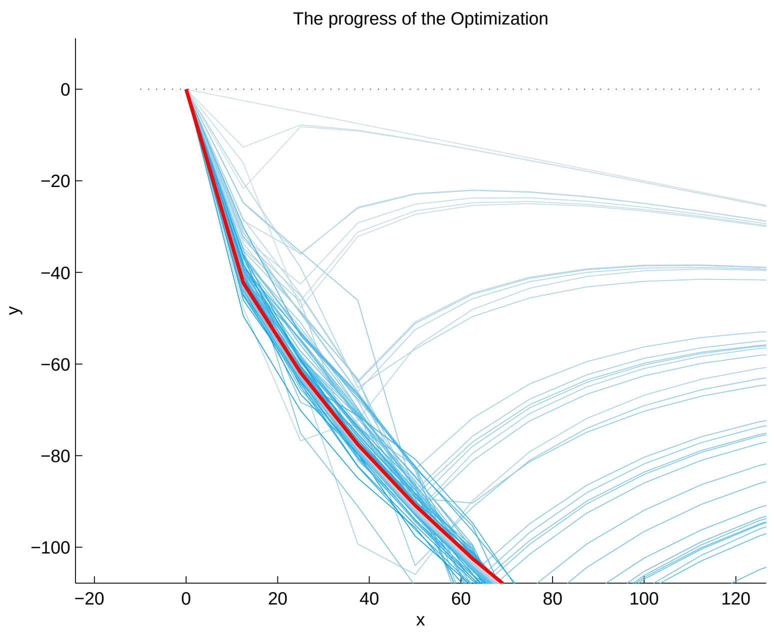

Figure 3.15: The source of the error

This can be mitigated by having smaller horizontal divisions near the origin and near the end, where the curvature tends to be larger. This is achieved by defining intervals of the form shown below

$$t = [0 : pi/(numvar + 1) : pi]';$$
$$x = L * (1 - cos(t))/2;$$

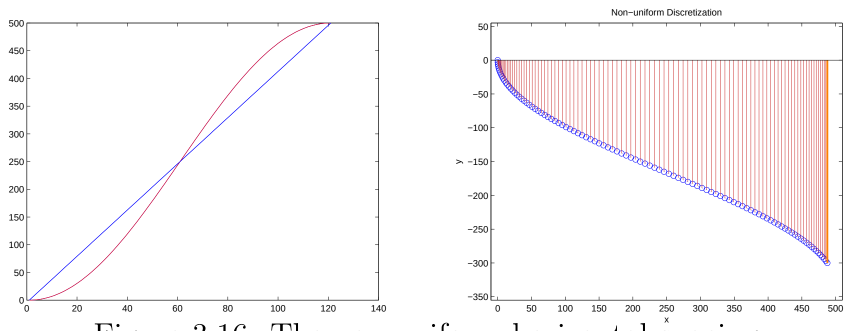

Figure 3.16: The non-uniform horizontal spacing

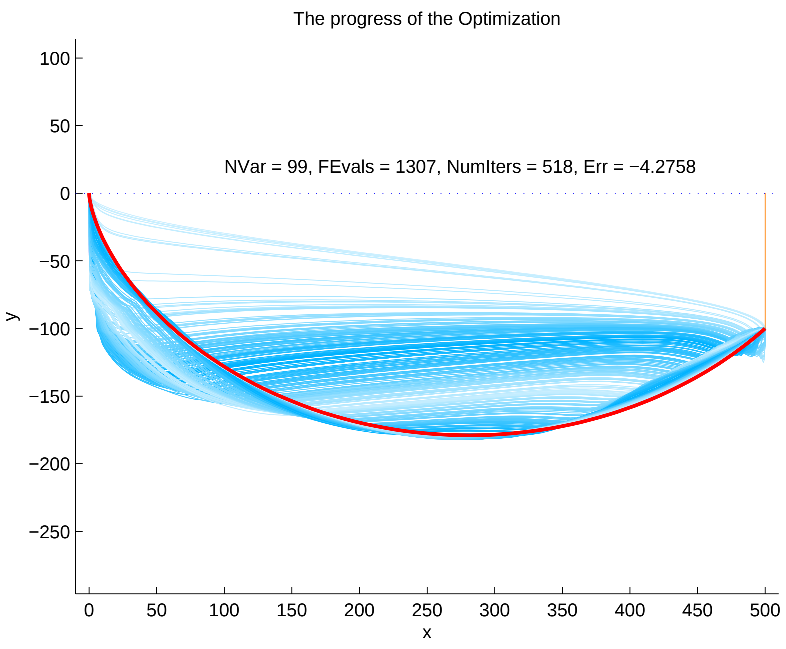

Figure 3.17: Brachistochrone with non-uniform spacing

We notice that this gives a good spacing near the origin and the end and an error performance which is better than the uniform spacing by at least an order of magnitude (-4.27 vs. -2.99). But sometimes, the fine spacing near the end is not always required. We can change the spacing using the knowledge of the curve near the optimal solution. For the final point under consideration, this suggests a non-uniform spacing of the form...

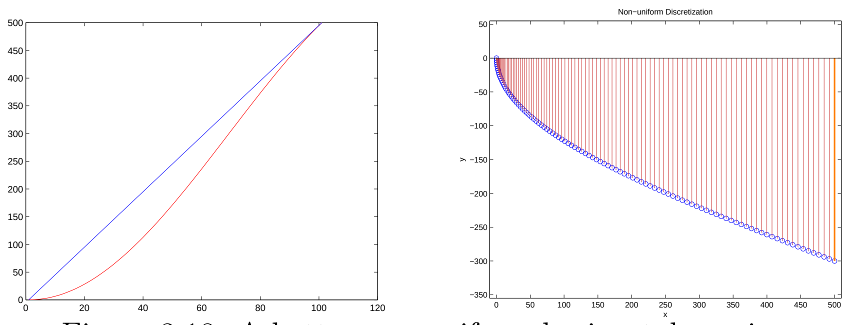

Figure 3.18: A better non-uniform horizontal spacing

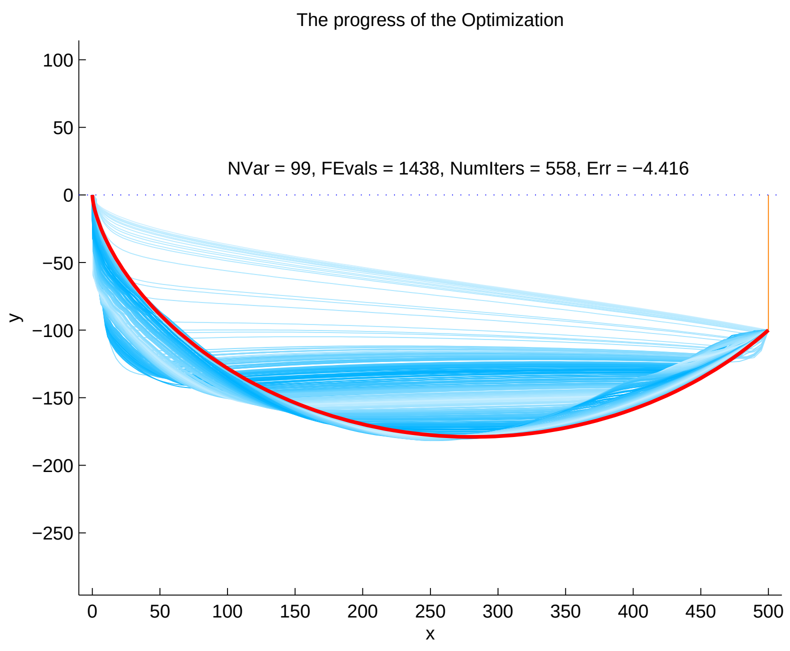

Figure 3.19: Brachistochrone with the better non-uniform spacing

We see that this gives a slightly better error performance (-4.4) for the same number of variables. We can see that the curve near the origin is much smoother.

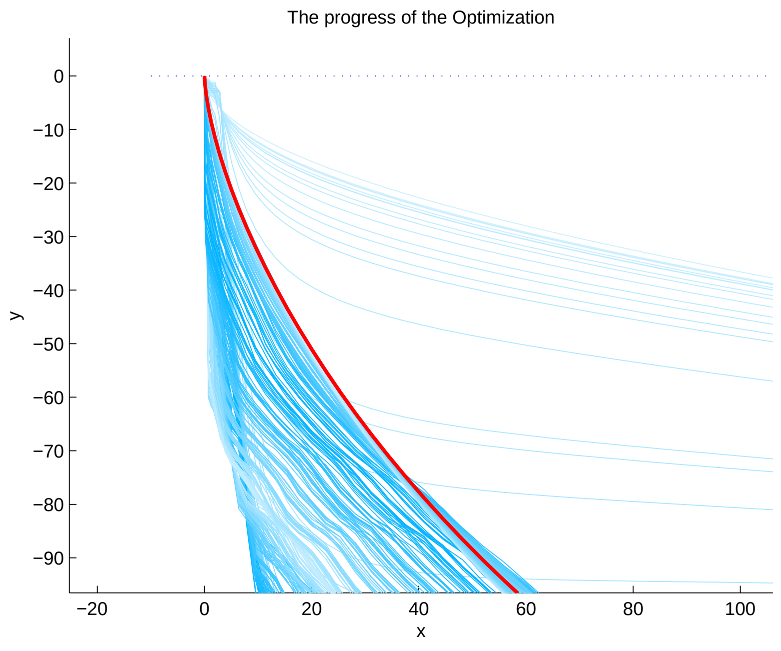

Figure 3.20: The curve near the origin

3.3 Brachistochrone in 2 dimensions

The solution of the brachistochrone in 2 dimensions gives an opportunity to see the contours of the function and the progress of the line search visually.

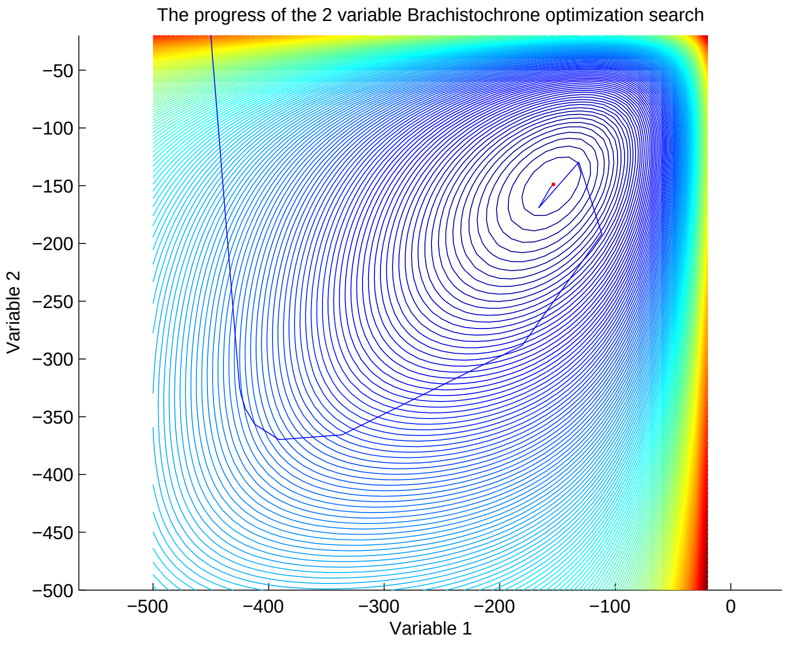

Figure 3.21: Brachistochrone in 2 variables

3.4 Brachistochrone with constraints

We now add a twist and solve the problem with a constraint. Suppose the the particle has to pass through a particular point en-route to the destination under the influence of gravity, what is the brachistochrone? Figure 3.24 and 3.25 show two such paths with different constraints.

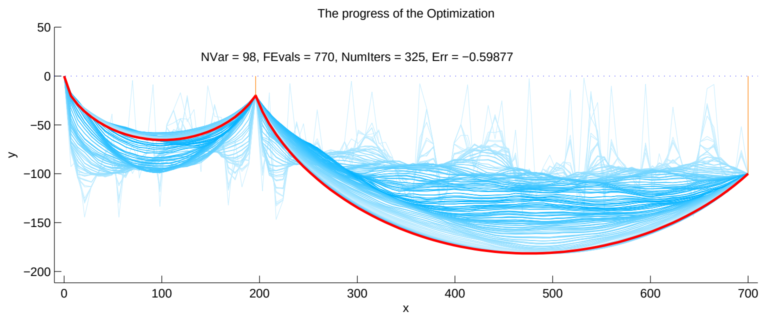

Figure 3.22: Brachistochrone with constraint (random init)

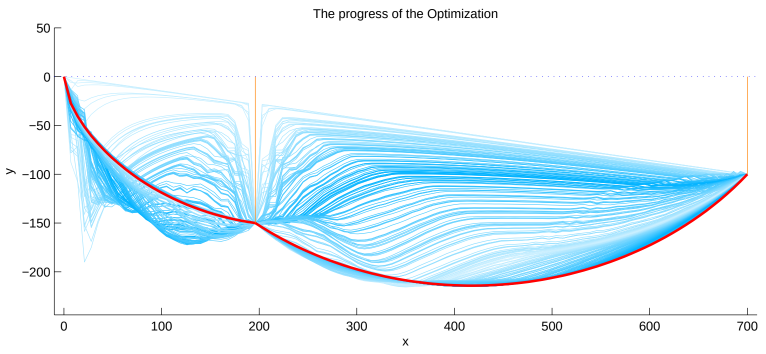

Figure 3.23: Brachistochrone with constraint

It turns out that adding a constraint also answers another question - What is the brachistochrone if the particle does not start at rest but has an initial velocity (magnitude is specified)? The question can be answered by having a constraint as the first point after the origin, such that the difference in height between origin and constraint is deduced from the conservation of energy.

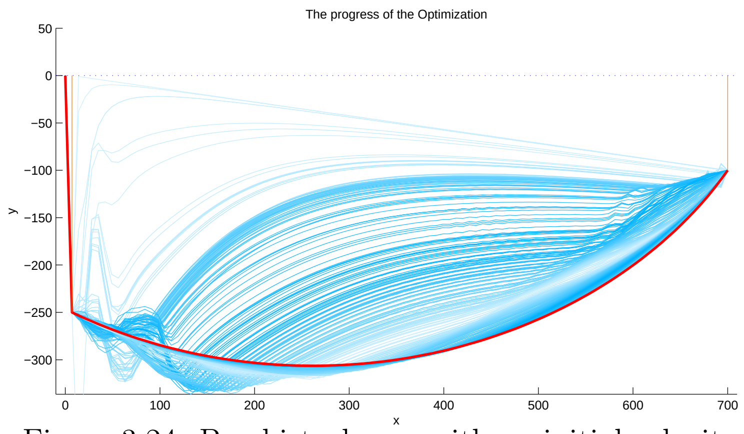

Figure 3.24: Brachistochrone with an initial velocity

The destination point can now be higher than the starting location, but it still has to be lower than the origin. As expected we see that the point goes below its present position before rising to the destination point.

Chapter 4

Analysis

4.1 Order of algorithm

A study of the graphs given in the appendix shows that the algorithm runs in $\bigcirc(n^3)$ and is at least as fast as matrix inversion solution of an N dimensional quadratic problem (assuming we dont really "invert" the matrix, but solve by factorization). The overall running time is very good considering the fact that the objective function is not quadratic.

4.2 Algorithm accuracy

We saw that non-uniform discretization gives better results than uniform discretization for the same number of variables. The relative accuracy of the uniform-discretization is in the range of 10^{-3} to 10^{-5}, the model being most accurate when the optimal path is almost a straight line. The accuracy of the approximate curve becomes better as the number of variables is increased. This can be also be seen from the graphs in the appendix.

Non-uniform discretization gives an accuracy which is at least an order of magnitude better than uniform discretization. The comparison is illustrated in Table 4.1.

4.3 Non-uniform Discretization

Non uniform discretization is observed to give a better approximation to the analytical solution. The error is at least better by an order of magnitude. Though a good non-uniform discretization requires the knowledge of the curve near the optimal solution, this can be achieved by first solving

θ	x	y	err_{ud}	err_{nud}
0.4235	71	-500.0	-4.6286	-6.3224
1.0570	183	-500.0	-3.8531	-5.5364
1.6833	310	-500.0	-3.4842	-5.1477
2.3123	470	-500.0	-3.2605	-4.8941
2.9458	500	-360.0	-3.1201	-4.7124
3.5675	500	-240.0	-3.0397	-4.5789
4.1994	500	-147.0	-3.0025	-4.4730
4.8278	500	-76.0	-3.0001	-4.3839
5.4574	500	-26.0	-3.0179	-4.2898
6.0823	500	-1.6	-2.9331	-4.1137

Table 4.1: Error: uniform (err_{ud}) vs. non-uniform discretization (err_{nud})

the problem with uniform discretization and then using the solution as the starting point for the non-uniform discretization.

4.4 Path of fastest Ascent for an aeroplane

The result from the "Brachistochrone with an inital velocity" suggests that a jet which would like to ascend to some height in the minimum time should follow the cycloidal path. Of course, this assumes that the jet does not accelerate and only expends fuel to offset the friction.

Appendix A

Further Thoughts

A.1 Analytical Hessian

The hessian can be computed analytically in case the approximate hessian is becoming ill conditioned near the solution. Also, it can be easily seen that the analytical hessian is tri-diagonal and hence is not very expensive to compute.

A.2 Optimal Discretization

We showed that a uniform discretization in the horizontal direction is not optimal and a better solution is given by a non-uniform discretization based on the curvature of the brachistochrone. In order to reach the most optimal solution possible with 'n' variables, we need to optimize over the space of possible horizontal divisions. This just adds another n variables to the problem but is very similar to the original problem and can be solved by similar techniques.

A.3 Curve with constant acceleration

The question, "If a constant acceleration is applied by a jet while ascending to some height, what is the curve of fastest ascent?", can also be solved by applying similar optimization techniques to an appropriate objective function. The solution of the brachistochrone problem also suggests the following hypothesis...

Suppose there are 2 jet planes (same acceleration and capability) flying at the same height with same speed, which are racing each other to reach a point x km away at the same horizontal level. The one which accelerates directly towards the point and takes a horizontal path shall not necessarily win. The path of shortest time actually requires the jet to dip to a lower level in order to speed up using gravity and then come back up. Even under acceleration, the curve of shortest time is never a straight line, unless the destination point is directly above or below.

A.4 Optimal path for space rocket

The optimization technique used in this problem can be extended to find the optimal path for a rocket. We should keep in mind that there are lot more variables in this problem, e.g. acceleration, gravity and mass are changing (which probably leads to variable acceleration). Also, friction is changing as the rocket rises higher. Although there are more variables, the nature of the optimization problem remains essentially the same and can be attacked using the techniques described.

Appendix B

Outputs

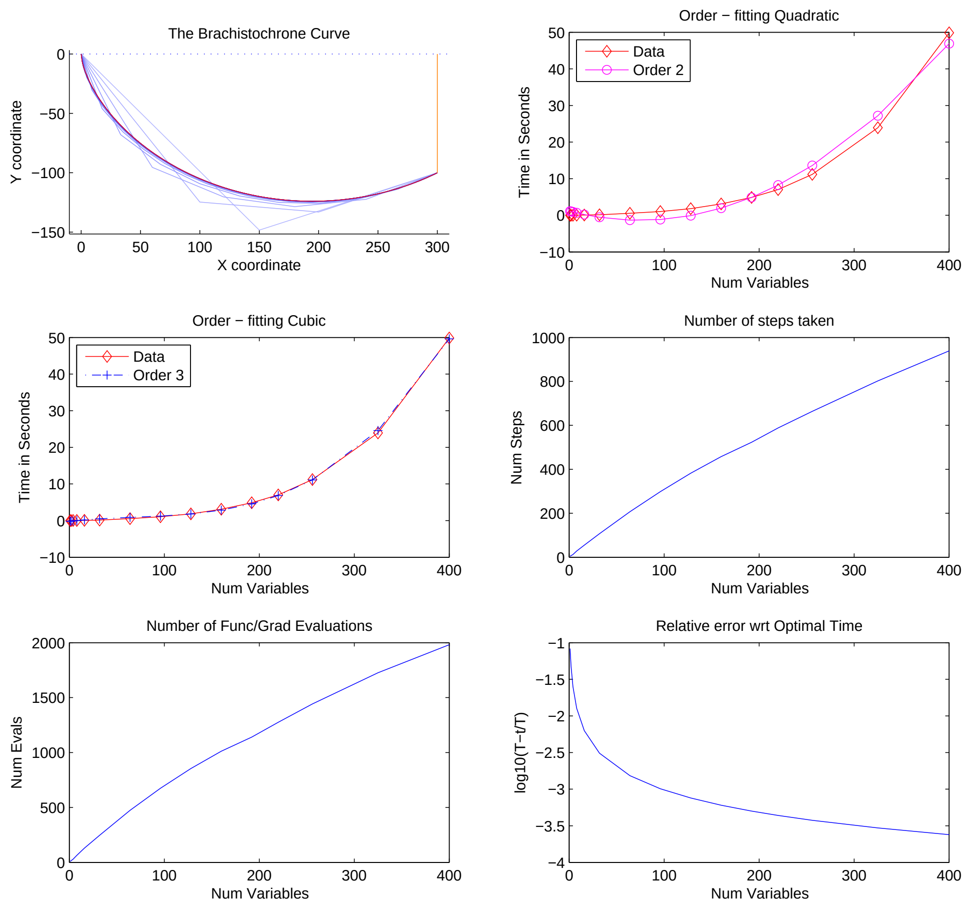

Figure B.1: Destination (300, -100) with analysis

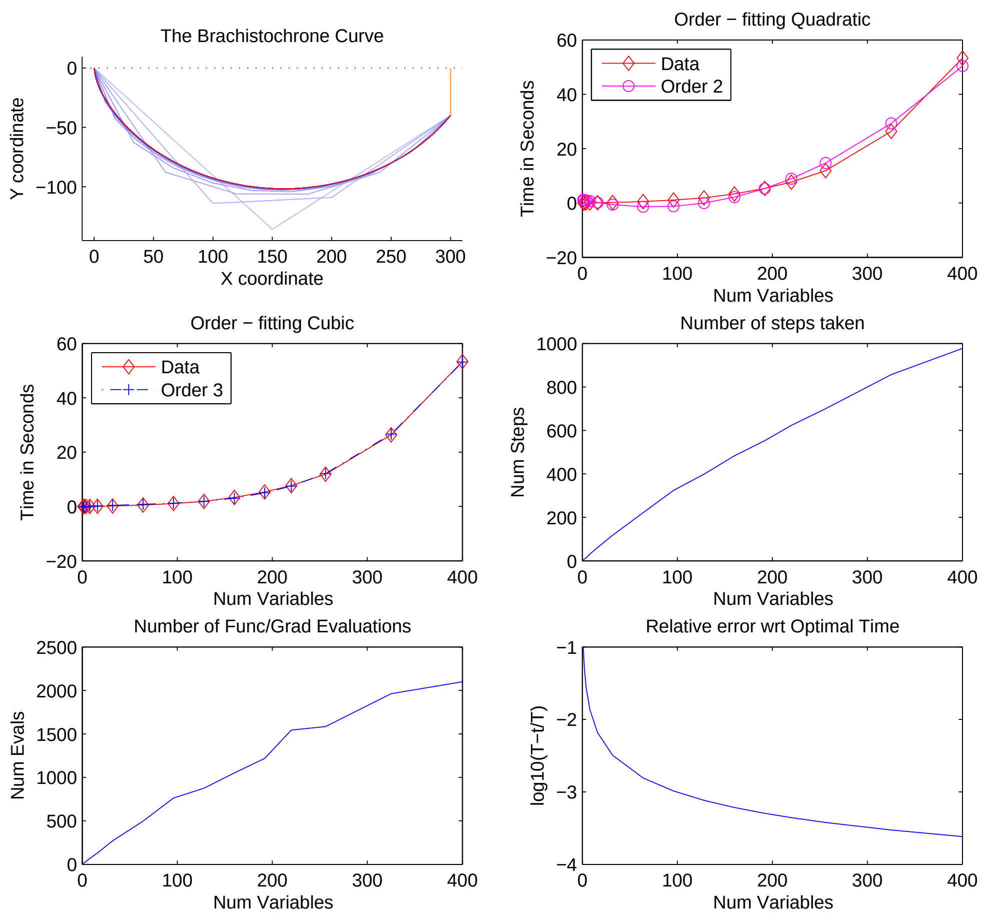

Figure B.2: Destination (300, -40) with analysis

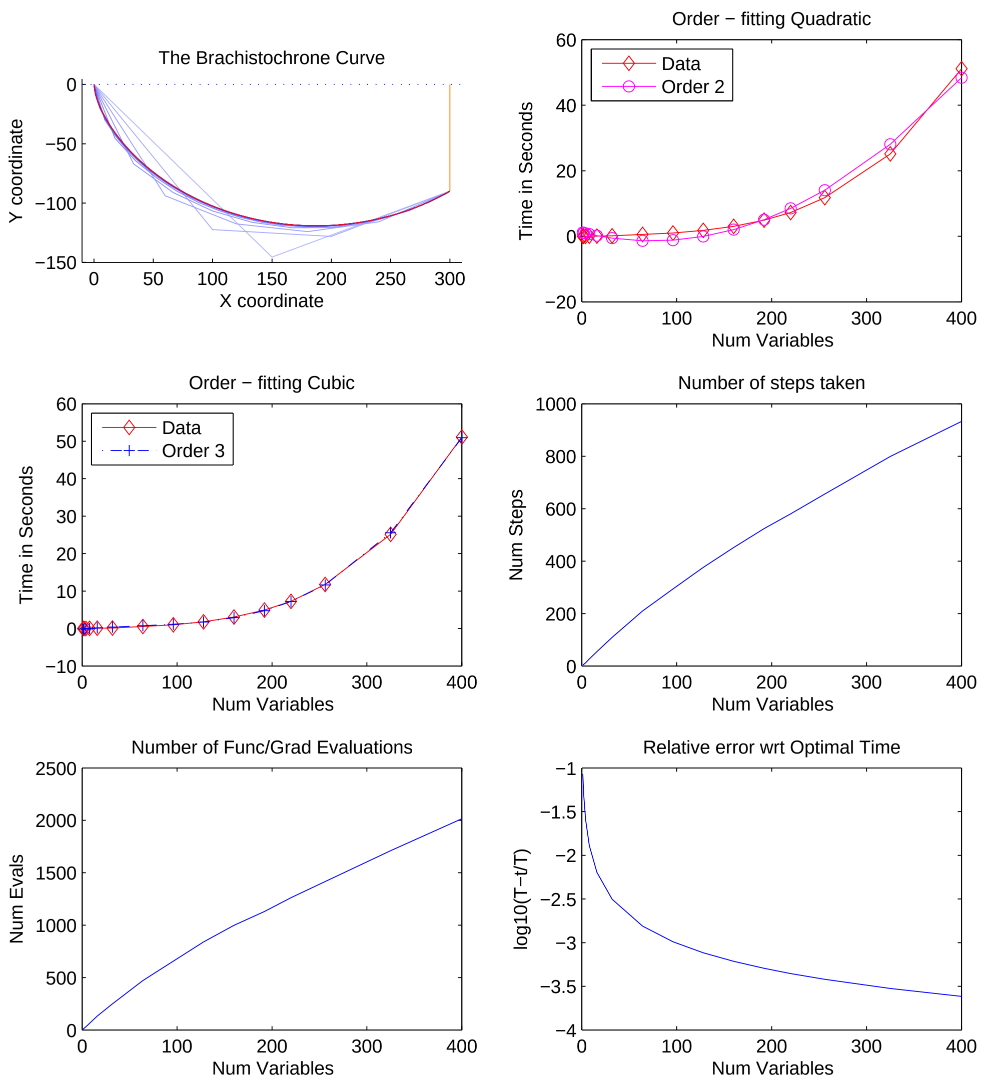

Figure B.3: Destination (300, -90) with analysis

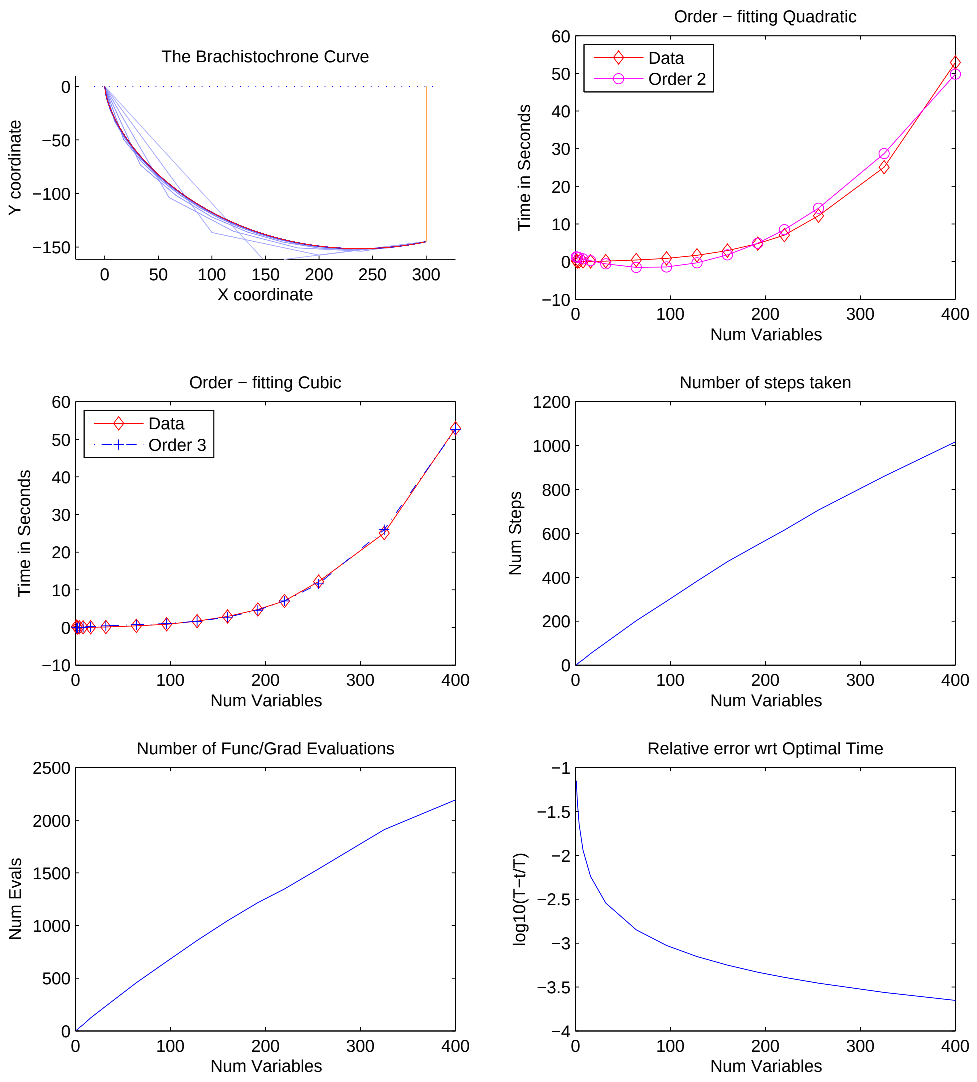

Figure B.4: Destination (300, -145) with analysis

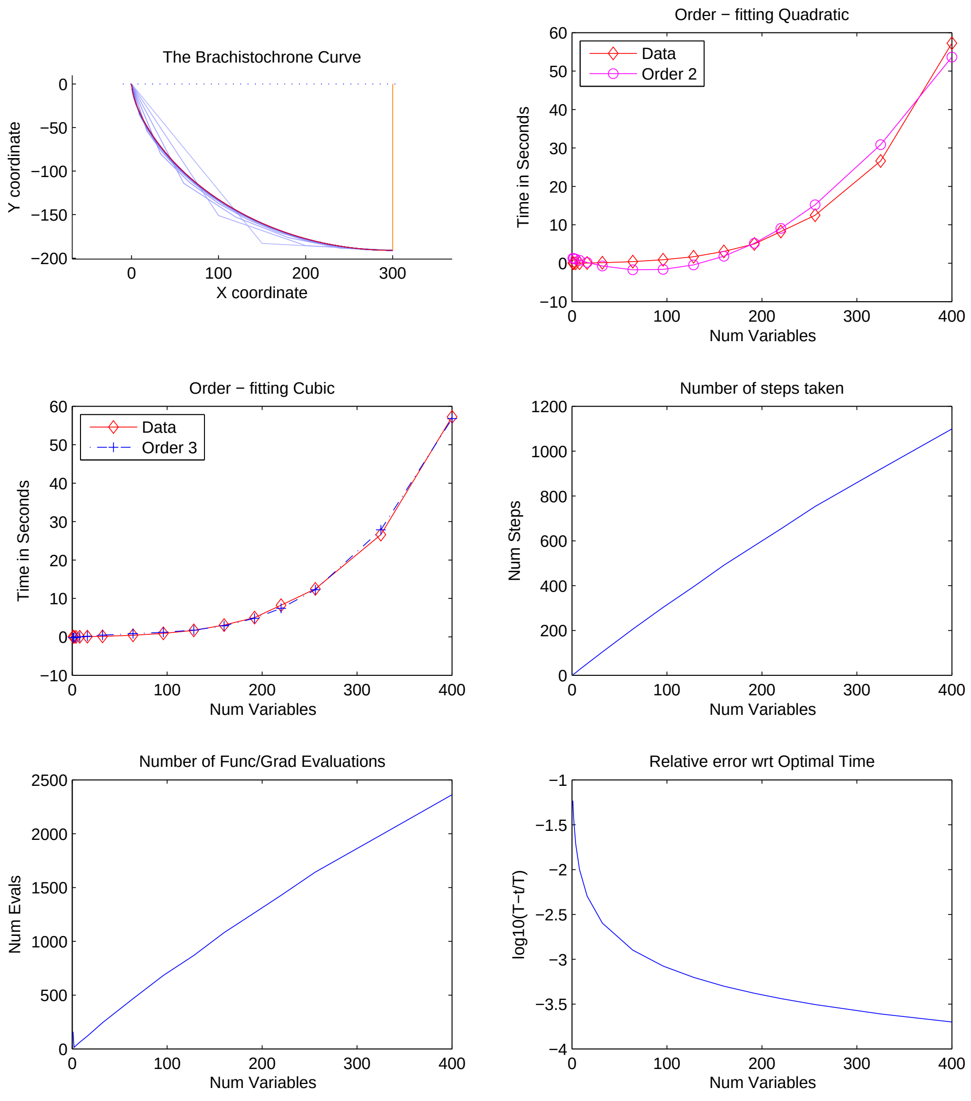

Figure B.5: Destination (300, -191) with analysis

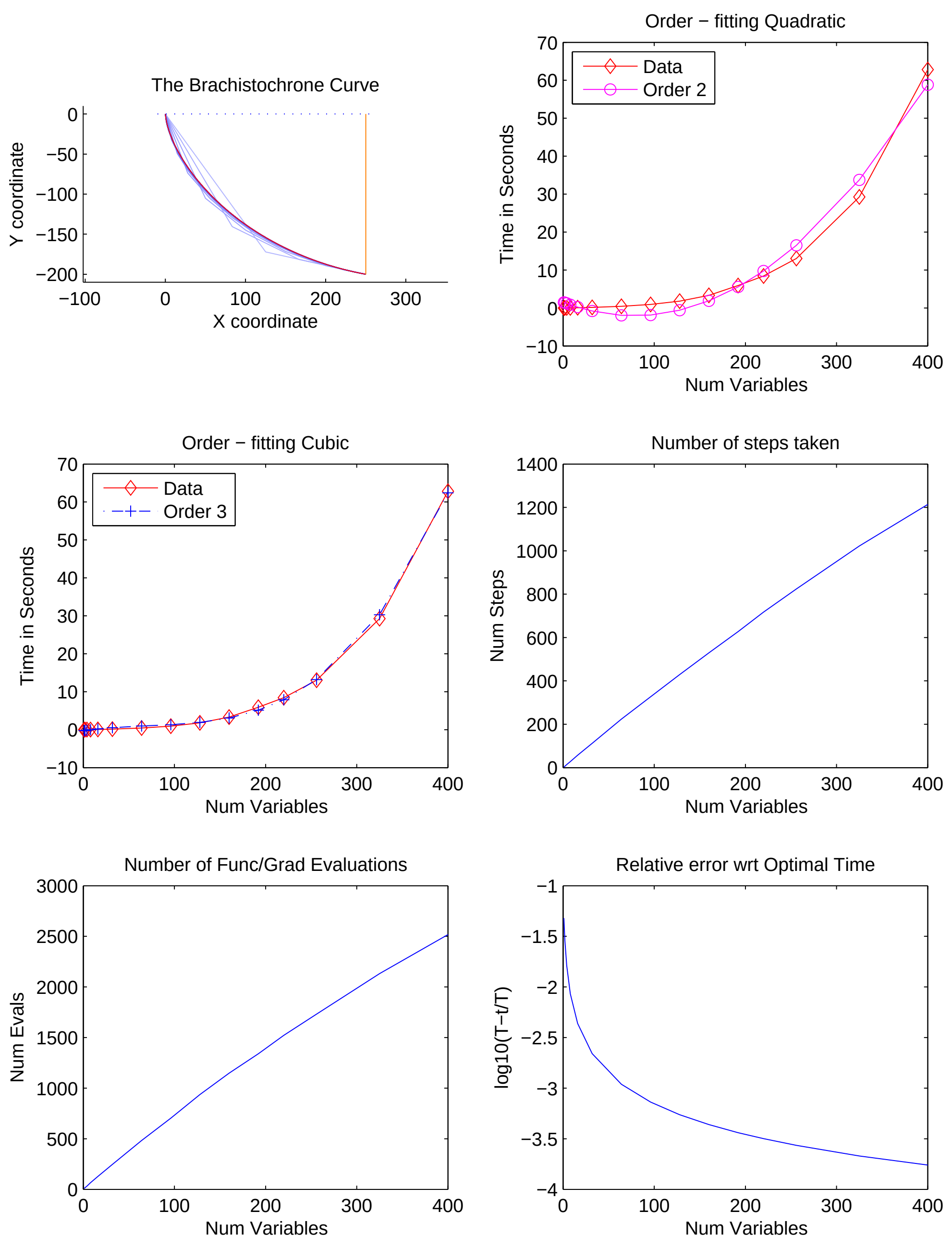

Figure B.6: Destination (250, -200) with analysis

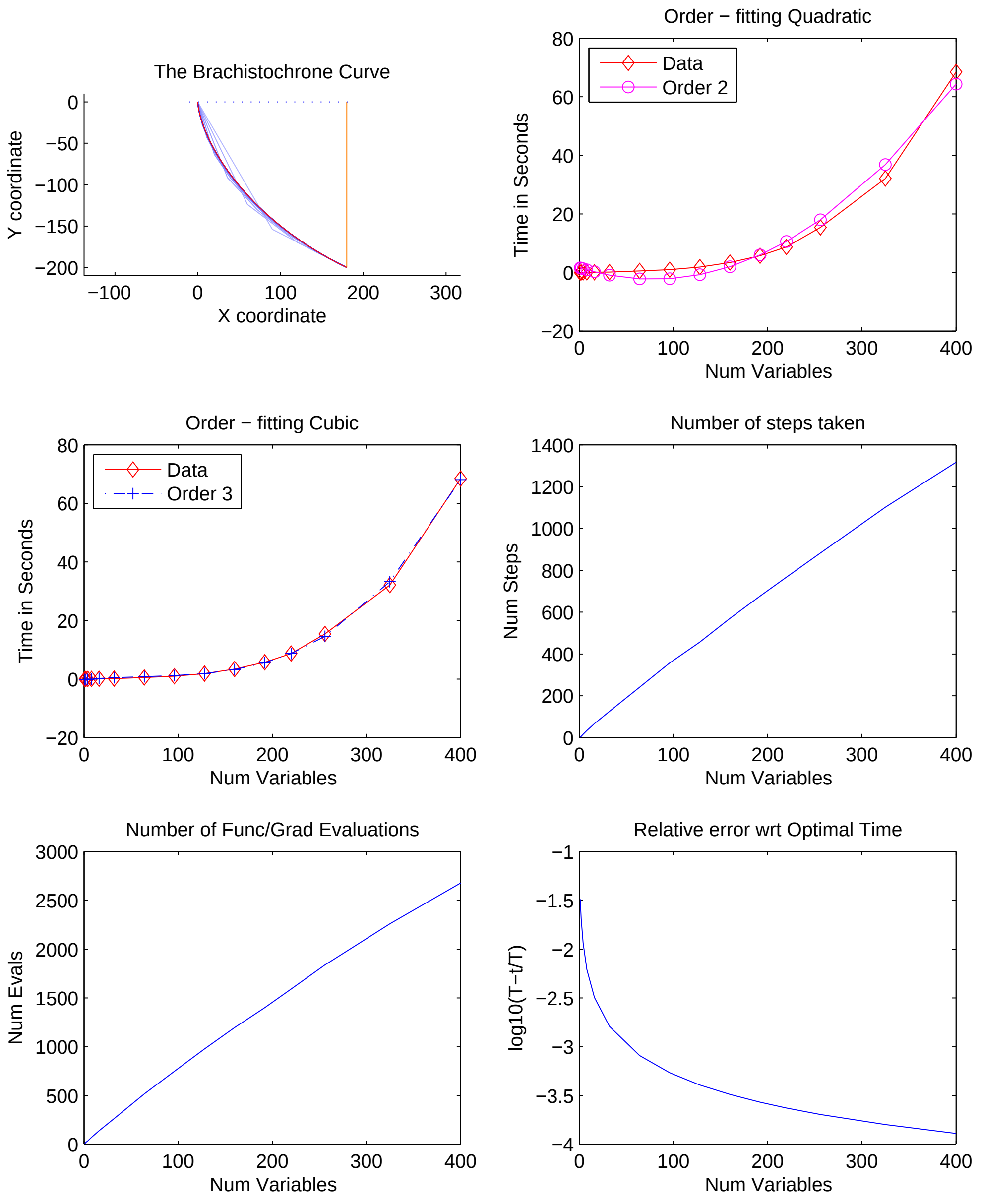

Figure B.7: Destination (180, -200) with analysis

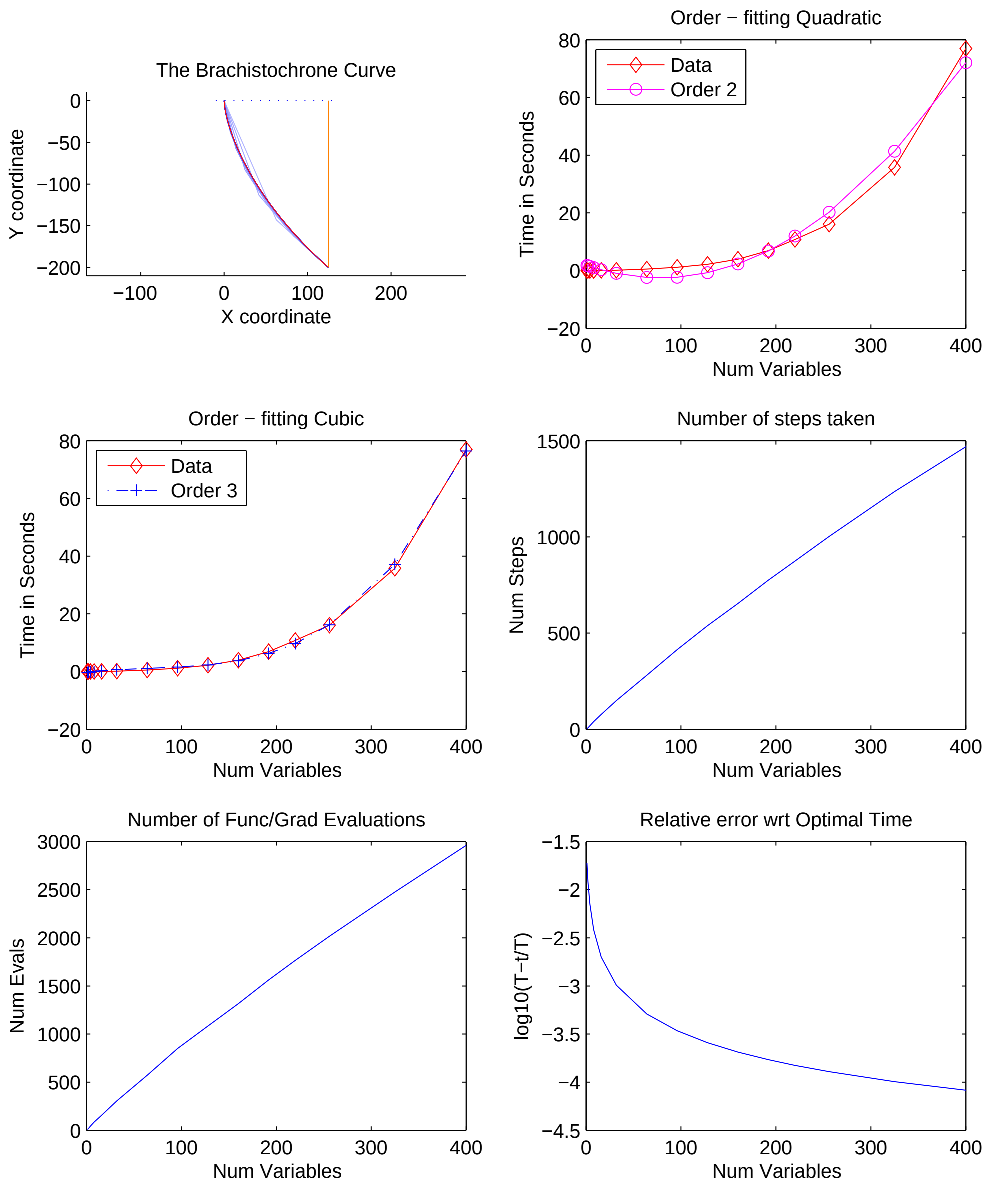

Figure B.8: Destination (125, -200) with analysis

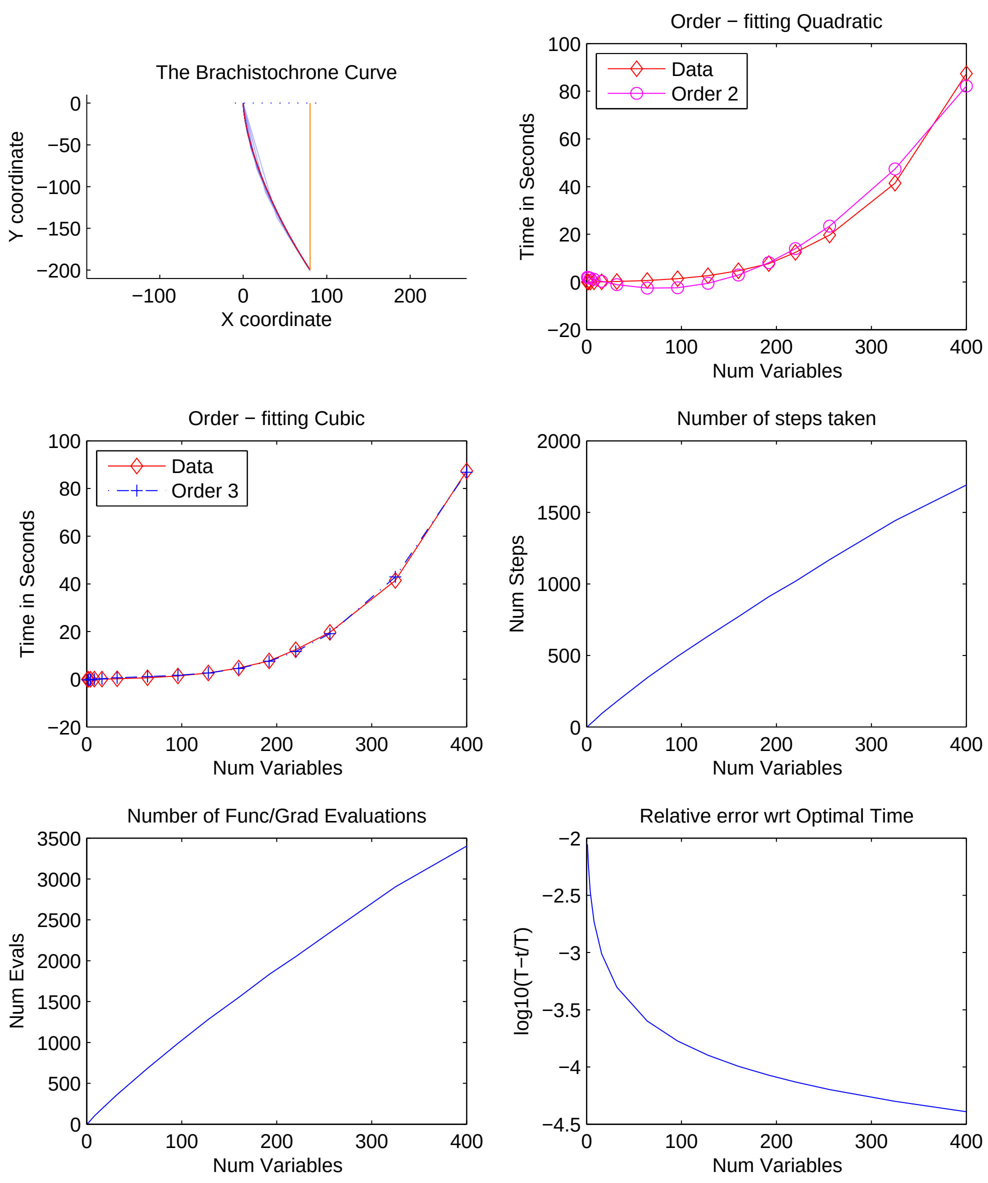

Figure B.9: Destination (80, -200) with analysis

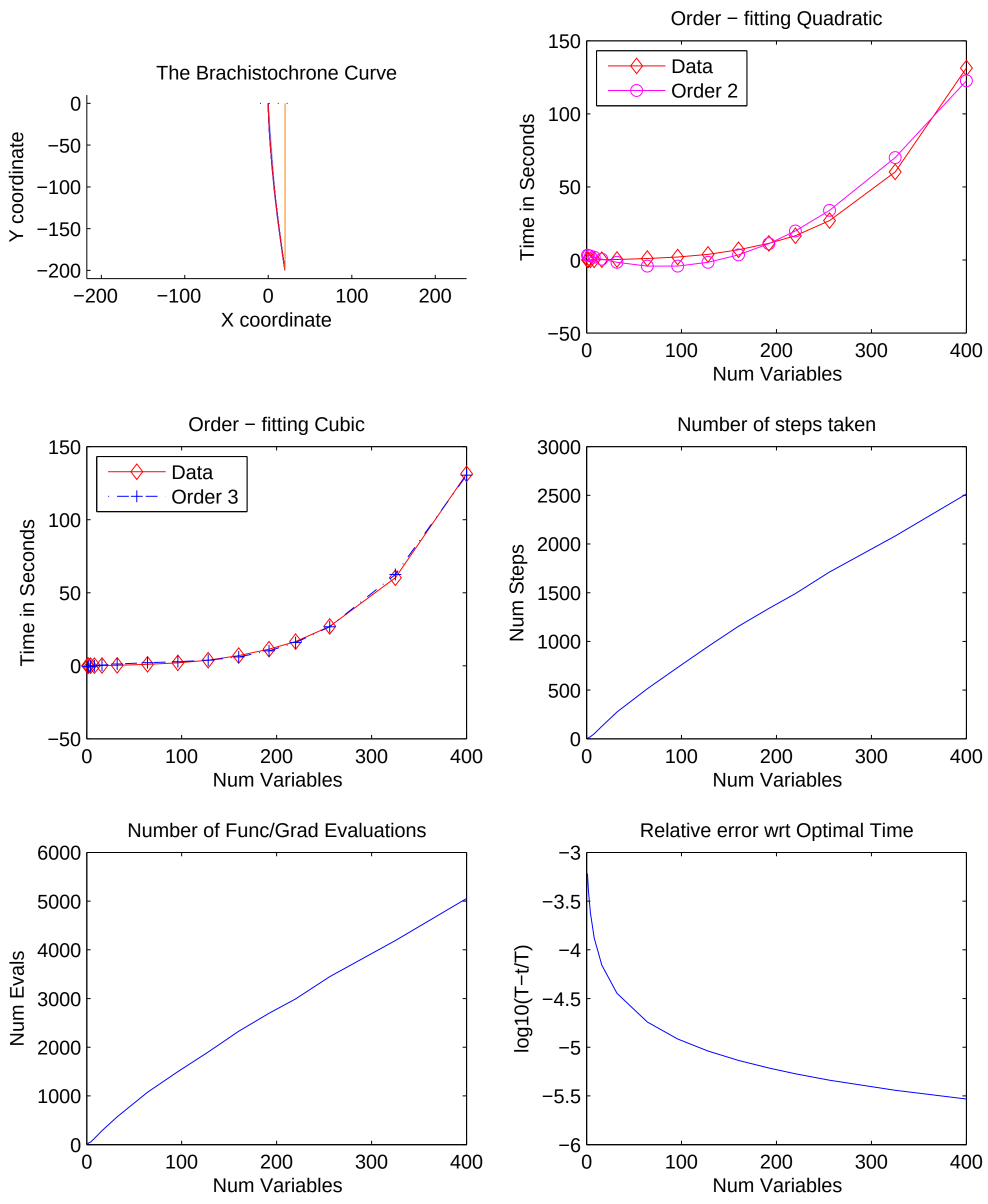

Figure B.10: Destination (20, -200) with analysis