Theoretical Background for Aerodynamic Shape Optimization

John C. Vassberg *
JetZero, Inc.
Corporate Headquarters
Long Beach, CA 90808, USA

Antony Jameson †
Department of Aerospace Engineering
Texas A&M University
College Station, TX 77843, USA

Von Karman Institute
Brussels, Belgium
17 May, 2022

Nomenclature

A	Hessian Matrix / Operator
a	Integrand of Integral Hessian Operator
C	Constant; Constrain equation
$\mathcal{E}$	Error of Discretization
F	Integrand of Cost Function
$\mathcal{G}$	Gradient of Cost Function
$\bar{\mathcal{G}}$	Implicitly-Smoothed Gradient
g	Acceleration due to Gravity $\simeq 32.2 \frac{ft}{sec^2}$
$GRMS$	Root-Mean-Square of Gradient Vector
H	Estimate of Inverse Hessian Matrix
I	Objective or Cost Function
I_R	Discrete I based on Rectangle Rule
$ITERS$	Iterations needed for 10^{-6} Reduction
i	Matrix Row Index
j	Spatial Index; Matrix Column Index
k	Generic Index
K	Upper Limit of Generic Index k
$\mathcal{L}$	Lagrangian Dual Cost Function
N	Number of Design Variables
n	Iteration Index
$NMESH$	Number of Multigrid Levels
NX	Number of Mesh Intervals
P	Multigrid Forcing Function
$\mathcal{P}$	Error Vector of Rank-One Scheme
Q	Multigrid Transfer Operator
$STEP$	Input Parameter Similar to CFL Number
$Surplus$	$I_R^C - I_R^D$
S	Geodesic Path Length
s	Arch Length
T	Total Time; Multigrid Transfer Operator
t	Parametric Variable; Time
v	Velocity
X	Design Variable
x	Independent Spatial Variable
$\mathcal{X}$	Design Space Vector $[X, Y, Z]^T$
Y	Design Variable
y	Design Variable
$YERR$	L_2 Norm of $y - y_{exact}$ vector
Z	Design Variable
α	Tridiagonal Coefficient of Implicit Scheme
β	Coefficient of Parabolic Term
γ	Recombination Coefficients
ϵ	Implicit Smoothing Parameter
λ	Time-Step Parameter
μ	Lagrange Multiplier
ξ	Free Variable
π	3.141592654...
∞	Infinity
$\delta*$	First Variation of
$\Delta*$	Change in
$\nabla*$	Gradient of
$\partial*$	Partial of
$\mathcal{O}(*)$	Order of
$(*)^T$	Matrix Transpose of
$(*)^{-1}$	Inverse Matrix of
$*'$	First Derivative of, $\frac{\partial}{\partial x}$
$*''$	Second Derivative of, $\frac{\partial^2}{\partial x^2}$

*Chief Design Officer
†TEES Eminent Professor

1 Introduction

This is the first of three lectures prepared by the authors for the von Karman Institute that deal with the subject of aerodynamic shape optimization. In this lecture we introduce some theoretical background on optimization techniques commonly used in the industry, apply some of the approaches to a couple of very simple model problems, and compare the results of these schemes. We also discuss their merits and defficiencies as they relate to the class of aerodynamic shape optimization problems the authors deal with on a daily basis. In the second lecture, we provide a set of sample applications, while the third lecture is focused on parameterization of the design space. However, before we continue with the simple model problems of this lecture, let's first review some properties of the aerodynamic shape optimization studies the authors are regularly concerned with.

In an airplane design environment, there is no need for an optimization based purely on the aerodynamics of the aircraft. The driving force behind (almost) every design change is related to how the modification improves the vehicle, not how it enhances any one of the many disciplines that comprise the design. And although we focus our lectures on the aerodynamics of an airplane, we also include the means by which other disciplines are linked into and affect the aerodynamic shape optimization subtask; these will be addressed in detail in the second lecture. Another characteristic of the problems we typically (but not always) work, is that the baseline configuration is itself within 1-2% of what may be possible, given the set of constraints that we are asked to satisfy. This is certainly true for commercial transport jet aircraft whose designs have been constantly evolving for the past half century or more. This class of problems is much more demanding than those in which the baseline is far from the optimum design. Frequently, it is easier to show a 25% improvement relative to a baseline that is 30% off optimum than it is to realize a 1% gain on a starting configuration that only has 2% to give.

Quite often the problem is very constrained; this is the case when the shape change is required to be a retrofitable modification that can be applied to aircraft already in service. Occasionally, we can begin with a clean slate, such as in the design of an all-new airplane. And the problems cover the full spectrum of studies in between these two extremes. Let's note a couple of items about this setting. First, in order to realize a true improvement to the baseline configuration, a high-fidelity and very accurate computational fluid dynamics (CFD) method must be employed to provide the aerodynamic metrics of lift, drag, pitching moment, spanload, etc. Even with this, measures should be taken to estimate the possible error band of the final analyses: this discussion is beyond the scope of these lectures. The second item to consider is related to the definition of the design space. A common practice is to use a set of basis functions which either describe the absolute shape of the geometry, or define a perturbation relative to the baseline configuration. In order to realize an improvement to the baseline shape, the design space should not be artificially constrained by the choice of the set of basis functions. This can be accomplished with either a small set of very-well-chosen basis functions, or with a large set of reasonably-chosen basis functions. The former approach places the burden on the user to establish an adequate design space; the latter approach places the burden on the optimization software to economically accommodate problems with large degrees of freedom. Over the past two decades, the authors have focused on solving the problem of aerodynamic shape optimization utilizing a design space of very large dimension. The interested reader can find copious examples of the alternative approaches throughout the literature.

With some understanding of where we are headed, let's now return to the simple model problems included herein, review various aspects of the optimization process, and discuss how these relate to the aerodynamic shape optimization problem at hand. The first model problem introduces some of the basics; the second one is a classic example in mathematical history.

2 The Spider & The Fly

In our first model problem, we will discuss how to set up a design space, how to numerically approximate the gradient and the Hessian matrix, how to impose active constraints, and how to navigate this design space from an initial state towards a local optimum using gradient-based search methods. We will also talk about some *traps* to avoid when setting up a problem of optimization.

The original *spider and fly* problem was first introduced by Dudeney [1] in 1903. In our version of the spider-fly problem, we have a wooden block with dimensions of 4 *in* wide, by 4 *in* tall, by 12 *in* long; the bottom of this block is resting on a solid flat surface. See Figure 1. On one of the square ends sits a spider,

located 1 *in* from the top and centered left-to-right. On the opposite side a fly is trapped in the spider's web; the fly is located 1 *in* from the bottom and centered left-to-right. The spider considers the path where he would initially travel 1 *in* straight up to the top, then 12 *in* axially across the top face, then 3 *in* downward to the fly; the length of this path is 16 *in*. This path represents a local optimum path.

As it turns out, the spider was a mathematician in a former lifetime, so he wonders if this is the global minimum-length path possible. In order to solve this enigma and determine the true geodesic, the spider sets up a problem of optimization. To cast this optimization problem into a mathematical formulation, the spider must some how constrain his motion to the surface of the wooden block, and furthermore, he knows he cannot traverse the bottom side of the block as it is resting on the solid flat surface. It is clear that the aforementioned 16 *in* path is a local optimum, and due to the symmetry of the problem, there are really only two other types of paths that need to be studied. In one type, the spider moves laterally 2^+ *in* toward the right/left side of the block, then 12^+ *in* across that side towards the back, then 2^+ *in* to the trapped fly. Here, the length of any path of this type is definitely in excess of 16 *in*, and therefore the global optimum cannot be a path of this type. The remaining path type allows the spider to move upward 1^+ *in* to the top of the block, continue diagonally towards the right/left side, then diagonally across and downward towards the back face, and finally 2^+ *in* to the fly. See Figure 2. It is not immediately obvious that the local optimum of this path type cannot also be the global optimum. Hence, the spider must investigate further to determine the geodesic from his position to that of the fly's.

Design Space Set Up

To set up the design space for this problem, the spider adopts a cartesian coordinate system aligned with the wooden block such that the origin coincides with the front-lower-left corner of the block. The x coordinate measures positive to the right, the y coordinate measures positive along the long side of the block away from the front face, and the z coordinate measures positive upward in the vertical direction. In this coordinate system, the spider's initial position on the front face is $(XS, YS, ZS) = (2, 0, 3)$, and the fly is trapped on the back face at $(XF, YF, ZF) = (2, 12, 1)$.

The path type to optimize can be partitioned into four segments, corresponding to the four block faces to be traversed. The first segment is described by end points $(2, 0, 3)$ and $(X, 0, 4)$. The second segment's end points are $(X, 0, 4)$ and $(4, Y, 4)$. The third, $(4, Y, 4)$ and $(4, 12, Z)$. The fourth, $(4, 12, Z)$ and $(2, 12, 1)$. Hence, the complete path can be described as the piecewise linear curve that connects $(2, 0, 3)$, $(X, 0, 4)$, $(4, Y, 4)$, $(4, 12, Z)$, and $(2, 12, 1)$. In this design space, there are precisely three design variables (X, Y, Z). Further, the design space is constrained by the inequalities:

$$\begin{array}{rcccl} 0 & \leq & X & \leq & 4, \\ 0 & \leq & Y & \leq & 12, \\ 0 & \leq & Z & \leq & 4. \end{array} \tag{1}$$

Hence, the design space as defined in this problem is constrained to the interior of the wooden block. While these constraints facilitate limiting the search for the optimium, once we find the optimum path, it will become obvious that these are non-active constraints. So we will follow up this problem with an addendum problem which introduces another constraint on the path to illustrate how one can handle constraints that may or may not become active at the constrained minimum.

Cost Function

The length of each segment is given as:

$$\begin{array}{rcl} S_1 & = & \left[1 + (X-2)^2\right]^{\frac{1}{2}}, \\ S_2 & = & \left[(X-4)^2 + Y^2\right]^{\frac{1}{2}}, \\ S_3 & = & \left[(Y-12)^2 + (Z-4)^2\right]^{\frac{1}{2}}, \\ S_4 & = & \left[(Z-1)^2 + 4\right]^{\frac{1}{2}}. \end{array} \tag{2}$$

The total path length (or objective/cost function) is defined as:

$$I \equiv S = S_1 + S_2 + S_3 + S_4. \tag{3}$$

The statement of optimization is to minimize I subject to the constraints of Eqn (1). Note that Eqn (3) is *not* a quadratic equation. Furthermore, this cost function is not equivalent to summing the squares of the line segment lengths, which is a quadratic equation. So, always be careful when defining your cost function. I emphasize this point, as this question has been asked in the past and argued with conviction.

There are many ways in which one can proceed to solve this problem of optimization. For example, one approach is to use evolution theory, where a population of random guesses of $(X, Y, Z)_i$ is evaluated for their associated set of cost-function values, I_i. This establishes a generation of information which can be used to coerse subsequent generations towards the optimum location within the design space. An improvement to basic evolution theory is the Genetic Algorithm (GA). GAs attempt to speed the evolution process by combining the *genes* of promising pairs from one generation to procreate the next. GAs also allow some fraction of mutations to occur in order to improve the chance of finding a global optimum. However in general, this is *not* guaranteed. These methods are relatively easy to set up and program as they do not require any gradient information, and in fact may be the best choice if the cost function does not smoothly vary throughout the design space. Unfortunately, they can be computationally very expensive, even for a problem with a modest number of design variables. Nonetheless, solving the spider-fly problem with evolution methods can be entertaining, and we recommend it to the ambitious student as a follow-up exercise to this lecture.

Gradient-Based Optimization

Let's now consider gradient-based optimization techniques. These optimization methods use derivatives of the objective function with respect to the design space to navigate the design space from an initial state to a local optimum. These techniques include steepest descent, Newton methods, and quasi-Newton methods, amoung others.

In the case of the spider-fly problem, the exact derivatives are easily found. However, in general this is not usually the case for most large-scale problems of interest, so one may have to resort to utilizing an approximate derivative. We will study both, and introduce a couple of basic methods that one can use to approximate the gradient of a cost function.

Exact Gradient

Let's return to the definition of the cost function given by Eqns (2-3). In this simple example problem it is straightforward to derive the first and second derivatives of the cost function. Hence, the first variation of the cost function is:

$$\delta I = I_X \delta X + I_Y \delta Y + I_Z \delta Z \equiv G \, \delta \mathcal{X} \tag{4}$$

where G is the gradient vector, $\mathcal{X}$ is the design space vector, and the partial derivatives with respect to the design space are given as:

$$\begin{aligned} I_X &= \tfrac{(X-2)}{S_1} + \tfrac{(X-4)}{S_2}, \\ I_Y &= \tfrac{Y}{S_2} + \tfrac{(Y-12)}{S_3}, \\ I_Z &= \tfrac{(Z-4)}{S_3} + \tfrac{(Z-1)}{S_4}. \end{aligned} \tag{5}$$

Exact Hessian

Now find the Hessian matrix (second derivatives) for this problem; it will be needed to navigate the design space with a Newton iteration. The Hessian matrix is:

$$A = \begin{bmatrix} I_{XX} & I_{YX} & I_{ZX} \\ I_{XY} & I_{YY} & I_{ZY} \\ I_{XZ} & I_{YZ} & I_{ZZ} \end{bmatrix} \tag{6}$$

where,

$$\begin{aligned} I_{XX} &= \tfrac{1}{S_1^3} + \tfrac{Y^2}{S_2^3} \\ I_{XY} &= I_{YX} = \tfrac{(4-X)Y}{S_2^3} \\ I_{XZ} &= I_{ZX} = 0 \\ I_{YY} &= \tfrac{(X-4)^2}{S_2^3} + \tfrac{(Z-4)^2}{S_3^3} \\ I_{YZ} &= I_{ZY} = \tfrac{(Y-12)(4-Z)}{S_3^3} \\ I_{ZZ} &= \tfrac{(Y-12)^2}{S_3^3} + \tfrac{4}{S_4^3} \end{aligned} \tag{7}$$

Approximate Gradient by Finite Differences

One can approximate the gradient by finite differences, where the amplitude of each design variable is independently perturbed by a small delta from the current state and the cost function reevaluated. In general, this requires $N+1$ function evaluations to approximate the gradient, where N is the number of design variables. (N is also referred to as the number of degrees of freedom, or as the dimension of the design space.)

Consider the Taylor series expansion of a function f.

$$f(x+\Delta x) = f(x) + \Delta x\ f_x(x) + \frac{\Delta x^2}{2} f_{xx}(x) + \ldots + \frac{\Delta x^n}{n!} f_n(x) + \ldots \tag{8}$$

A first-order accurate approximation of $f_x(x)$ can be determined from Eqn (8) with a forward differencing.

$$f_x(x) \simeq \frac{f(x+\Delta x) - f(x)}{\Delta x}. \tag{9}$$

Using Eqn (9), let's approximate I_X of Eqn (5).

$$I_X \simeq \frac{I(X+h,Y,Z) - I(X,Y,Z)}{h} \tag{10}$$

Where $h = \Delta x$ is a small perturbation of the X coordinate. Using Eqn (10) with $h = 10^{-3}$ at $(X,Y,Z) = (2,6,2)$ gives $I_X \simeq -0.31565661$, an error of about 0.1%. The exact value of I_X at this location is $-\frac{2}{\sqrt{40}} \simeq -0.31622777$. In exact mathematics, the accuracy of I_X improves as $h \to 0$. However, if we evaluate the approximation of this derivative with various values of h using finite-precision mathematics, one will quickly find that as h gets really small, the stability of this evaluation will eventually degrade. For example at $(X,Y,Z) = (2,6,2)$ and using 64-bit precision, the finite-difference estimate of I_X improves with decreasing h until $h < 10^{-7}$, as shown below in Table 1. However at this point, the trend reverses and the error of I_X begins to grow with decreasing h. And remember, this is a simple cost function. A cost function for aerodynamic shape optimization is typically related to the computed drag of a design. Computing a high-precision value for drag can be quite costly, and in this context, finding an appropriate value of h for each design variable can become problematic. In practice, using a second-order accurate finite differencing of the function not only doubles the cost of the gradient, it typically does not solve the stability issue of finite-precision computations.

Approximate Gradient by Complex Variables

Another approach to approximating the gradient is now presented which, as it turns out, is much more stable than finite differences. In general, the computational cost of this technique also scales with $\mathcal{O}(N)$ function evaluations.

Let's revisit the Taylor series of Eqn (8). This expansion also holds true for complex functions of complex variables. With this in mind, let's now consider an imaginary perturbation of X, where $\Delta x = ih$. If we take

	$log_{10}(Error\ I_X)$	
$log_{10}(h)$	Finite Difference	Complex Variable
-1	-1.244	-4.449
-2	-2.243	-6.449
-3	-3.243	-8.449
-4	-4.243	-10.449
-5	-5.243	-12.449
-6	-6.244	-14.449
-7	-7.192	-16.256
-8	-6.778	-16.256
-9	-5.977	-16.256
-10	-4.768	-16.256

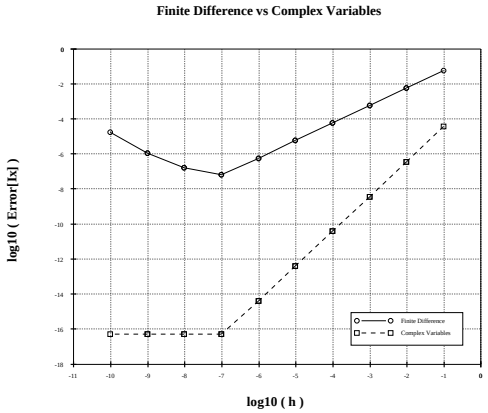


Table 1: Stability of Finite-Difference and Complex-Variable Methods.

the imaginary part of the expansion of $f(x+ih)$, we get a second-order accurate approximation of f_x, as given below.

$$f_x(x) \simeq \frac{Im[f(x+ih)]}{h}. \tag{11}$$

Now we can approximate I_X using Eqn (11).

$$I_X \simeq \frac{Im[I(X+ih,Y,Z)]}{h} \tag{12}$$

Eqn (12) for all $h \leq 10^{-3}$ at $(X,Y,Z) = (2,6,2)$ gives $I_X \simeq -0.31622777$, which is identical to the exact value of this derivative to 8 significant digits. Table 1 illustrates the improved accuracy and stability of using complex variables relative to using finite differences to approximate the gradient of a function. Note that while the finite-difference error reverses its trend at $h \leq 10^{-7}$, the complex-variable estimates level out.

Additional information on the complex-variable method for approximating gradients for the Navier-Stokes equations can be found in Anderson *et. al.* [2].

Steepest Descent

One can use the gradient to establish a steepest descent trajectory of the cost function through the design space. This trajectory begins with the initial state (baseline) and ends at a local minimum. From Eqn (4) it can be seen that a reduction in the cost function is realized if a sufficiently small step is taken in the negative gradient direction, and a local minimum is found when the magnitude of the gradient vanishes. Accordingly, we can iteratively update the value of the design variables in the following manner.

$$\mathcal{X}^{n+1} = \mathcal{X}^n + \delta\mathcal{X}^n, \tag{13}$$

where n is the iterative step number, and

$$\delta\mathcal{X}^n = -\lambda G. \tag{14}$$

Here, $\lambda > 0$ is a step-size parameter. Thus,

$$\delta I^n = G\ \delta\mathcal{X}^n = -\lambda G^2 \leq 0. \tag{15}$$

In order to initialize the trajectory search for the spider-fly problem, we arbitrarily select the center of the allowable design space as our baseline path. This corresponds to the following initial state for the design space vector, $\mathcal{X}^0$, the gradient vector, G^0, and the Hessian matrix, A^0. Note that for this arbitrarily chosen state, the objective function, I^0, is already less than that of the local-minimum path of Figure 1; this is

nothing more than a coincidence.

$$\mathcal{X}^0 = \begin{bmatrix} 2 \\ 6 \\ 2 \end{bmatrix}, \quad G^0 = \begin{bmatrix} \frac{-2}{\sqrt{40}} \\ 0.0 \\ (\frac{-3}{\sqrt{40}} + \frac{1}{\sqrt{5}}) \end{bmatrix} \approx \begin{bmatrix} -0.31623 \\ 0.0 \\ -0.02713 \end{bmatrix},$$
$$A^0 \approx \begin{bmatrix} 1.14230 & 0.04743 & 0.0 \\ 0.04743 & 0.03162 & -0.04743 \\ 0.0 & -0.04743 & 0.50007 \end{bmatrix}, \tag{16}$$

with

$$I^0 = (1 + 2\sqrt{40} + \sqrt{5}) \approx 15.88518.$$

In a steepest descent procedure, the value of the free parameter λ is usually limited by the stability of the iterative process. We will discuss this further in our second model problem. Through some experimentation, a value of $\lambda = 1.885$ yields about the fastest convergence to the optimum state for the spider-fly problem. Using 64-bit mathematical operations, the exact minimum-distance path is found within machine-level-zero in 295 iterations. The convergence of the gradient is shown in Figure 3. Here, the magnitude of the gradient has been reduced more than 7.5 orders of magnitude. Figure 4 provides the trajectory of the optimization process through the design space from start to finish. Notice the top view of this trajectory. Here one can see the high-frequency zig-zag nature of this navigation, but it appears that the general trend of the trajectory is correct. If the high-frequency behavior could be filtered and the low-frequency trend amplified, then convergence to the optimum could be accelerated. Such a technique will be addressed in our second model problem of this lecture.

Newton Iteration

Now let us solve this optimization problem with a Newton iteration. To develop a Newton iteration, simply replace the variation of the design variables of Eqn (14) with

$$\delta\mathcal{X}^n = -A^{-1}G = -HG, \tag{17}$$

where $H = A^{-1}$ is the inverse of the Hessian matrix of Eqns (6-7). Using this approach, the exact solution to machine-level-zero is found in just 3 steps. Figure 5 provides the convergence of the gradient, and as expected, this convergence has a quadratic behavior. Figure 6 illustrates the corresponding navigation through the design space. Data from the Newton iteration are also given in Table 2.

In this model problem, the superior performance of the Newton iteration over that of the steepest descent method may lead one to abandon steepest descent in favor of an approach that utilizes the Hessian. However, let's investigate the requirements of a Newton method in more depth. First, the construction of the Hessian matrix can cost $\mathcal{O}(N)$ times that of the gradient, unless the Hessian happens to be a sparse matrix. In the case of the spider-fly problem, this additional computational work trades very favorably (i.e., $3(1+3) << 295$). However, when N is moderately large, this may no longer be the case. A second issue is that the terms of the Hessian matrix may not be explicitly available, and in fact this is usually the case for aerodynamic shape optimization problems.

n	X^n	Y^n	Z^n	I^n
0	2.000000	6.000000	2.000000	15.88518
1	2.319023	4.984009	1.641696	15.81167
2	2.333268	4.999744	1.666556	15.81139
3	2.333333	5.000000	1.666667	15.81139

Table 2: Convergence of Newton Iteration on the Spider-Fly Problem.

Quasi-Newton Methods

To consider this scenario, let's assume that neither the Hessian matrix, A, or its inverse, H, are readily available for the spider-fly problem. Under this circumstance, a quasi-Newton method can be employed.

For the purpose of providing this case as a training exercise, we will document in detail the results of an application of a Rank-1 (R1) quasi-Newton iteration to the spider-fly problem. In this approach, the inverse of the Hessian matrix, H, is approximated and updated concurrently with the trajectory search. The Rank-1 updates of H are given by the following relationship.

$$H^{n+1} = H^n + \frac{(\mathcal{P}^n)(\mathcal{P}^n)^T}{(\mathcal{P}^n)^T \delta G^n}, \tag{18}$$

where

$$\delta G^n = G^{n+1} - G^n$$

and

$$\mathcal{P}^n = \delta \mathcal{X}^n - H^n \delta G^n.$$

Here, n is the iteration index, with $n = 0$ representing the initial state, and H^0 being initialized as the identity matrix. $\mathcal{P}$ is an error vector of the Rank-1 approximation of the Hessian. In the case of the spider-fly problem, H^n is a 3x3 matrix, and $\delta\mathcal{X}^n$, δG^n & $\mathcal{P}^n$ are 3-dimensional vectors. Note that $[(\mathcal{P}^n)(\mathcal{P}^n)^T]$ is a 3x3 matrix, and $[(\mathcal{P}^n)^T \delta G^n]$ is a scalar inner product. The numerical results of the R1 trajectory are illustrated in Figures 7-8, and tabulated in Table 3. Although it takes 6 to 7 iterations before the Hessian inverse is sufficiently approximated, the R1 iterations eventually exhibit quadratic convergence. Close inspection of Table 3 reveals some trends worth noting. Recall that vector $\mathcal{P}^n$ is a measure of the error of the Hessian inverse H^n. These data show that $\mathcal{P}$ exhibits quadratic convergence after 5 iterations. The gradient vector G also exhibits quadratic convergence after 6 iterations. However, the final Hessian inverse H^{10}, although close to the exact matrix H^∞, still contains terms which are off in the fourth decimal place.

Nash Equilibrium

For completeness, we will review one more popular technique - the Nash equilibrium. In this approach, between iterations, independent sub-searches are performed in each of the N coordinate directions of the design space. For example, in the case of the spider-fly problem, a sub-optimization is performed in the X direction, to find $X^\star$, while holding Y^n & Z^n fixed. Similarily, sub-optimizations are also done on Y & Z, to determine $Y^\star$ & $Z^\star$, respectively. A significant attribute of this approach is that each of the N sub-optimizations are independent of the others, and as such, the whole set of sub-optimizaitons can be performed in parallel. For the spider-fly, these sub-optimizations are defined as:

$$\begin{array}{lcll} minimize\ I(X^\star, Y^n, Z^n) & \rightarrow & I_x(X^\star, Y^n, Z^n) = 0 & \rightarrow \quad X^\star, \\ minimize\ I(X^n, Y^\star, Z^n) & \rightarrow & I_x(X^n, Y^\star, Z^n) = 0 & \rightarrow \quad Y^\star, \\ minimize\ I(X^n, Y^n, Z^\star) & \rightarrow & I_x(X^n, Y^n, Z^\star) = 0 & \rightarrow \quad Z^\star. \end{array}$$

By manipulating Eqn 5, these minimizations can be reduced to:

$$X^\star = \frac{2(2+Y^n)}{(1+Y^n)}, \qquad Y^\star = \frac{12(4-X^n)}{(8-X^n-Z^n)}, \qquad Z^\star = 4 - \frac{3(12-Y^n)}{(14-Y^n)}. \tag{19}$$

Once the sub-optimizations are completed, the design vector is updated as:

$$\begin{array}{rcl} X^{n+1} & = & X^\star, \\ Y^{n+1} & = & Y^\star, \\ Z^{n+1} & = & Z^\star. \end{array}$$

This process is repeated until the desired level of convergence has been achieved. Note that each of the intermediate cost functions will be less than or equal to I^n, the cost function at step n. However, this does not imply that the magnitude of the intermediate gradients are likewise bounded by the magnitude of G^n. Furthermore, it is not guaranteed that $I^{n+1} \leq I^n$, or $|G^{n+1}| \leq |G^n|$. Nonetheless, when a Nash equilibrium is located, it coincides with a local optimum in the design space. Figures 9-10 illustrate the convergence of the Nash approach as applied to the spider-fly problem. The *Error* depicted in Figure 9 is defined as: $Error = I^n - I_{min}$. While the convergence of the objective function I^n exhibits a monotonic behavior, the convergence of the gradient G^n does not. Select data are also provided in Table 4 for reference.

n	X^n	Y^n	Z^n	I^n
0	2.000000	6.000000	2.000000	15.88518
1	2.316228	6.000000	1.869014	15.82842
2	2.309340	5.995497	1.854977	15.82729
3	2.283594	5.931327	1.731183	15.82250
4	2.268113	6.064459	1.736156	15.82602
5	2.329076	5.002280	1.654099	15.81144
6	2.325976	4.997523	1.643056	15.81157
7	2.333299	4.999719	1.666628	15.81139
8	2.333331	5.000017	1.666668	15.81139
9	2.333333	5.000002	1.666667	15.81139
10	2.333333	5.000000	1.666667	15.81139

n	G^n			$\mathcal{P}^n$		
0	-0.3162278	0.0000000	0.1309858	-0.0313201	-0.0204757	-0.0638312
1	0.0313201	0.0204757	0.0638312	-0.0027101	-0.0067548	-0.0130308
2	0.0241196	0.0207513	0.0566640	0.0032314	-0.0277891	-0.0010379
3	-0.0051403	0.0239077	-0.0068202	-0.0092369	0.1609362	0.0124328
4	-0.0156349	0.0272111	-0.0109428	0.0038082	0.0058428	0.0135643
5	-0.0042385	0.0003440	-0.0070051	-0.0061541	-0.0018454	-0.0198081
6	-0.0076998	0.0004624	-0.0129071	0.0000316	0.0002953	0.0000405
7	-0.0000509	-0.0000095	-0.0000100	0.0000021	-0.0000171	-0.0000017
8	-0.0000014	0.0000003	0.0000003	0.0000003	-0.0000015	-0.0000002
9	-0.0000003	0.0000000	0.0000001	0.0000000	0.0000000	0.0000000
10	0.0000000	0.0000000	0.0000000			

n	H^n			n	H^n		
	1.0000000	0.0000000	0.0000000		1.0178515	-2.0173360	-0.1120774
0	0.0000000	1.0000000	0.0000000	6	-2.0173360	39.0361115	2.7720896
	0.0000000	0.0000000	1.0000000		-0.1120774	2.7720896	1.9924568
	0.8602224	-0.0913802	-0.2848703		1.0194495	-2.0023937	-0.1100296
1	-0.0913802	0.9402598	-0.1862351	7	-2.0023937	39.1758307	2.7912373
	-0.2848703	-0.1862351	0.4194274		-0.1100296	2.7912373	1.9950809
	0.9263627	0.0734691	0.0331463		0.9628110	-1.5479228	-0.0640429
2	0.0734691	1.3511333	0.6063951	8	-1.5479228	35.5291298	2.4222374
	0.0331463	0.6063951	1.9485177		-0.0640429	2.4222374	1.9577428
	0.8366376	0.8450854	0.0619643		1.0930870	-2.1085463	-0.1558491
3	0.8450854	-5.2845988	0.3585666	9	-2.1085463	37.9416902	2.8173117
	0.0619643	0.3585666	1.9392619		-0.1558491	2.8173117	2.0224391
	0.9844233	-1.7298270	-0.1369557		1.0931086	-2.1081974	-0.1562477
4	-1.7298270	39.5788215	3.8244048	10	-2.1081974	37.9473320	2.8108673
	-0.1369557	3.8244048	2.2070087		-0.1562477	2.8108673	2.0298003
	0.7433885	-2.0996388	-0.9954916		1.0931330	-2.1081851	-0.1561619
5	-2.0996388	39.0114314	2.5071815	∞	-2.1081851	37.9473319	2.8109135
	-0.9954916	2.5071815	-0.8509886		-0.1561619	2.8109135	2.0301042

Table 3: Convergence of Rank-1 quasi-Newton Iteration on the Spider-Fly Problem.

n	X^n	Y^n	Z^n	I^n
0	2.000000	6.000000	2.000000	15.88518
1	2.285714	6.000000	1.750000	15.82411
2	2.285714	5.189189	1.750000	15.81388
3	2.323144	5.189189	1.680982	15.81186
4	2.323144	5.035762	1.680982	15.81148
5	2.331358	5.035762	1.669326	15.81141
6	2.331358	5.006782	1.669326	15.81139
7	2.332957	5.006782	1.667169	15.81139
8	2.332957	5.001287	1.667169	15.81139
9	2.333262	5.001287	1.666762	15.81139
10	2.333262	5.000244	1.666762	15.81139
11	2.333320	5.000244	1.666685	15.81139
12	2.333320	5.000046	1.666685	15.81139
13	2.333331	5.000046	1.666670	15.81139
14	2.333331	5.000009	1.666670	15.81139
15	2.333333	5.000009	1.666667	15.81139
16	2.333333	5.000002	1.666667	15.81139
17	2.333333	5.000002	1.666667	15.81139
18	2.333333	5.000000	1.666667	15.81139
19	2.333333	5.000000	1.666667	15.81139

Table 4: Convergence of Nash Equilibrium on the Spider-Fly Problem.

Design Space

In the case of the spider-fly problem, the set up is straight forward. In fact, it is somewhat difficult to envision how to set it up otherwise. Our definition of the design space is directly linked to the intersection of the spider's path with the three edges of the wooden block. All possible straight-line-segment paths that cross these three edges are represented in the constrained design space. The human mind is an amazing thing; it routinely filters information that *does not* apply to a given situation. Unfortunately, at times it hides some pertinent data. In the case of the spider-fly problem, this may indeed occur. To explain what we mean by this, we define the spider-fly problem from a different perspective.

The spider-fly problem falls within the class of problems which seek the geodesic between two points on an arbitrary surface. The geodesic is the minimum-length path that connects the two points and is confined to the surface. For example, the geodesic between two cities on Earth is given as the great-circle arc (assumes the Earth is a perfect sphere). Now replace the wooden-block with that of a super-ellipsoid surface given by:

$$\left[\frac{|x-2|}{2}\right]^p + \left[\frac{|y-6|}{6}\right]^p + \left[\frac{|z-2|}{2}\right]^p = 1 \tag{20}$$

where $p \geq 2$ is an arbitrary power. The spider's initial position is:

$$\begin{aligned} XS &= 2 \\ YS &= 6\left[1 - \left[1 - \frac{1}{2^p}\right]^{\frac{1}{p}}\right] \\ ZS &= 3 \end{aligned} \tag{21}$$

and the location of the trapped fly is:

$$\begin{aligned} XF &= 2 \\ YF &= 12 - YS \\ ZF &= 1 \end{aligned} \tag{22}$$

In the limit as $p \to \infty$, the super-ellipsoid surface approaches that of the original wooden block.

Yet one would probably set up the corresponding optimization problem very differently than as we did before. Most likely the geodesic would be approximated by a discrete path of $N+1$ piecewise-linear segments. This can be done in any number of ways. For example, one can seek a uniform segment-length path, or possibly a path with constant Δy segments. If the constant Δy approach is chosen, then the discrete design space can be defined by N angular positions at each of the interior y stations.

Let's make some observations regarding this set up of the geodesic problem. First, the evaluation of the discrete cost function (total length of the discrete path) will not be equal to that of the continuum geodesic. As a consequence, the nodes of the optimum discrete path will not fall on the continuum geodesic. The accuracy of the discrete problem can be improved by increasing the dimension of the design space, but the computational costs will escalate as well. In this approximation of the spider-fly problem, the value of the discrete cost function will be less than the length of the continuum geodesic. This is due to the fact that each discrete line segment takes a *short cut* from end-point to end-point as compared with the curved continuum geodesic. Furthermore, for the constant Δy representation, the error will degrade to first-order accurate on Δy in the limit as $p \rightarrow \infty$. Given these observations, one should wonder if there may be a better way to seek the discrete geodesic besides driving the gradient of the discrete cost function to zero. One approach that addresses this will be introduced later, when we get to our next model problem.

Exact Solution

Before we finally leave the spider-fly problem, has the reader determined how to directly compute the exact length of the geodesic on the wooden block? The solution to this problem can be found quite simply if one thinks of the wooden block instead as a cardboard box which can be unfolded and flattened out as illustrated by Figures 11-12. Here, $I_{min} = \sqrt{250} \approx 15.81139\ in.$ The corresponding optimum position in our design space is precisely $(X, Y, Z)_{opt} = (\frac{7}{3}, 5, \frac{5}{3})$.

This concludes our discussion on the spider-fly model problem. We hope this exercise has peaked the reader's interest, as well as challenged him/her to think beyond the norm. Our second model problem is based on the classic brachistochrone. Here, we will investigate additional quasi-Newton methods, but again, they all require $\mathcal{O}(N)$ iterations to establish the Hessian. Hence, for the class of optimization problems where N is very large, we conclude that a Newton iteration may not be the best choice. Instead, we will revisit steepest descent as the foundation of our optimization process, and develop techniques that accelerate the convergence of the steepest-descent trajectory.

Active Constraints and KKT Conditions

In the original spider-fly problem, as described above, the constraints of Eqn (1) were all inactive for the optimum path. In practice however, optimization problems of interest typically include active constraints. Therefore, we will revisit the spider-fly brain teaser, but this time with an active constraint added to the problem statement.

The spider notices that in the "unconstrained" problem above, the optimum path requires that he walks a distance of ≈ 5.270463 inches over the top-side of the wooden block. The length of this path segment has been designated S_2 in Eqn (2). Unfortunately, it is 12:00 noon and the Sun is beating down hard on the top of the block. The spider happens to know that this surface is too hot for him to comfortably traverse any more than 4 inches over it, so he wants to include a constraint that $S_2 \leq 4$ inches. First, we put this constraint into standard form per Boyd and Vandenberghe [3], and define the constraint equation and its gradient as:

$$\begin{aligned} C(X,Y) &= S_2(X,Y) - 4 \leq 0, \\ C_X &= (X-4)/S_2, \\ C_Y &= Y/S_2. \end{aligned} \tag{23}$$

Introduce a Lagrange dual cost function,

$$\mathcal{L}(X,Y,Z) \equiv S(X,Y,Z) + \mu C(X,Y), \tag{24}$$

and its gradient,

$$\nabla\mathcal{L}(X,Y,Z) = \nabla S(X,Y,Z) + \mu \nabla C(X,Y). \tag{25}$$

Here $\mu \geq 0$ is a Lagrange multiplier, the value of which will be determined during the minimization of Eqn (24) while satisfying the constraint of Eqn (23). More precisely, we satisfy the Karush-Kuhn-Tucker (KKT) conditions of optimality, which for this problem are:

$$\begin{aligned} C &\leq 0, \\ \mu &\geq 0, \\ \mu C &= 0, \\ |\nabla \mathcal{L}| &= 0. \end{aligned} \tag{26}$$

One can adapt the steepest descent method of Eqns (13-14) with an intermediate step which first projects the design vector into the allowable design space, then updates the value of μ. If $C \leq 0$ and $\mu = 0$, then proceed as usual. However, if $C > 0$, we project (X,Y) in the negative ∇C direction to the location where $C = 0$. Else, if $\mu C \neq 0$, then project in the positive ∇C direction to the location where $C = 0$. After either projection, an update to μ is made as follows.

$$\mu = max \left[-\frac{\nabla S \cdot \nabla C}{|\nabla C|} \, , \, 0 \right]. \tag{27}$$

With a non-zero value of μ, the constraint is active and $\nabla \mathcal{L}$ will be locally tangent to the constraint surface, therefore, the update on the design vector will remain inside or close to the allowable space. This process is repeated until all of the KKT conditions are satisfied within a user-specifed level of tolerance. For this optimum design vector, $\nabla \mathcal{L} = 0$ implies $\nabla S = -\mu \nabla C$, which in turn implies that the iso-surface of S is tangent to the iso-surface of constraint C at $(X, Y, Z)_{opt}$. Here, the optimum design state with $S_2 \leq 4$ is:

$$\begin{aligned} (X, Y, Z)_{opt} &\approx (2.443682 \, , \, 3.684817 \, , \, 1.581667), \\ \mu &\approx 0.04234745, \\ S_{opt} &\approx 15.83659. \end{aligned} \tag{28}$$

Figure 13 illustrates the convergence of $|\nabla \mathcal{L}|$ for the steepest descent process described above. Note that at almost every iteration, there are two symbols shown. The first symbol represents the projection of the design vector into the allowable space, while the second illustrates the steepest-descent step. Also notice that in this particular case, adding the active constraint actually improved convergence of the optimization. This behaviour is not necessarily typical. Figure 14 provides the trajectory through the design space for this constrained optimization. Since the trajectory bounces about, for clarity, the final converged solution is depicted with a hollow blue dot in a larger blue circle. Figures 15-16 compare the constrained path (blue line) with the previously computed geodesic (red line). The corresponding (X,Y,Z) design vectors are illustrated with blue or red dots, for the constrained or the unconstrained problems, respectively. Also included in these Figures are the cylindrical constraint boundary depicted as green curves, and the (X,Y,Z) projections of the constrained problem shown with gray lines. It may appear that the blue line in Figure 16 is comprised of two straight line segments, one from the spider to the edge between the top and side faces, and one from there to the fly. However, this is not the case. There is, in fact, an imperceptible kink at the edge between the front and top faces. Whereas the line from the top-side edge to the fly is indeed a straight line.

This concludes our discussion about the spider-fly problem, and we move on to another more challenging model problem based on the classical brachistochrone.

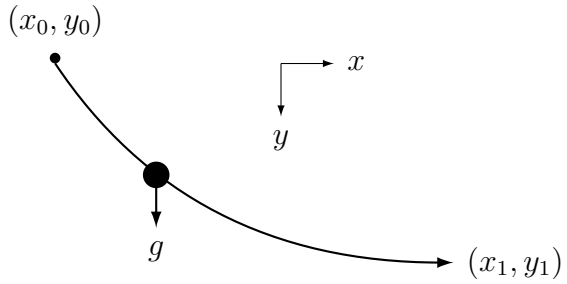
3 Brachistochrone Problem

The second model problem that we will study in this lecture is based on the classic brachistochrone.

Much of the fundamental theory to the branch of mathematics known as the calculus of variations is attributable to the Bernoulli brothers, Johann and Jakob, who were friendly rivals. They would design new mathematical problems to stimulate each other in the form of challenges. One of which is known as the *brachistochrone*, which dates back to June 1696 when Johann Bernoulli [4] proposed it as a challenge to the mathematical community. Only 5 solutions were provided initially; these came from Sir Isaac Newton, Gottfried Leibniz, Guillaume de L'Hopital, Johann and Jakob Bernoulli. [Actually, Galileo Galilei [5] studied this problem first in 1638, but incorrectly deduced that the optimum path is a circular arc.]

The authors' original work on this problem is documented in References [6]-[7].

The brachistochrone problem [8] is the determination of the path $y(x)$ connecting initial and final points (x_0, y_0) and (x_1, y_1) such that the time taken by a particle traversing this path, subject only to the force of gravity, is a minimum.



The total time is given by

$$T = \int_{x_0}^{x_1} \frac{ds}{v},$$

where the velocity of a particle falling under the influence of gravity, g, and starting from rest at $y = 0$, is

$$v = \sqrt{2gy}.$$

Denoting $\frac{dy}{dx}$ by y', and setting $ds = \sqrt{(1 + y'^2)}dx$ one finds that $T = \frac{I}{\sqrt{2g}}$ where

$$I = \int_{x_0}^{x_1} F(y, y')dx$$

with

$$F(y, y') = \sqrt{\frac{1 + y'^2}{y}}.$$

Then

$$\begin{aligned} \mathcal{G} &= \frac{\partial F}{\partial y} - \frac{d}{dx}\frac{\partial F}{\partial y'} \qquad (29) \\ &= -\frac{\sqrt{1 + y'^2}}{2y^{\frac{3}{2}}} - \frac{d}{dx}\frac{y'}{\sqrt{y(1 + y'^2)}} \end{aligned}$$

which may be simplified to

$$\mathcal{G} = -\frac{1 + y'^2 + 2yy''}{2(y(1 + y'^2))^{\frac{3}{2}}}. \qquad (30)$$

[Note that Eqn (29) is the Euler-Lagrange equation when set to zero.] In this case, since F is not a funciton of x,

$$\begin{aligned} &\left(y'\frac{\partial F}{\partial y'} - F\right)' \\ &= y''\frac{\partial F}{\partial y'} + y'\frac{d}{dx}\frac{\partial F}{\partial y'} - \frac{\partial F}{\partial y'}y'' - \frac{\partial F}{\partial y}y' \\ &= y'\left(\frac{d}{dx}\frac{\partial F}{\partial y'} - \frac{\partial F}{\partial y}\right) \\ &= -y'\mathcal{G}. \end{aligned}$$

On the optimal path $\mathcal{G} = 0$ and hence,

$$\left(y'\frac{\partial F}{\partial y'} - F\right) \textit{ is constant.}$$

Here

$$\left(y'\frac{\partial F}{\partial y'} - F\right) = \frac{-1}{\sqrt{y(1+y'^2)}}.$$

Hence it follows that

$$\sqrt{y(1+y'^2)} = C,$$

where C is a constant. The classical solution is obtained by the substitution

$$\begin{aligned} y(t) &= \frac{1}{2}C^2(1-cos(t)) \\ &= C^2 sin^2\left(\frac{t}{2}\right). \end{aligned} \tag{31}$$

Then

$$y'^2 = \frac{C^2}{y} - 1,$$

$$\begin{aligned} y' &= \sqrt{\frac{C^2-y}{y}} \\ &= cot\left(\frac{t}{2}\right), \end{aligned}$$

and

$$\begin{aligned} x &= \int \frac{dy}{y'} \\ &= \int tan\left(\frac{t}{2}\right)\frac{dy}{dt}dt, \end{aligned}$$

where

$$\frac{dy}{dt} = C^2 sin\left(\frac{t}{2}\right) cos\left(\frac{t}{2}\right).$$

Thus, if one choses $x_0 = 0$,

$$\begin{aligned} x(t) &= C^2 \int sin^2\left(\frac{t}{2}\right) dt \\ &= \frac{1}{2}C^2 \int (1-cos(t))dt \\ &= \frac{1}{2}C^2(t-sin(t)). \end{aligned} \tag{32}$$

The optimal path described parametrically in t by Eqns (31) & (32) is a cycloid. However, when a numerical method is to be adopted, a discussion regarding the discrete problem is in order. Nonetheless, having the exact analytical solution of the brachistochrone provides significant value when comparing various aspects of different numerical techniques. The next sections discuss numerical procedures for solving the brachistochrone problem. Section 4 discusses the approximation of the gradient, while Section 5 discusses alternative descent procedures.

4 Gradient Calculations

This section describes two approaches to approximating the gradient of the objective function, I. The first is based on deriving the gradient of the continuous problem, then approximating this continuous gradient through numerical discretization. We will refer to this as the continuous gradient. The second scheme determines the exact gradient of the discrete cost function. We will refer to this as the discrete gradient. In either case, the resulting gradient calculations are only approximations of the exact gradient of the objective function of the continuous problem. A discussion at the end of this section addresses the benefits and deficiencies of each approach.

4.1 Continuous Gradient

Numerical optimization methods use the gradient $\mathcal{G}$ as the basis of a search method. In developing the gradient for the brachistochrone problem, let the trajectory be represented by the discrete values

$$y_j = y(x_j),$$

at

$$x_j = j\Delta x,$$

where Δx is the mesh interval, $0 \leq j \leq N+1$ and N is the number of design variables. Note that y_0 and y_{N+1} are fixed end-points of the path. The gradient is evaluated by applying a second-order finite difference approximation to the continuous gradient, Eqn (30), by substituting

$$y'_j = \frac{y_{j+1} - y_{j-1}}{2\Delta x},$$

$$y''_j = \frac{y_{j+1} - 2y_j + y_{j-1}}{\Delta x^2},$$

for y' and y'' at the j^{th} point. This yields the following approximation of the continuous gradient at the discrete points j.

$$\mathcal{G}_j = -\frac{1 + y'^2_j + 2y_j y''_j}{2(y_j(1+y'^2_j))^{\frac{3}{2}}}. \tag{33}$$

4.2 Discrete Gradient

In the second approach, the discrete cost function is calculated by approximating I by the rectangle rule of integration.

$$I_R = \sum_{j=0}^{N} F_{j+\frac{1}{2}} \Delta x,$$

where

$$F_{j+\frac{1}{2}} = \sqrt{\frac{1 + y'^2_{j+\frac{1}{2}}}{y_{j+\frac{1}{2}}}},$$

with

$$y_{j+\frac{1}{2}} = \frac{1}{2}(y_{j+1} + y_j),$$

$$y'_{j+\frac{1}{2}} = \frac{(y_{j+1} - y_j)}{\Delta x}.$$

The rectangle rule of integration gives second-order accuracy for a smooth integrand. In this case, its use allows the evaluation of I even when $y_0 = 0$ at the left boundary, where $v = 0$ and F becomes unbounded.

The discrete gradient is now evaluated by directly differentiating I_R,

$$\mathcal{G}_j = \frac{\partial I_R}{\partial y_j}.$$

It may be verified that

$$\mathcal{G}_j = \mathcal{B}_{j-\frac{1}{2}} - \mathcal{B}_{j+\frac{1}{2}} - \frac{\Delta x}{2}(\mathcal{A}_{j+\frac{1}{2}} + \mathcal{A}_{j-\frac{1}{2}}), \tag{34}$$

where $\mathcal{A}$ and $\mathcal{B}$ are calculated by evaluating

$$\mathcal{A}_{j+\frac{1}{2}} = \frac{\sqrt{1+y'^2}}{2y^{\frac{3}{2}}},$$

$$\mathcal{B}_{j+\frac{1}{2}} = \frac{y'}{\sqrt{y(1+y'^2)}},$$

with the values $y_{j+\frac{1}{2}}$ and $y'_{j+\frac{1}{2}}$.

4.3 Continuous vs Discrete Gradient

Consider for a moment, that an optimization of the continuous problem is the actual goal. In general, the optimum state of the discrete problem will not coincide with that of the continuous system, but of course, one hopes that it will be "close."

An advantage of the discrete gradient is that it directly relates to the objective function as it is evaluated numerically. Consequently, if a local optimum is found by driving the discrete gradient to zero, it may be easily verified that this is indeed a local optimum by making small perturbations and re-evaluating the objective function. If the search procedure involves line searches to find the minimum along a given search direction, the searches will also be compatible with the calculated gradient.

When the continuous gradient is driven to zero, local perturbations of the discrete cost function do not necessarily verify that a local minimum has been obtained. However, we offer a conjecture that an optimization based on the continuous gradient may be closer to the optimum of the continuous problem than that based on the discrete gradient. Our reasoning is based on the following argument. Consider a smooth curve defined by $I = f(y)$, where $f(y)$ is a known function. Now assume that $f(y)$ cannot be evaluated directly, only measured inexactly. Through these measurements, $f(y)$ is approximated by $\hat{f}(y)$, where

$$\hat{f}(y) = f(y) + \mathcal{E}_f$$

and $\mathcal{E}_f$ is the error associated with discretization of $f(y)$. This error can become further amplified if a derivative of $f(y)$ is sought, especially if $\mathcal{E}_f$ is a high-frequency error. Conversely, if the quantity $g(y) = f'(y)$ can be measured, we have

$$\hat{g}(y) = g(y) + \mathcal{E}_g$$

where $\mathcal{E}_g$ is the error associated with the discretization of $g(y)$.

Now if the inequality

$$||\mathcal{E}_g|| < ||\frac{d}{dy}\mathcal{E}_f||$$

holds true, then $\hat{g}(y)$ more accurately represents the exact gradient than $\hat{f}'(y)$ does, and it should follow that an optimization based on $\hat{g}(y)$ will more accurately recover the true optimum than one driven by $\hat{f}'(y)$.

In the next section, several search methods are discussed which have been under investigation in this effort.

5 Search Methods

A variety of search methods have been evaluated, including steepest descent, modified steepest descent with smoothing, implicit descent, multigrid steepest descent, Krylov acceleration, and quasi-Newton methods. These are outlined below. In all cases, the values y_j are regarded as the design variables.

5.1 Steepest Descent

Here a simple step is taken in the negative gradient direction. Such a strategy is reviewed further in Reference [9]. Denoting the iterations with the superscript n, we have

$$y_j^{n+1} = y_j^n - \lambda \mathcal{G}_j^n.$$

This may be regarded as a forward Euler discretization of a time dependent process with $\lambda = \Delta t$. Hence,

$$\frac{\partial y}{\partial t} = -\mathcal{G}.$$

Substituting for $\mathcal{G}$ from Eqn (30) it can be seen that y is the solution of the nonlinear parabolic equation

$$\frac{\partial y}{\partial t} = \frac{1 + y'^2 + 2yy''}{2(y(1+y'^2))^{\frac{3}{2}}}. \tag{35}$$

Thus it is possible to estimate the time step limit for stable integration. This is dominated by the parabolic term $\beta y''$ where

$$\beta = \frac{y}{(y(1+y'^2))^{\frac{3}{2}}}.$$

This gives the following estimate on the time step limit for a stable forward Euler explicit scheme.

$$\Delta t^\star = \frac{\Delta x^2}{2\beta}.$$

Thus the number of iterations can be expected to grow as the square of the number of design variables. Hence, this approach is prohibitively expensive to apply to large scale engineering problems of interest.

5.2 Smoothed (Sobolev) Descent

In trajectory optimization problems such as the brachistochrone problem, one may anticipate that the optimum solution will be a smooth curve. This suggests that neighboring points defining the trajectory should not be perturbed independently during the optimization process, but rather be moved so that the modified trajectory remains smooth. In the case of aerodynamic design, the described aerodynamic shapes will generally be smooth (except at occasional corners such as the trailing edge of the wing or at component intersections). This motivates the introduction of smoothing directly into the optimization process.

Here, the gradient $\mathcal{G}$ is replaced by a smoothed gradient $\bar{\mathcal{G}}$ defined by the implicit smoothing equation.

$$\bar{\mathcal{G}} - \frac{\partial}{\partial x}\epsilon\frac{\partial \bar{\mathcal{G}}}{\partial x} = \mathcal{G}. \tag{36}$$

Now one sets

$$\delta y = -\lambda \bar{\mathcal{G}} \tag{37}$$

where

$$\lambda = \Delta t \equiv STEP\ \Delta t^\star$$

and $STEP$ is a CFL-like input parameter.

Then to first order the variation in I is

$$\delta I = \int_{x_0}^{x_1} \mathcal{G}\delta y dx = -\lambda \int_{x_0}^{x_1} \left(\bar{\mathcal{G}} - \frac{\partial}{\partial x}\epsilon\frac{\partial \bar{\mathcal{G}}}{\partial x}\right)\bar{\mathcal{G}} dx.$$

Then, integrating by parts and using the fact that the end points are fixed,

$$\delta I = -\lambda \int_{x_0}^{x_1} \left(\bar{\mathcal{G}}^2 + \epsilon\left(\frac{\partial \bar{\mathcal{G}}}{\partial x}\right)^2\right) dx.$$

Thus descent is assured with an arbitrarily large value of the smoothing parameter ϵ.

The smoothing acts as a preconditioner. In practice it is noted that λ ($or \Delta t$) can be doubled when ϵ is doubled over a wide range, and it is possible to increase Δt to very large values. It should also be noted that if the optimimum shape turns out not to be a smooth shape, this preconditioner still allows the optimum to be recovered. However, the preconditioner forces the trajectory to approach the non-smooth optimum shape from a set of increasingly-conforming smooth shapes. Next, it is shown that the implicit smoothing operator is equivalent to casting the gradient in a Sobolev space.

Define a modified Sobolev inner product

$$\langle u, v\rangle = \int_\Omega (uv + \epsilon \nabla u \cdot \nabla v) d\Omega\ ,$$

then

$$\langle u, v\rangle = (u, v) + (\epsilon\nabla u, \nabla v)$$

where the (u, v) is the standard inner product in L_2. Integration by parts yields

$$\langle u, v\rangle = (u - \nabla(\epsilon\nabla u), v) + \int_{\partial\Omega} \epsilon v \frac{\partial u}{\partial n} d\partial\Omega.$$

Using the inner product notation the variation of the cost function I can be expressed as

$$\delta I = (\mathcal{G}, \delta y) = \langle \bar{\mathcal{G}}, \delta y\rangle = \left(\bar{\mathcal{G}} - \nabla\left(\epsilon\nabla\bar{\mathcal{G}}\right), \delta y\right)\ .$$

Therefore we can solve implicitly for $\bar{\mathcal{G}}$

$$\bar{\mathcal{G}} - \nabla\left(\epsilon\nabla\bar{\mathcal{G}}\right) = \mathcal{G}.$$

Then, if one sets $\delta y = -\lambda\bar{\mathcal{G}}$,

$$\delta I = -\lambda\langle\bar{\mathcal{G}},\bar{\mathcal{G}}\rangle = -\lambda\left(\mathcal{G},\bar{\mathcal{G}}\right) \ ,$$

and an improvement is assured if λ is sufficiently small and positive, unless the process has already reached a stationary point at which $\bar{\mathcal{G}} = 0$ (and therefore $\mathcal{G} = 0$).

5.3 Implicit Descent

As it turns out, if the gradient is dominated by a y'' term, as is the case with the brachistochrone, see Eqn (35), the smoothed descent given by Eqn (37) can be shown to be equivalent to an implicit stepping scheme if the appropriate choice of ϵ is used.

Consider the parabolic equation

$$\frac{\partial y}{\partial t} = \beta\frac{\partial^2 y}{\partial x^2}$$

where β is variable. Solving this with an implicit scheme, the resulting system is

$$-\alpha\delta y_{j-1} + (1+2\alpha)\delta y_j - \alpha\delta y_{j+1} = -\Delta t\hat{\mathcal{G}}_j \tag{38}$$

where δy_j is the correction to y_j,

$$\alpha = \frac{\beta\Delta t}{\Delta x^2} = \frac{\Delta t}{2\Delta t^\star} = \frac{1}{2}\ STEP \tag{39}$$

and

$$\hat{\mathcal{G}}_j = \frac{\beta}{\Delta x^2}(y^n_{j-1} - 2y^n_j + y^n_{j+1}).$$

Combining Eqns (36 & 37), the discrete smoothed descent method assumes the form of Eqn (38) with

$$\alpha = \frac{\epsilon}{\Delta x^2}. \tag{40}$$

Comparing Eqn (39) with Eqn (40), one can see using the smoothed gradient is equivalent to an implicit time stepping scheme if $\epsilon = \frac{1}{2}STEP\ \Delta x^2$. Furthermore, a Newton iteration is recovered as $\Delta t \to \infty$.

While it is fortunate that the brachistochrone problem can be solved by an implicit time stepping scheme, in general, this may not be the case with the more complicated optimization of aerodynamic design. However, in this paper, we will use the performance of the implicit scheme to establish a goal for the multigrid descent methods described below.

5.4 Multigrid Descent

Radical improvements in the rate of convergence to a steady state can be realized by the multigrid time-stepping technique. The concept of acceleration by the introduction of multiple grids was first proposed by Fedorenko [10]. There is now a fairly well-developed theory of multigrid methods for elliptic equations based on the concept that the updating scheme acts as a smoothing operator on each grid [11, 12]. In the case of a time-dependent problem, one may expect that it should be possible to accelerate the evolution of a hyperbolic system to steady state by using large time steps on coarse grids so that disturbances will be more rapidly expelled through the outer boundary. Various multigrid time-stepping schemes have been designed to take advantage of this effect [13, 14, 15, 16, 17]. The present work adapts the scheme devised by the first author to solve the Euler equations [15] to the equation $\frac{dy}{dt} = -\mathcal{G}$.

A sequence of K meshes is generated by eliminating alternate points along each coordinate direction of mesh-level k to produce mesh-level $k+1$. Note that $k = 1$ refers to the finest mesh of the sequence. In order to give a precise description of the multigrid scheme, subscripts may be used to indicate grid level. Several transfer operations need to be defined. First, the solution vector, y, on grid k must be initialized as

$$y^{(0)}_k = T_{k,k-1}\ y_{k-1} \qquad , \qquad 2 \le k \le K$$

where y_{k-1} is the current value of the solution on grid $k-1$, and $T_{k,k-1}$ is a transfer operator. Next it is necessary to transfer a residual forcing function, P, such that the solution on grid k is driven by the residuals of grid $k-1$. This can be accomplished by setting

$$P_k = Q_{k,k-1}\ \mathcal{G}_{k-1}(y_{k-1}) - \mathcal{G}_k(y_k^{(0)}),$$

where $Q_{k,k-1}$ is another transfer operator. Now, $\mathcal{G}_k$ is replaced by $\mathcal{G}_k + P_k$ in the time-stepping such that

$$y_k^+ = y_k^{(0)} - \Delta t_k\ [\mathcal{G}_k(y_k) + P_k]$$

where the superscript $^+$ denotes the updated value. The resulting solution vector, y_k^+, then provides the initial data for grid $k+1$. Finally, the accumulated correction on grid k has to be transferred back to grid $k-1$ with the aid of an interpolation operator, $I_{k-1,k}$. Thus one sets

$$y_{k-1}^{++} = y_{k-1}^+ + I_{k-1,k}\ \left(y_k^{++} - y_k^{(0)}\right)$$

where the superscript $^{++}$ denotes the result of both the time step on grid k and the interpolated correction from grid $k+1$.

A W-cycle of the type shown below proves to be a particularly effective strategy for managing the work split between the meshes. In this figure, the solid nodes indicate that full evaluations are being performed

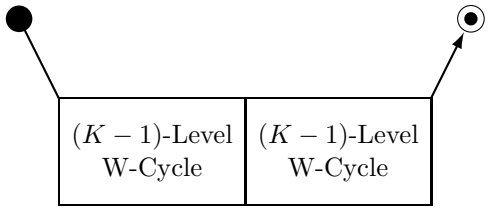
$k = 1$

$k = 2$

$k = 3$

Thr

while the open nodes indicate that only transfers of solution updates are computed.



Recursive Stencil for a
K-Level Multigrid W-Cycle

In a three-dimensional setting, the number of cells is reduced by a factor of 8 on each coarser grid. By examination of these illustrations, it can be verified that the work measured in units which correspond to an iteration on the finest grid is of the order of

$$1 + \frac{2}{8} + \frac{4}{64} + ... + \frac{1}{4^K} < \frac{4}{3}.$$

Since Eqn (35) is a parabolic equation, its convergence can be accelerated by standard multigrid techniques. This has been implemented here, with options for both V and W cycles.

5.5 Krylov Acceleration

Consider that K linearly-independent solution-gradient states are known. This system spans a K-dimensional Krylov subspace of the N-dimensional problem. If the system is linear, then the states supported by this

subspace can be described by the following recombination of known y-$\mathcal{G}$ pairs.

$$y^\star = \sum_{k=1}^{K} \gamma_k y^k \quad , \quad \mathcal{G}^\star = \sum_{k=1}^{K} \gamma_k \, \mathcal{G}^k,$$

where

$$\sum_{k=1}^{K} \gamma_k = 1.$$

Convergence of the descent optimization procedures can be accelerated by minimizing the L_2 Norm of $\mathcal{G}^\star$ to determine the appropriate recombination coefficients γ_k. See references [18, 19]. Now, the iterative process becomes

$$y_j^{n+1} = y_j^\star - \lambda \mathcal{G}_j^\star.$$

It should be noted that the above acceleration can fail if the preconditioned state of the current iteration is not linearly independent of the previous Krylov subspace. This can occur if the previous Krylov subspace happens to contain the zero-gradient state, and this "failure" is fortuitous since the converged solution is prematurely determined. Another cause for failure is a poor choice of preconditioner, however, this should not be the case with the method of steepest descent as the gradient at $y^\star$ should be nearly perpendicular to the previous Krylov subspace. In a linear setting, it is guaranteed that the new state as given by the preconditioner is not contained within the previous Krylov subspace if the L_2 Norm of the gradient of the new state is less than that of $\mathcal{G}^\star$ of the previous iteration.

5.6 Quasi-Newton Methods

Quasi-Newton methods are widely regarded as the method of choice for general optimization problems whenever the gradient can be readily calculated. These methods estimate the Hessian (A) or its inverse (A^{-1}) from changes in the gradient ($\delta\mathcal{G}$) during the search steps. By the definition of A, to first order

$$\delta\mathcal{G} = A \, \delta y.$$

Let H^n be an estimate of A^{-1} at the n^{th} step. Then it should be required to satisfy

$$H^n \delta\mathcal{G}^n = \delta y^n. \tag{41}$$

This can be satisfied by various recursive updating formulas for H, which also have the hereditary property that if A is constant, as is the case of a quadratic form, Eqn (41) holds for all of the previous steps $\delta y^k, k < n$. Consequently H becomes an exact estimate of A^{-1} in N steps.

Three quasi-Newton methods have been tested in this work. The first is the rank one (R1) correction

$$H^{n+1} = H^n + \frac{\mathcal{P}^n (\mathcal{P}^n)^T}{(\mathcal{P}^n)^T \delta\mathcal{G}^n}. \tag{42}$$

where

$$\mathcal{P}^n = \delta y^n - H^n \delta\mathcal{G}^n.$$

The second method is the Davidon-Fletcher-Powell (DFP) rank-two updating formula

$$\begin{aligned} H^{n+1} &= H^n \\ &+ \frac{\delta y^n (\delta y^n)^T}{(\delta y^n)^T \delta\mathcal{G}^n} - \frac{H^n \delta\mathcal{G}^n (\delta\mathcal{G}^n)^T H^n}{(\delta\mathcal{G}^n)^T H^n \delta\mathcal{G}^n}. \end{aligned} \tag{43}$$

In order to assure N-step convergence for a quadratic form, this method requires the use of exact line searches to find the minimum objective function in the search direction at each step. This is because the hereditary property depends on the orthogonality of each new gradient with the previous search direction, i.e., $(\mathcal{G}^{n+1})^T \delta y^n = 0$. Hence, many more evaluations of the objective function are required for DFP than R1. However, DFP has an advantage in that the updated estimates of the inverse Hessian, H^n, remain positive-definite.

The third quasi-Newton method tested uses the Broyden-Fanno-Goldfarb-Shannon (BFGS) updating formula

$$\begin{aligned} H^{n+1} &= H^n \\ &+ \left(1 + \frac{(\delta\mathcal{G}^n)^T H^n \delta\mathcal{G}^n}{(\delta\mathcal{G}^n)^T \delta y^n}\right) \frac{\delta y^n (\delta y^n)^T}{(\delta\mathcal{G}^n)^T \delta y^n} \\ &- \frac{H^n \delta\mathcal{G}^n (\delta y^n)^T + \delta y^n (\delta\mathcal{G}^n)^T H^n}{(\delta\mathcal{G}^n)^T \delta y^n}. \end{aligned} \tag{44}$$

This corresponds to the DFP formula applied to estimate A rather than A^{-1}. As with DFP, BFGS also requires exact line searches to assure N-step convergence for a quadratic form. In practice however, BFGS is usually preferred over DFP because it is considered to be more tolerant to inexact line searches, and therefore a more robust procedure.

We should note that the line searches performed for the Rank-2 quasi-Newton methods were implemented to locate the position where the local gradient is orthogonal to the search direction. For the discrete gradient, this location coincides with the minimum measurable cost function along the line. However, for the continuous gradient, this may not be the case. The motivation for this choice of line search (as opposed to finding the minimum measurable cost) is to ensure that the Rank-2 systems remain positive-definite for the continuous gradient optimizations. (See Reference [20], pp 54-55).

6 Results

The various methods described in this paper have been exercised to better understand their accuracy and performance. Specifically, three items are addressed. The first is whether the discrete or the continuous gradient is necessarily better than the other either by yielding more accurate solutions, or by reducing the cost of the search procedure. The second is whether an alternate search strategy can be devised which out performs the quasi-Newton approaches, when applied to trajectory and shape optimization problems, where the solution is expected to be smooth. And the third item discussed is regarding the robustness of the various methods as applied to the brachistochrone problem.

6.1 Accuracy

Both the continuous and discrete gradient-based optimizations have been applied to the brachistochrone problem over a wide range of mesh sizes. The model problem analyzed in this work is given by Eqns (31 & 32) with $C = 1$. Furthermore, to minimize the effects of the left-hand singularity, the boundary conditions were determined by using $t_0 = \frac{\pi}{2}$ and $t_1 = \pi$. A summary of this data related to accuracy is provided in Figures 17-21.

Figures 17 & 18 illustrate the convergence of the optimization process being driven by the continuous and discrete gradients, respectively. Here, the number of design variables is $N = 31$ and convergence is achieved using the implicit stepping approach. Figure 19 is analogous to Figure 17 for the case of $N = 511$ design variables. In these plots, the L_2 Norm of both gradients and a measure of the converged path's accuracy are provided. The accuracy of these results are monitored with the root-mean-square difference of the optimized discrete trajectory with that of the exact solution.

$$Y_{ERR} = \sqrt{\frac{\sum_{j=1}^{N} \left(y_j - y_{exact}(x_j)\right)^2}{N}}$$

Whether the optimization is driven by the continuous or discrete gradient, the convergence histories look very similar. The main difference between these results is that the converged value of Y_{ERR} levels off at a lower value for the continuous gradient ($10^{-4.619}$) than it does for the discrete gradient ($10^{-4.332}$). This character persists over the complete range of mesh sizes tested in this work as shown in Figure 20. Here, NX is the number of mesh intervals, thus $N = NX - 1$. While the advantage of the continuous gradient is apparent for the brachistochrone problem, in general, this may not be the case. Yet, these data clearly illustrate that an optimization based on the discrete gradient is not necessarily the best choice. In either case, Figure 20 illustrates that both approaches are second-order accurate.

Using Y_{ERR} as a measure of accuracy is only possible because the exact optimal solution is known for this problem. Typically, problems of optimization do not know what the exact solution is and therefore must rely on another figure of merit, usually the measurable cost function. For the brachistochrone problem, this is I_R. Even though Figure 20 clearly illustrates that the continuous gradient is more accurate than the discrete gradient, we know that I_R^C will be greater than (or equal to) I_R^D by definition. Here I_R^C is the measured cost function as optimized by the continuous gradient, while I_R^D is that optimized by the discrete gradient. This surplus, defined by

$$Surplus = I_R^C - I_R^D$$

is illustrated in Figure 21 as a function of mesh size. Interpretating this surplus as a disadvantage of the continuous gradient relative to the discrete gradient is unwarranted, as evidenced by Figure 20. However, since both approaches are second-order accurate, it should come as no surprise that this surplus also diminishes in a second-order manner with increasing number of design variables.

6.2 Performance

The performances of the various optimization techniques are verified in this section. Detailed illustrations of path histories as well as gradient convergences are given in Figures 22-38 for the different strategies on a problem involving $N = 31$ & 511 design variables.

Since an implicit stepping for the brachistochrone is possible, its performance is used as a goal to achieve in the design of an explicit scheme. As shown in Figures 22 & 24, the transient path shapes of the implicit scheme rapidly converges to the final state within a few iterations, regardless of problem dimension. The rapid convergence of this scheme is also exemplified by the quick decay of the corresponding gradients as shown in Figures 23 & 25.

Figures 26-29 provide path histories and gradient convergences for the three quasi-Newton methods. It is interesting to note that the character of the paths from iteration to iteration is distinctly different than that of the implicit scheme shown before in Figures 22 & 24. The main cause for this difference is that the quasi-Newton methods effectively take a uniform time step, whereas the other descent schemes investigated in this work employ a non-uniform time step dictated only by stability considerations. It is also interesting to note that for $N = 31$, the two methods which estimate A^{-1} (Rank-1 and DFP), follow nearly identical convergence histories, while the BFGS method requires about two extra iterations to converge the gradient norm to an equivalent level. For example, notice in Figure 27 that the Rank-1 and DFP schemes required 38 iterations to converge the gradient 6 orders of magnitude, while BFGS took about 40 cycles. These quasi-Newton methods were tested on problems up to $N = 511$ design variables; they consistently exhibited a convergence behavior where the gradient norm remains relatively flat until N iterations have been performed, then rapidly converging to machine-level zero in about 10 more steps. This behavior is precisely what the theory suggests should occur and this trend is definitely supported by Figures 27 & 29 for $N = 31$ & 511, respectively. However, the reader is reminded that the Rank-1 algorithm does not require a line search each iteration, and therefore may evaluate the gradient up to an order of magnitude fewer times than either of the Rank-2 schemes. Robustness issues aside, this can result in a significant difference in the computational times between the various quasi-Newton methods.

Figures 30-31 give the results of the steepest descent approach. Recall that in Section 5.1 the stability analysis of this stepping concluded that the number of iterations required for convergence was directly proportional to N^2. Even for the small case of $N = 31$ shown in these figures, more than 512 cycles were performed before the path visually converged and almost 6000 iterations were needed before machine-level zero was attained. (Because of the CPU requirements involved, a solution for $N = 511$ was not even attempted.) Clearly, this approach is unacceptable in the setting of large engineering problems of interest. In this solution, $STEP = 1.0$ was used as it provides the fastest convergence rates for the steepest descent approach. A test with $STEP = 1.01$ verified our stability analysis as the solution quickly diverged.

In contrast with the previous results, the path history for a smoothed descent is illustrated in Figure 32. In this solution, $STEP = 100$ was used and to the eye, the final path is determined in less than 8 cycles. Figure 33 gives the convergence histories for a variety of values of $STEP = 12.5, 25, 50, 100$. These data confirm that a doubling of the step size essentially halves the number of iterations required for convergence, up to a point.

Figures 34-35 provide the results of applying the smoothed descent scheme coupled with Krylov acceleration. In this case, the path history looks to settle down after only 4 iterations while the gradient norm

exhibits a quadratic-like convergence behavior. Figure 35 indicates that the Krylov accelerated convergence is almost invariant of the value of $STEP$ in the preconditioned step. To obtain 6 orders of magnitude reduction requires between 12 & 14 steps for $STEP \geq 25$. For reference, this figure also includes the best convergence curve of Figure 33 to better illustrate the positive effect that this acceleration scheme has on the convergence of the optimization process.

Application of a multigrid W-cycle to drive the convergence of the brachistochrone is now studied. Figures 36-37 give the corresponding path history and gradient convergence plots as seen before. Amazingly, the path of the second cycle is almost within symbol width of the final path. The effect of number of mesh levels, $NMESH$, used in the multigrid convergence is shown in Figure 37. Here, with $NMESH = 1$, a smoothed descent scheme is recovered; increasing the number of multigrid levels significantly increases the convergence rate. Furthermore, while this figure depicts the convergence for $NX = 32$, it was observed that the multigrid convergence history was grid independent if one always used $NMESH = log_2(NX)$.

The enhancements incorporated beyond the original steepest descent method have systematically improved the convergence of the optimization process to the point where the performance of the implicit scheme has been achieved. The final enhancement to the multigrid scheme is to incorporate the Krylov acceleration. Figure 38 compares the complete ensemble of enhancements to our explicit scheme with the performance of the implicit stepping. For all practical purposes, the convergence rates of these two schemes are identical. For comparison, this figure also includes the $NMESH = 5$ multigrid convergence of Figure 37, which is noticeably slower than that with the Krylov acceleration. However, the convergence histories depicted in this figure, for these three schemes, are essentially independent of the dimensionality, N. (See Figure 39.)

Figure 39 compares the performance of several of the various schemes. Here, $ITERS$ is the number of iterations required for the gradient norm to be reduced at least six orders of magnitude. This figure represents problems ranging in size from $N = 3$ design variables up to $N = 8191$. (In this figure, NX is the number of mesh intervals, thus $N = NX - 1$.) The basic steepest descent approach exhibits an $\mathcal{O}(N^2)$ behavior in the number of iterations required, while the quasi-Newton methods indicate $\mathcal{O}(N)$ as expected. Using the full number of multigrid levels and Krylov acceleration provides a grid independent result, requiring about 8 iterations to converge 6 orders. Finally, this is shown to compare favorably with the implicit stepping performance which also yields a grid independent character, requiring only 7 iterations.

6.3 Robustness

We include a brief, qualitative synopsis of the robustness of the various schemes studied herein. While no attempt was made to establish a metric which one could monitor a scheme's level of robustness, the successful application of some of these methods proved to be sensitive to certain implementation issues. For example, and previously noted, the Krylov acceleration can "bomb" if a machine-level exact solution is found in less than N iterations. Figure 39 clearly shows that this occurred for all cases where $N > 8$.

The behavior of the robustness of the various quasi-Newton schemes seemed contrary to popular understanding. The Rank-1 scheme had no issue converging any of the cases tested. This included optimizations driven by both the continuous and discrete gradients for problems with $3 \leq N \leq 511$ design variables. However, the Rank-2 schemes were fairly sensitive to the level of convergence of the line searches, and surprisingly, BFGS seemed more sensitive to "inexact" line searches than DFP.

Subject only to the stability limits discussed in § 5.1 & 5.2, the explicit time-stepping schemes of steepest descent, smoothed steepest descent, and multigrid descent proved to be very robust. These optimizations converged as anticipated for every case tested. This included optimizations based on both the continuous and discrete gradients for problems up to $N = 8191$ design variables.

7 Conclusions

In this lecture, two model problems of optimization have been discussed. The first model problem of the spider-and-fly was used to introduce some basic concepts of optimization set up; it also illustrated some mental *traps* to avoid. The second model problem of the brachistochrone was used to develop various aspects of the optimization process in much greater detail.

In the spider-fly problem, we showed how one can set up the objective function and its associated design space, including constraints. Here, the cost function of the discrete problem was equivalent to that of the

continuum; this is normally not the case. Direct differentiation of the cost function was used to establish the gradient and Hessian matrix; also equivalent to that of the continuum. Steepest descent, Newton iteration, quasi-Newton, and Nash equilibrium methods were demonstrated. One *trap* avoided was on the choice of which of these methods is most appropriate for optimizations that utilize design spaces of very large dimension. Another *trap* discussed was how a developer envisions the set up of the discrete problem.

A variety of optimization techniques have been applied to the brachistochrone problem. These include gradient approaches based on both the continuous as well as discrete forms. Results show that, at least for the brachistochrone problem, the continuous gradient yields a slightly more accurate solution than does the discrete gradient; a possible reason for this outcome is included.

The solution of the nonlinear optimization problem is accomplished with explicit and implicit time stepping schemes which are also compared with several quasi-Newton algorithms. The data presented herein illustrate that an explicit time-stepping scheme can be constructed whose convergence properties are invariant with the dimensionality of the problem. Furthermore, the performance of this explicit scheme rivals that of the implicit time stepping. These trends have been verified on problems ranging from $N = 3$ design variables up to $N = 8191$ unknowns.

Finally, the performance of quasi-Newton methods is consistent with theoretical estimates in that their convergence (i.e., iterations required) is linearly dependent on the number of design variables.

For large engineering optimization problems of interest, $N > \mathcal{O}(10^3)$, it is obvious that quasi-Newton methods can become prohibitively expensive in terms of both computational and memory requirements. However, it is encouraging that the possibility exists of constructing an efficient explicit scheme based on the straightforward techniques investigated under this research.

References

[1] H. E. Dudeney. The Spider and The Fly. *The Weekly Dispatch Newspaper*, 14 June 1903. Also published in *The Canterbury Puzzles and Other Curious Problems*, London, 1907.

[2] W. K. Anderson, III J. C. Newman, D. L. Whitfield, and E. J. Nielsen. Sensitivity analysis for Navier-Stokes equations on unstructured meshes using complex variables. *AIAA Journal*, 39(1):56–63, January 2001. Also, AIAA Paper 99-3294.

[3] S. Boyd and L. Vandenberghe. *Convex Optimization*. Cambridge University Press, New York, NY, 2004. ISBN 0 521 83378 7.

[4] Johann Bernoulli. *Acta Eruditorum*. J.F. Gleditsch Publishing, Leipzig, Germany, June 1696. Otto Mencke, Editor.

[5] Galileo Galilei. *Discourses and mathematical demonstrations concerning the two new sciences*. Published, Leyden, Holland, 1638.

[6] A. Jameson and J. C. Vassberg. Studies of alternative numerical optimization methods applied to the brachistochrone problem. *Computational Fluid Dynamics Journal*, 9(3), October 2000. Special Issue on CFD and Education, Dedicated to Dr. Koichi Oshima; Emeritus Professor of University of Tokyo.

[7] A. Jameson and J. C. Vassberg. Studies of alternative numerical optimization methods applied to the brachistochrone problem. In *Proceedings of OptiCON'99*, Newport Beach, CA, October 1999.

[8] G. B. Thomas. *Calculus and Analytic Geometry*. Addison-Wesley, Reading, MA, 1960. Series in Mathematics.

[9] E. Chong and S. Zak. *Introduction to Optimization*. Wiley, New York, NY, 1995.

[10] R. P. Fedorenko. The speed of convergence of one iterative process. *Comp. Mathematics and Mathematical Physics*, 4:227–235, 1964.

[11] A. Brandt. Multi-level adaptive solutions to boundary value problems. *Mathematical Computing*, 31:333–390, 1977.

[12] W. Hackbusch. On the multi-grid method applied to difference equations. *Computing*, 20:291–306, 1978.

[13] R. H. Ni. A multiple grid scheme for solving the Euler equations. *AIAA Journal*, 20:1565–1571, 1982.

[14] M. G. Hall. Cell vertex multigrid schemes for solution of the Euler equations. Reading, April 1985. Proceedings IMA Conference on Numerical Methods for Fluid Dynamics.

[15] A. Jameson. Multigrid algorithms for compressible flow calculations. In W. Hackbusch and U. Trottenberg, editors, *Lecture Notes in Mathematics, Vol. 1228*, pages 166–201. 2nd European Conf. Multigrid Methods, Cologne, 1985, Springer-Verlag, 1986.

[16] A. Jameson. Solution of the Euler equations for two dimensional transonic flow by a multigrid method. *Applied Mathematics and Computations*, 13:327–356, 1983.

[17] A. Jameson. Solution of the Euler equations by a multigrid method. *Applied Mathematics and Computations*, 13:327–356, 1983.

[18] Y. Saad. *Iterative Methods for Sparse Linear Systems*. PWS Publishing Company, Boston, MA, 1996. PWS Series in Computer Science.

[19] R. W. Clark. A new iterative matrix solution procedure for three-dimension panel methods. *AIAA paper 85-0176*, AIAA Aerospace Sciences Meeting, Reno, NV, January 1985.

[20] R. Fletcher. *Practical Methods of Optimization*. John Wiley & Sons, New York, NY, 1995. Second Edition.

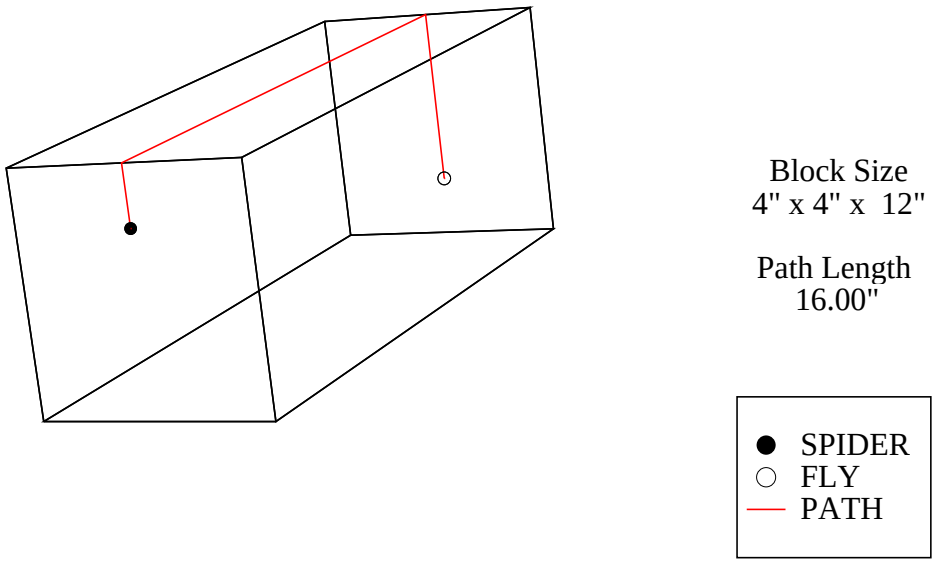


Figure 1: Obvious local-minimum path between Spider and Fly.

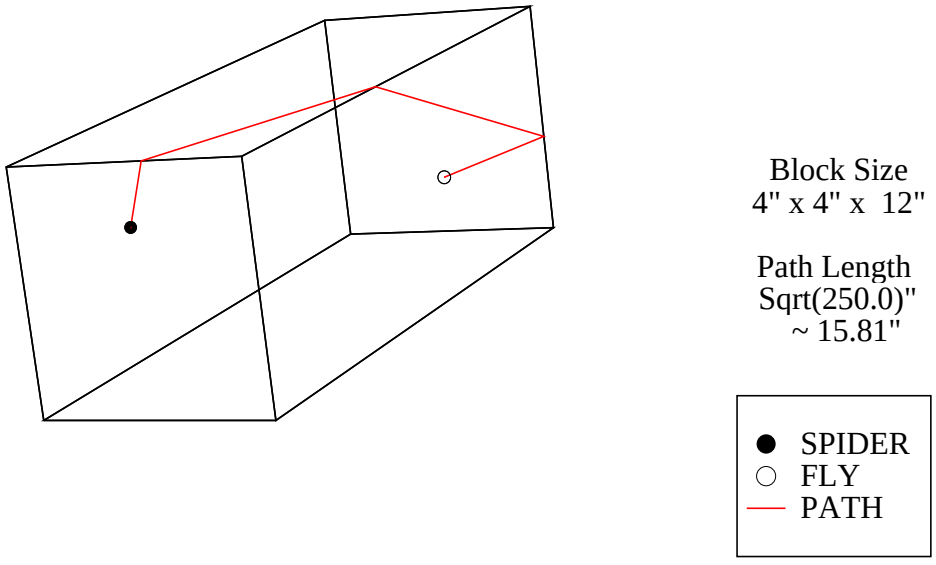


Figure 2: Non-obvious global-minimum path between Spider and Fly.

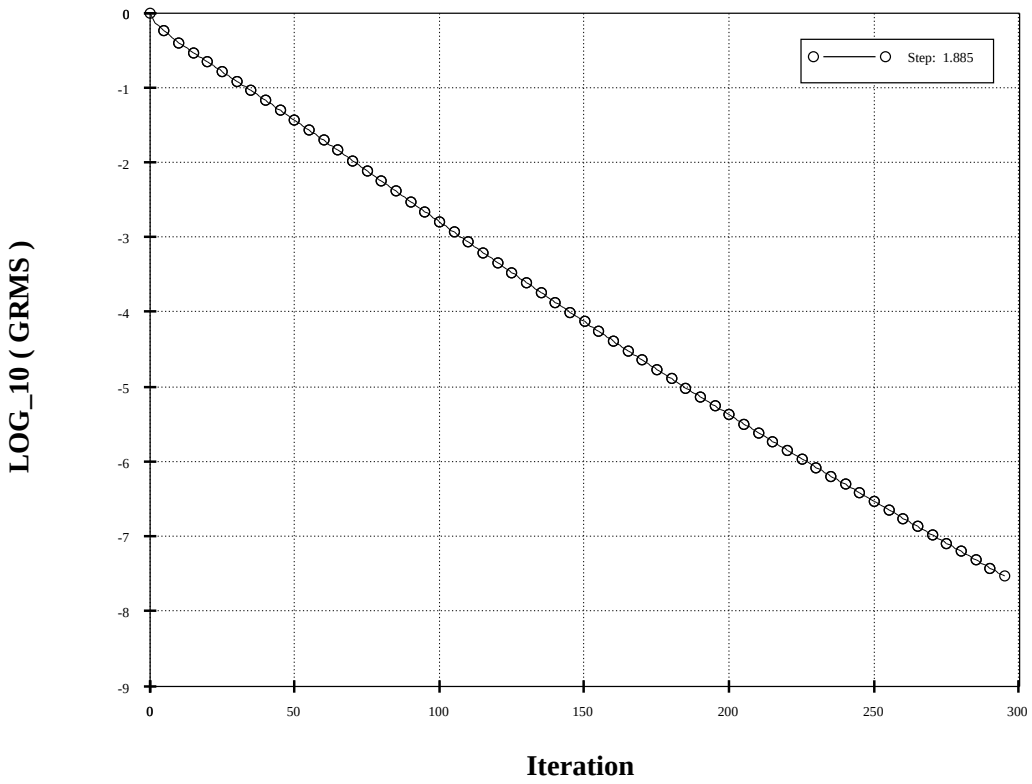


Figure 3: Convergence of Gradient for Steepest Descent.

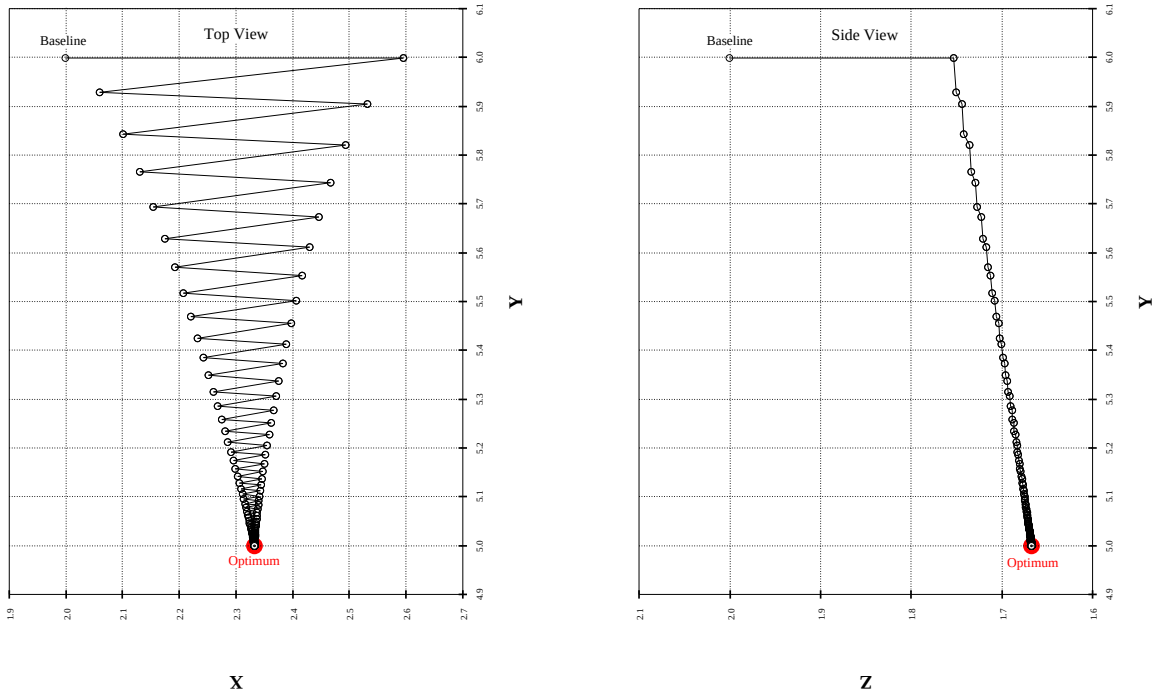


Figure 4: Steepest-Descent Trajectory through the Design Space.

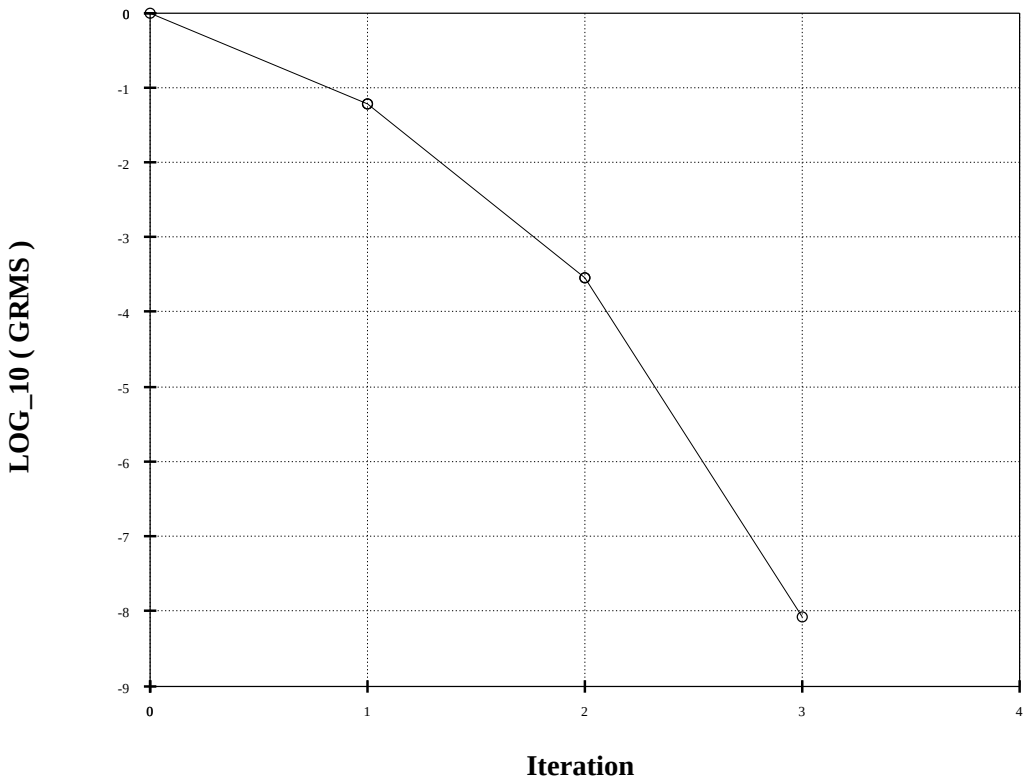


Figure 5: Convergence of Gradient for Newton Iteration.

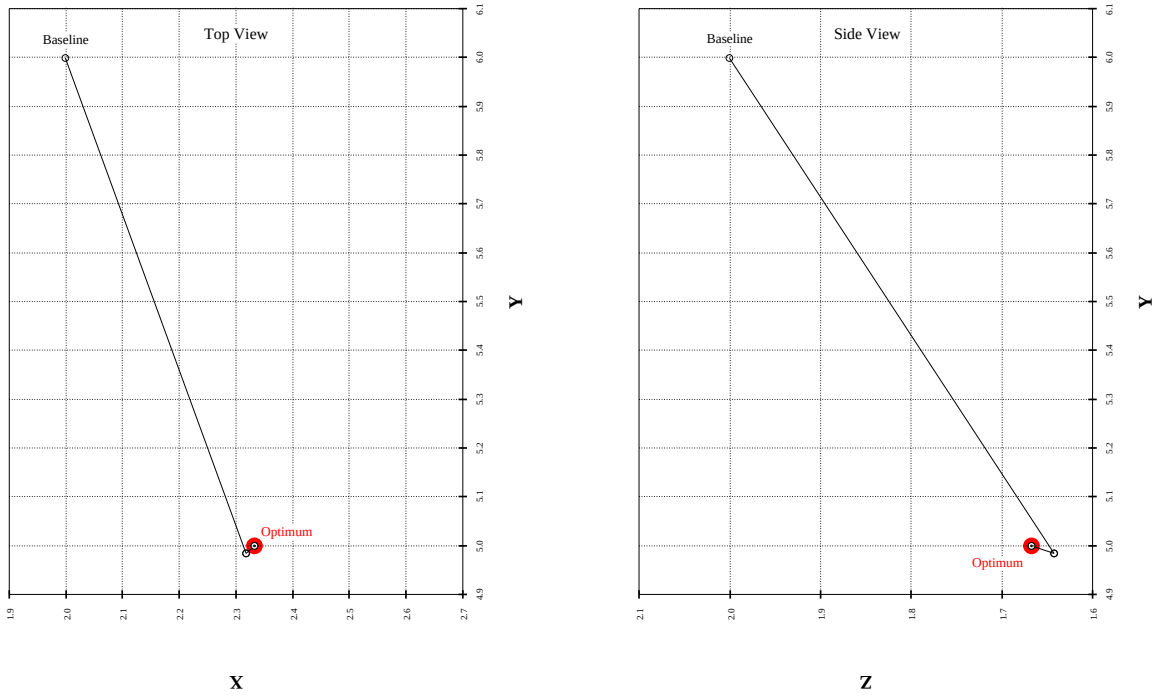


Figure 6: Newton-Iteration Trajectory through the Design Space.

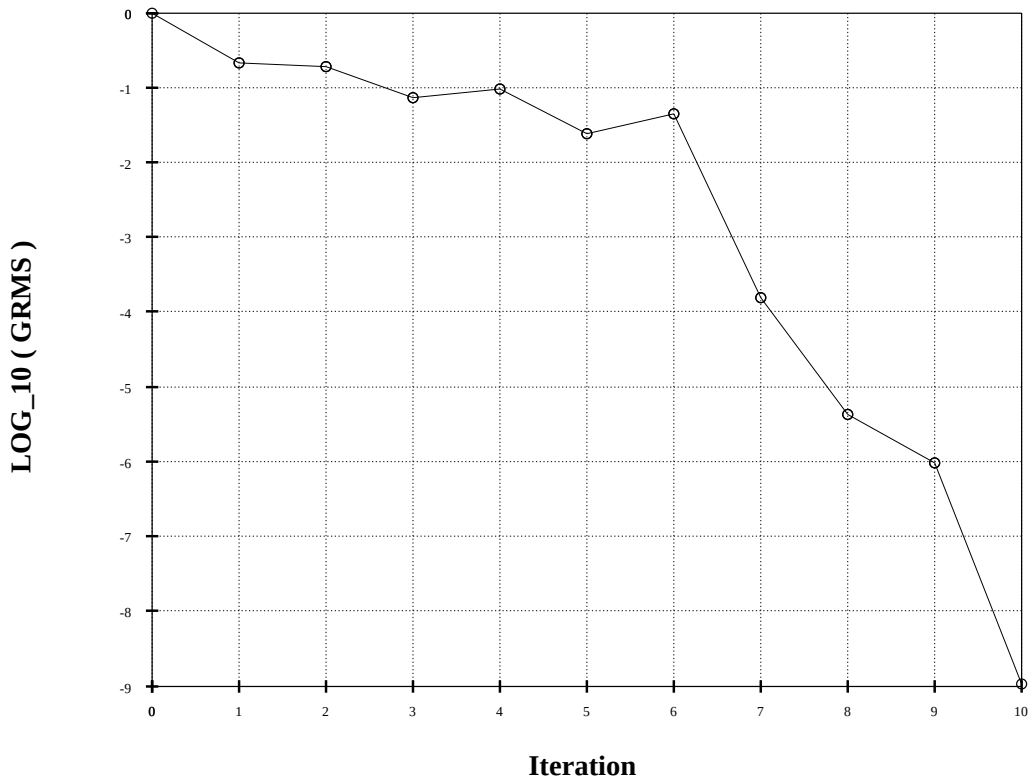


Figure 7: Convergence of Gradient for Rank-1 quasi-Newton Iteration.

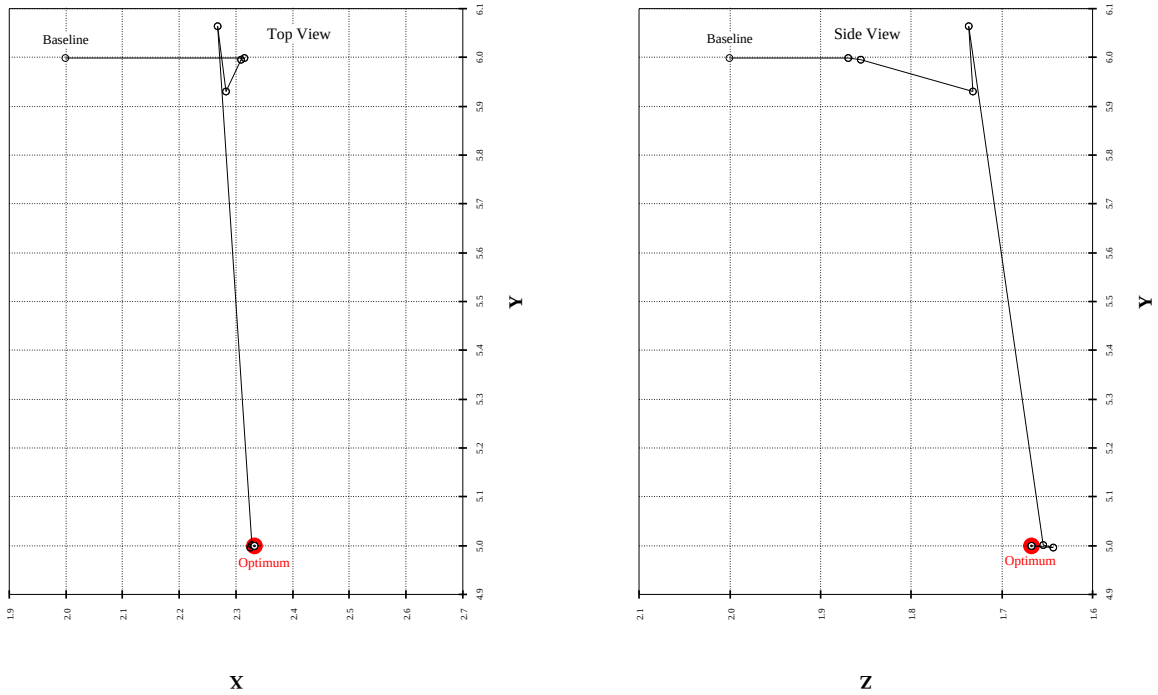


Figure 8: Rank-1 quasi-Newton Trajectory through the Design Space.

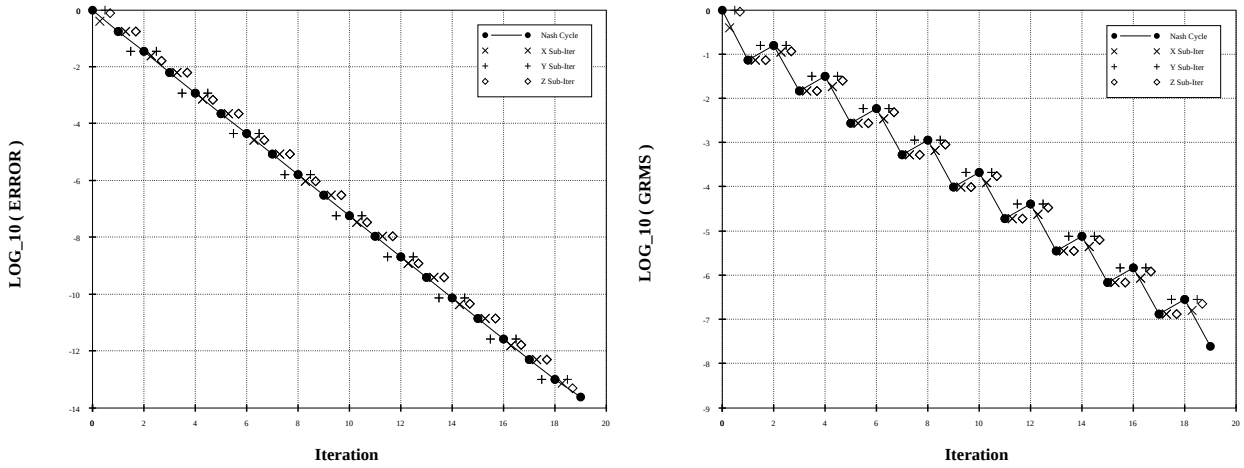


Figure 9: Convergence of Error and Gradient for Nash Equilibrium.

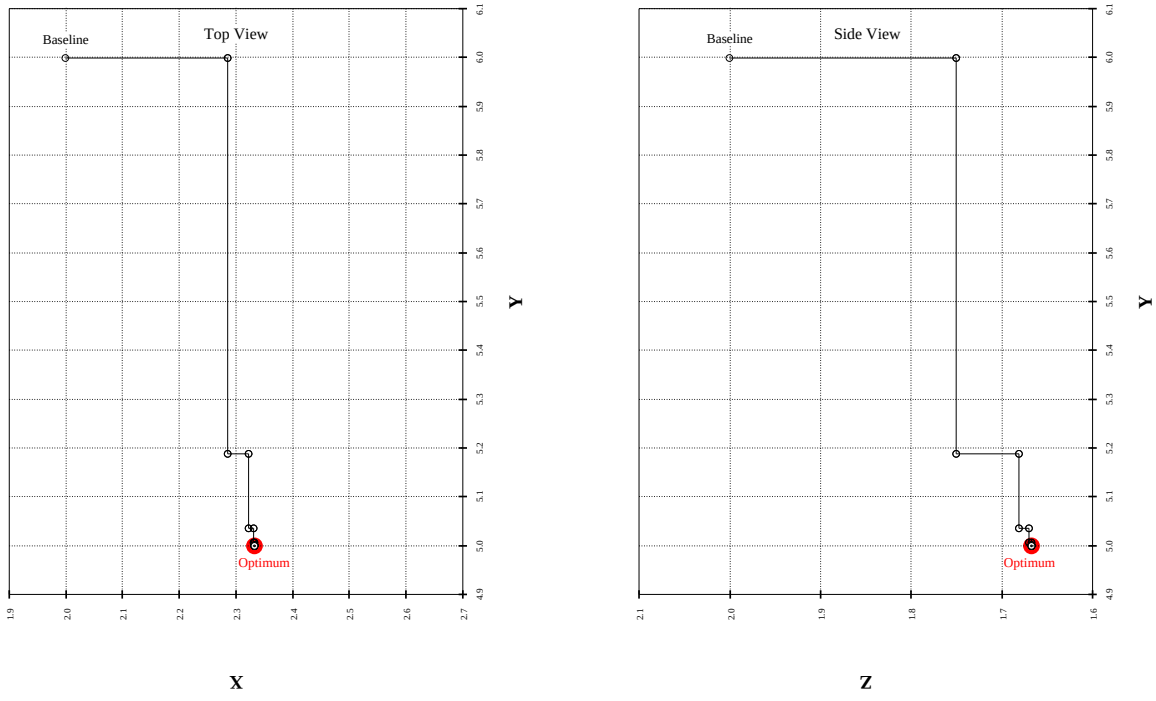


Figure 10: Nash Equilibrium Trajectory through the Design Space.

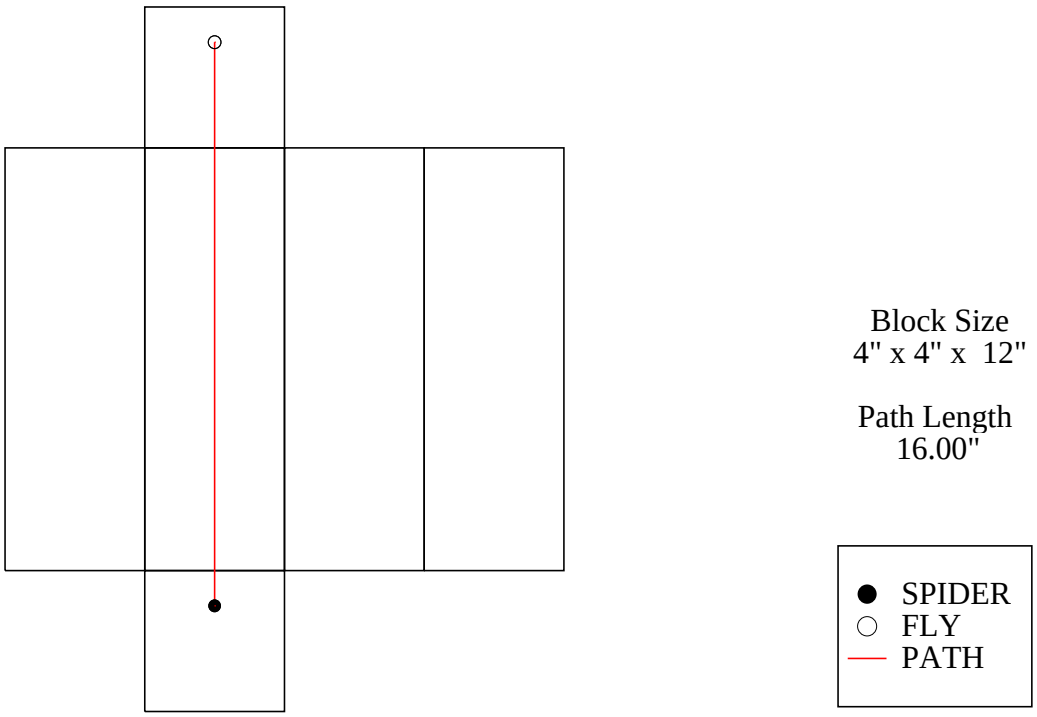


Figure 11: Obvious local-minimum path between Spider and Fly on flattened box.

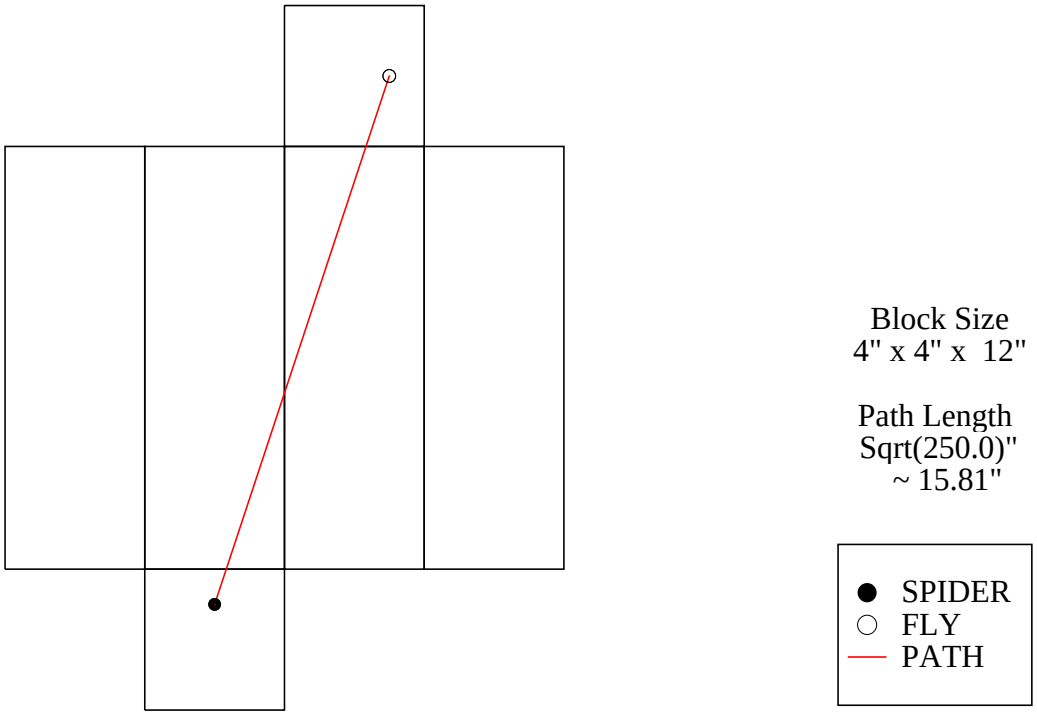


Figure 12: Non-obvious global-minimum path between Spider and Fly on flattened box.

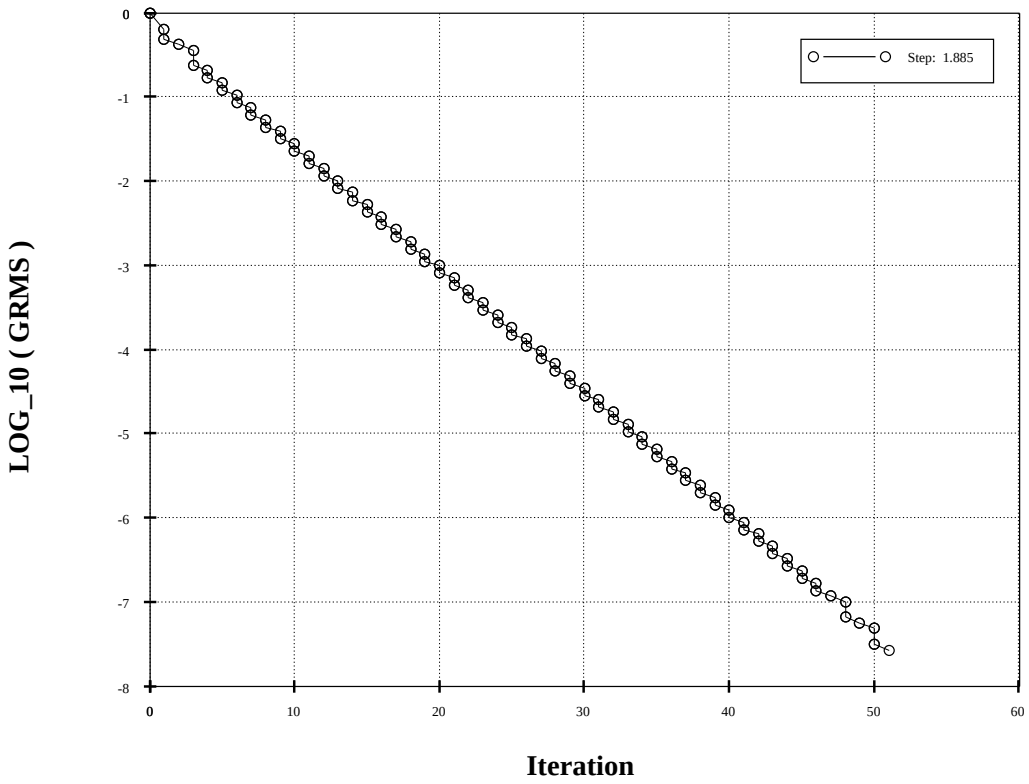


Figure 13: Convergence of Gradient for Steepest Descent with $S_2 \leq 4$.

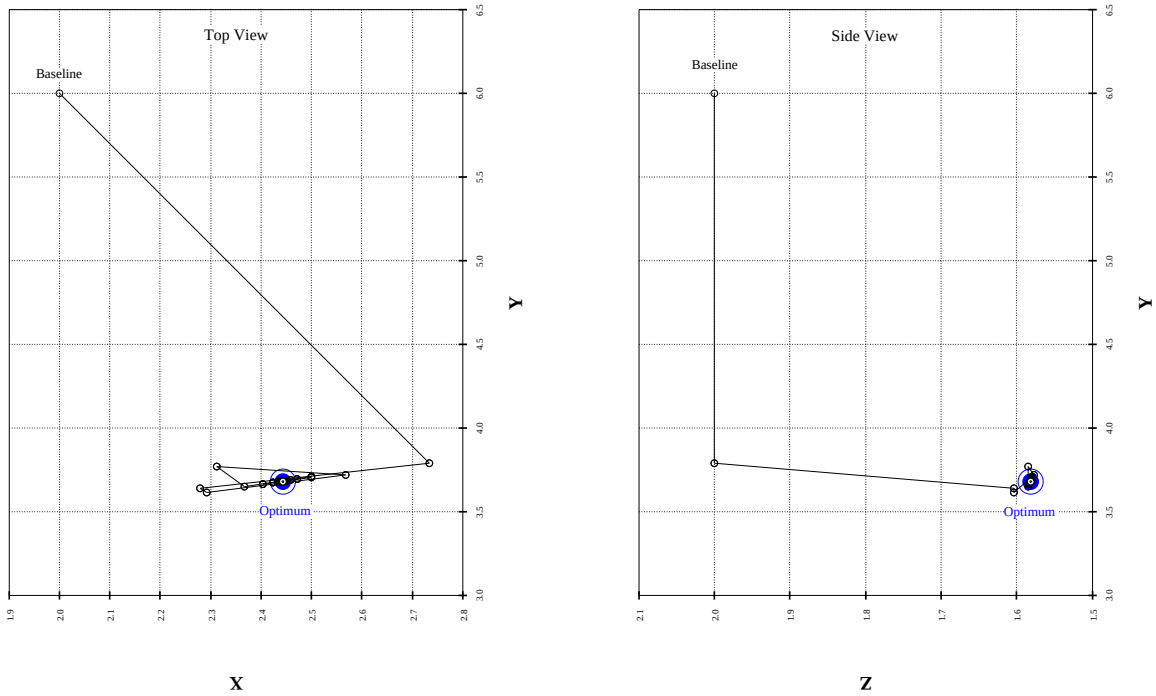


Figure 14: Steepest-Descent Trajectory through the Design Space with $S_2 \leq 4$.

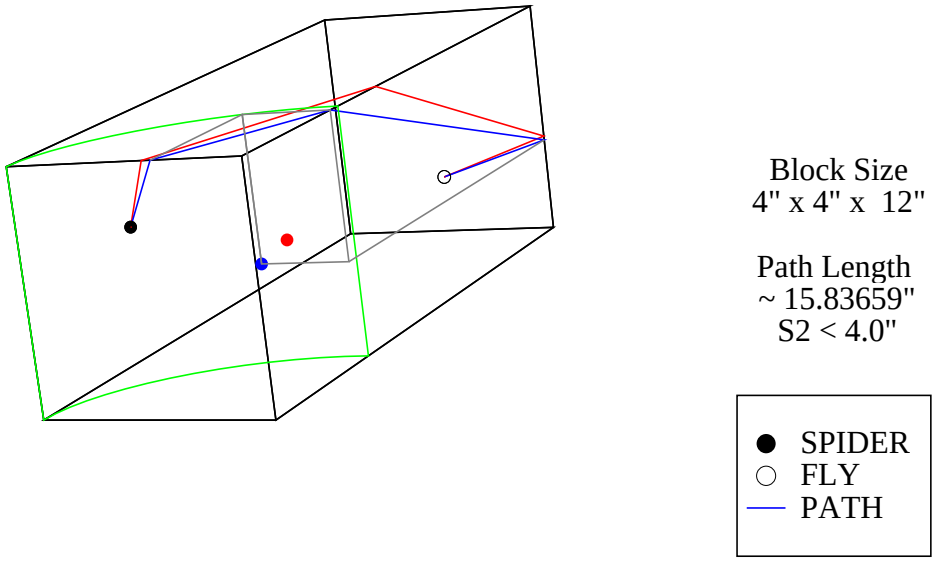


Figure 15: Minimum path between Spider and Fly with $S_2 \leq 4$.

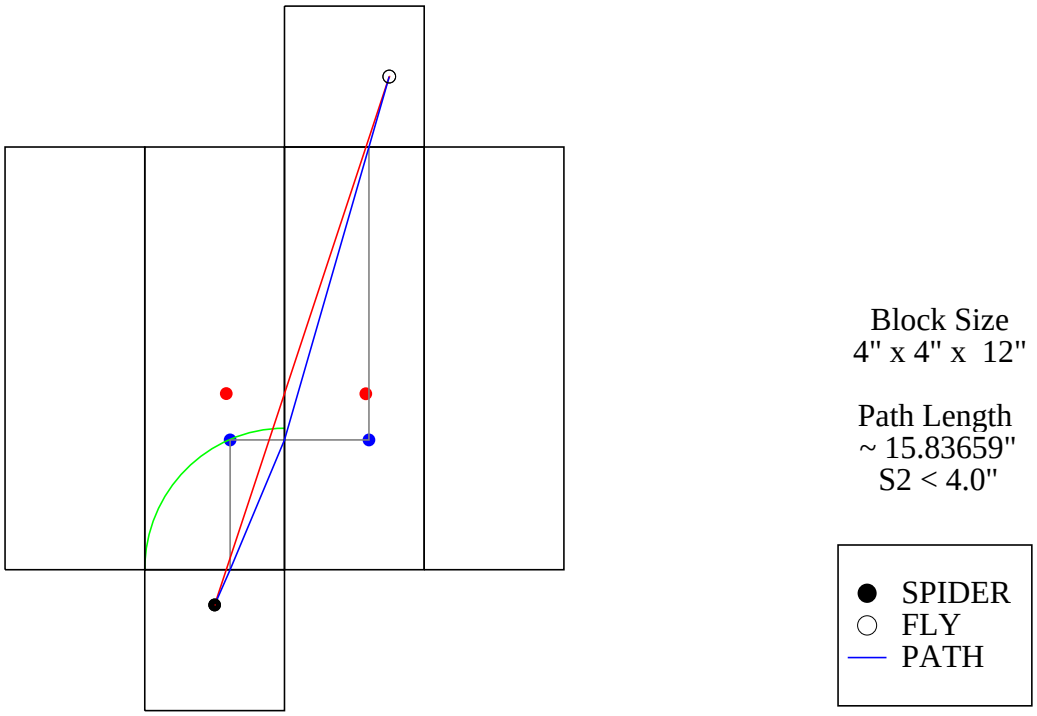


Figure 16: Minimum path between Spider and Fly on flattened box with $S_2 \leq 4$.

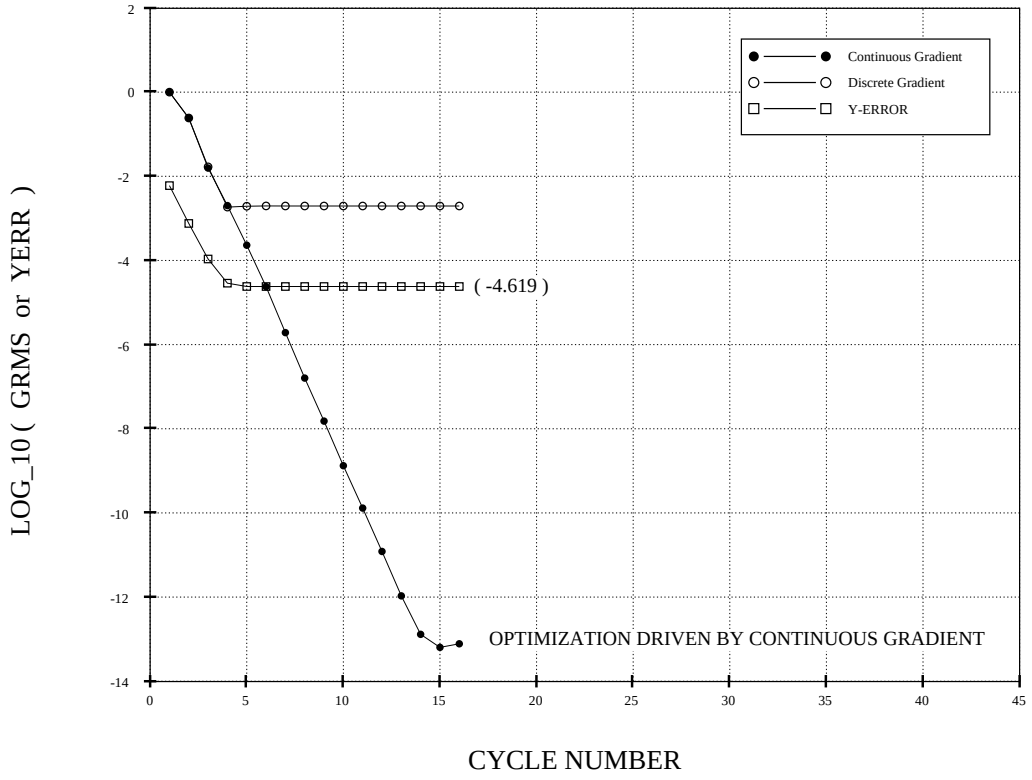


Figure 17: Convergence histories of implicit stepping using continuous gradient with N=31.

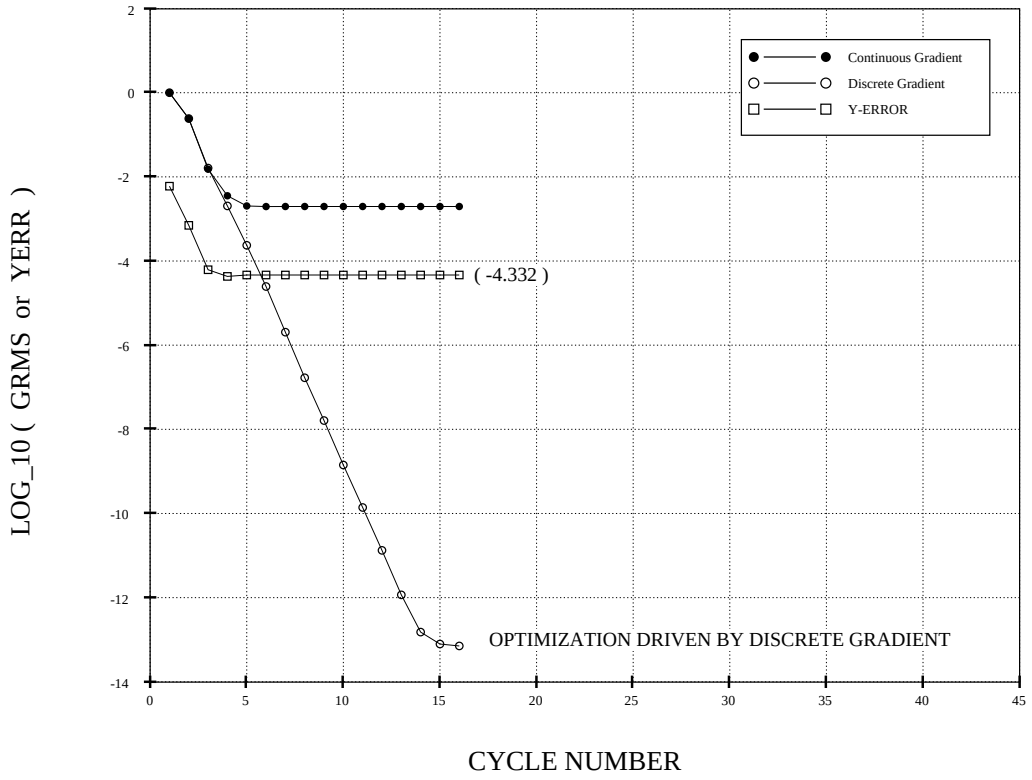


Figure 18: Convergence histories of implicit stepping using discrete gradient with N=31.

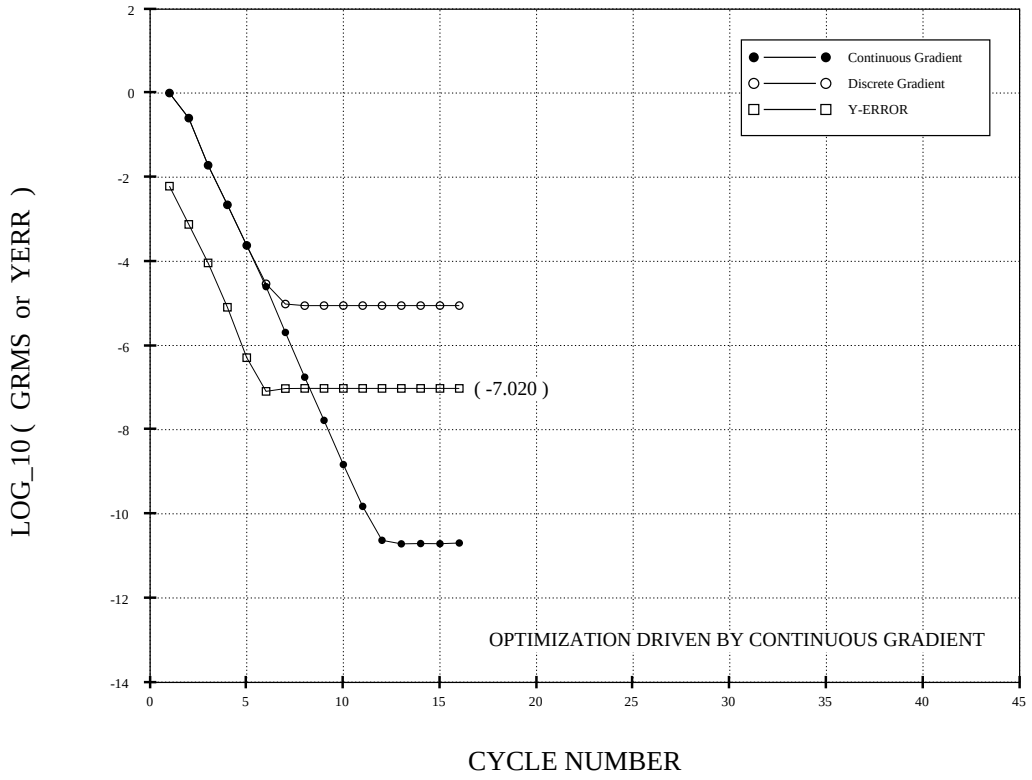


Figure 19: Convergence histories of implicit stepping using continuous gradient with N=511.

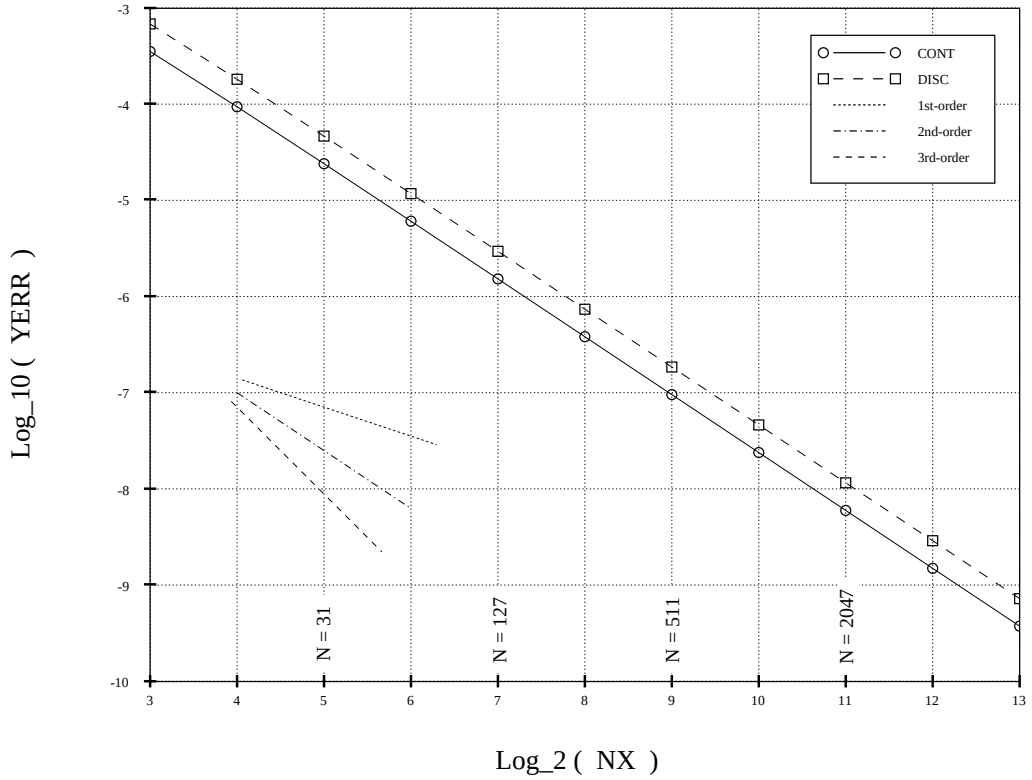


Figure 20: Computed path errors as a function of mesh size.

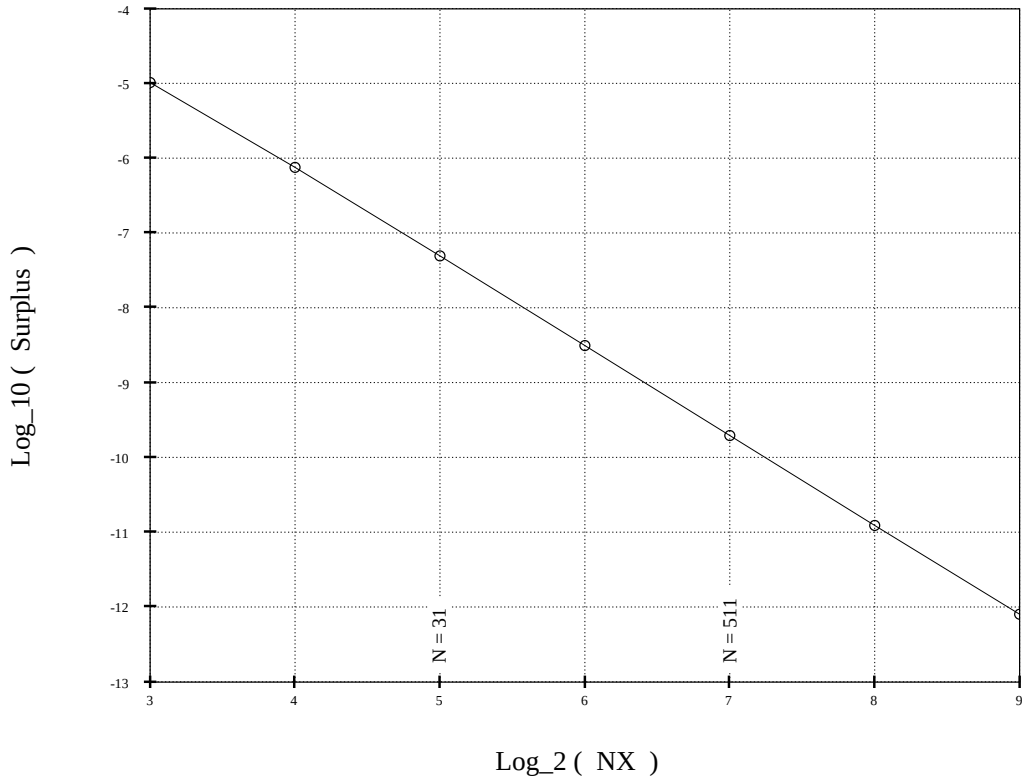


Figure 21: Difference of measurable cost function between the continuous and discrete gradients.

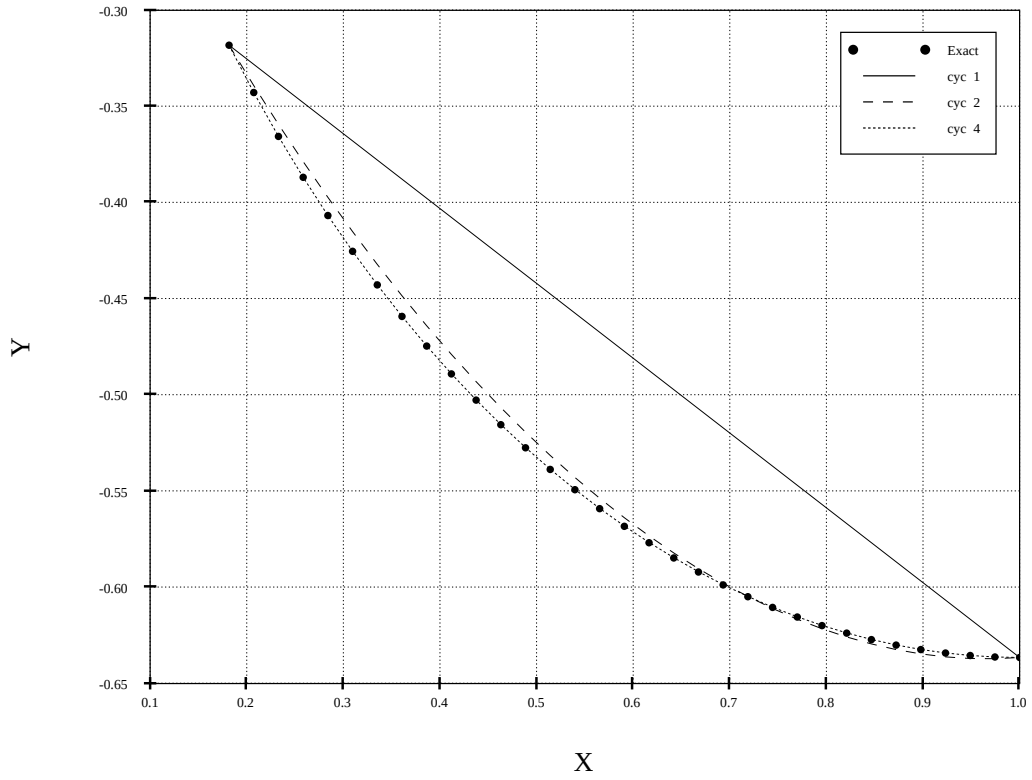


Figure 22: History of paths of implicit stepping with N=31.

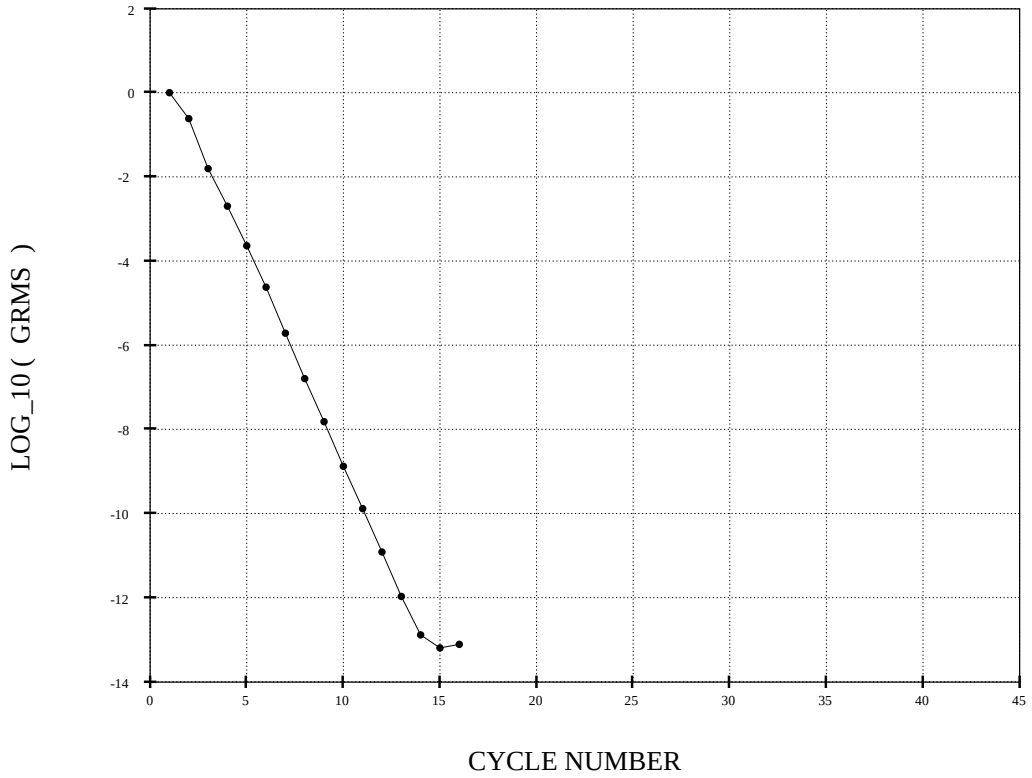


Figure 23: Convergence history of implicit stepping with N=31.

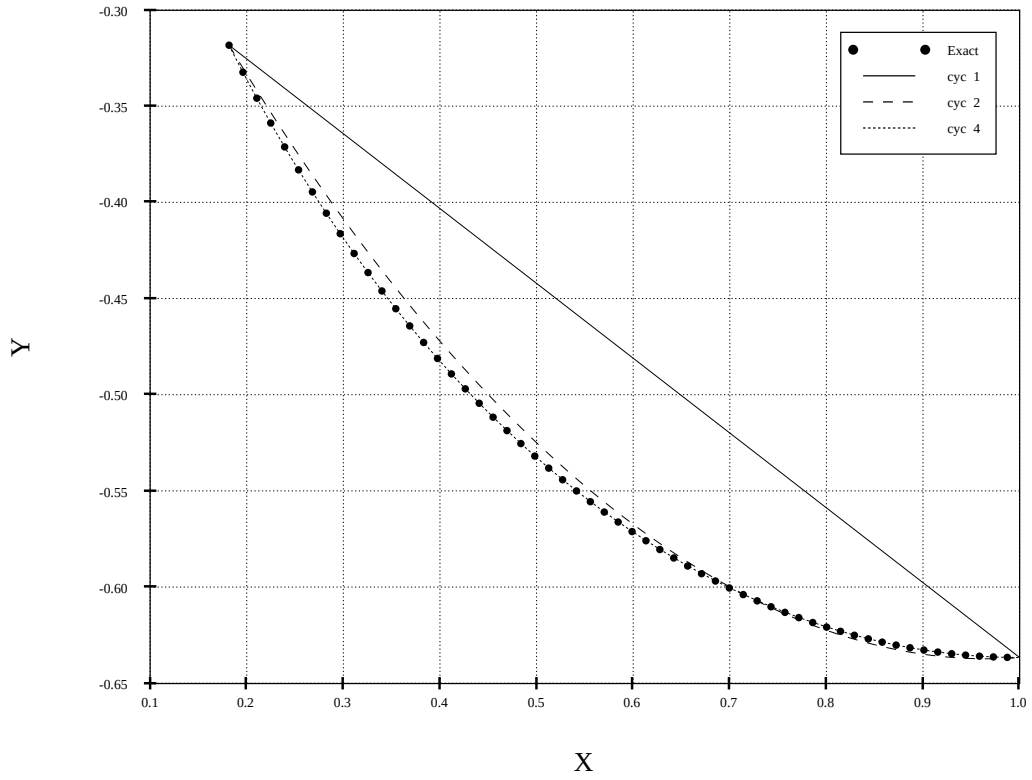


Figure 24: History of paths of implicit stepping with N=511.

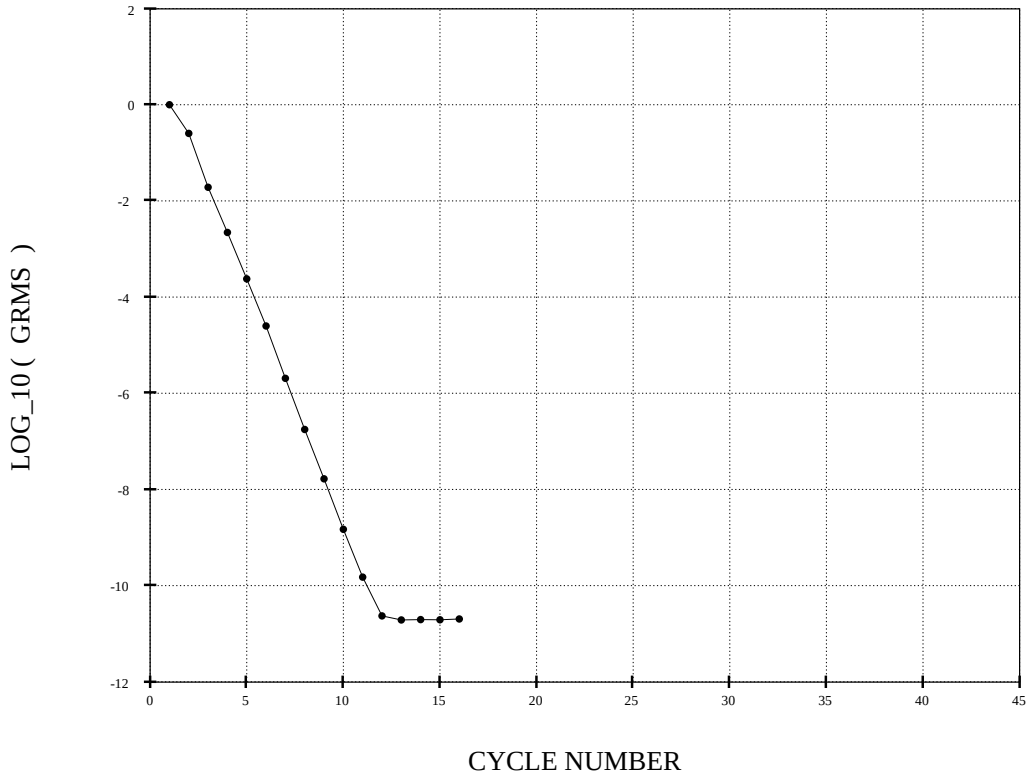


Figure 25: Convergence history of implicit stepping with N=511.

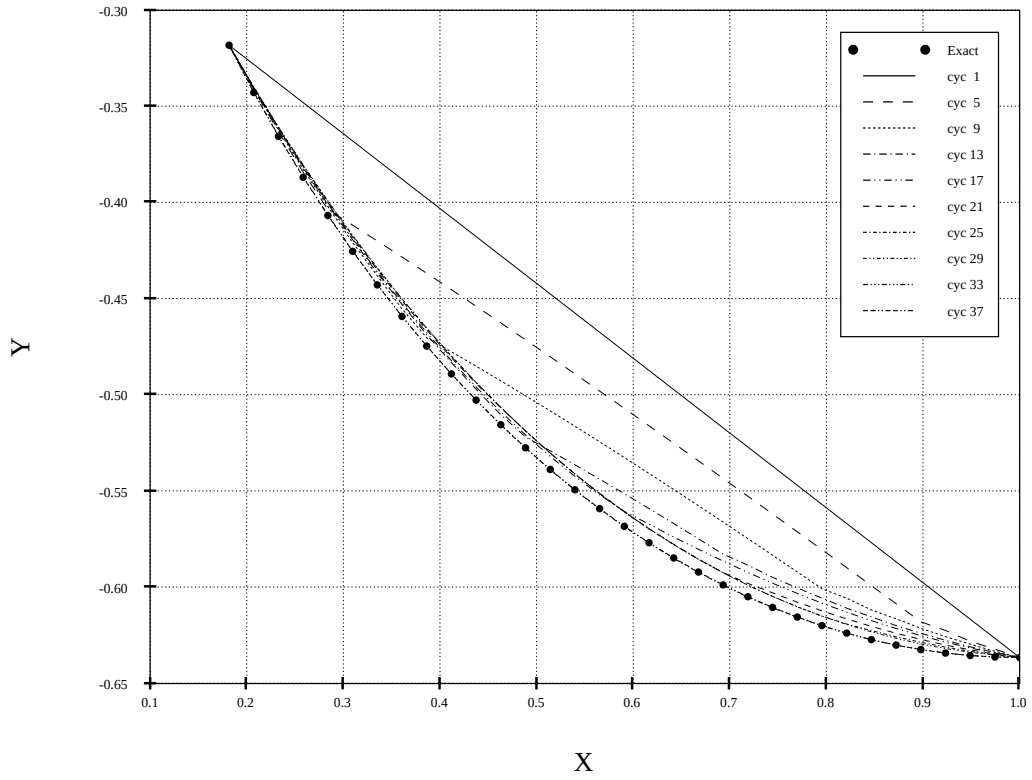


Figure 26: History of paths for Rank-1 quasi-Newton with N=31.

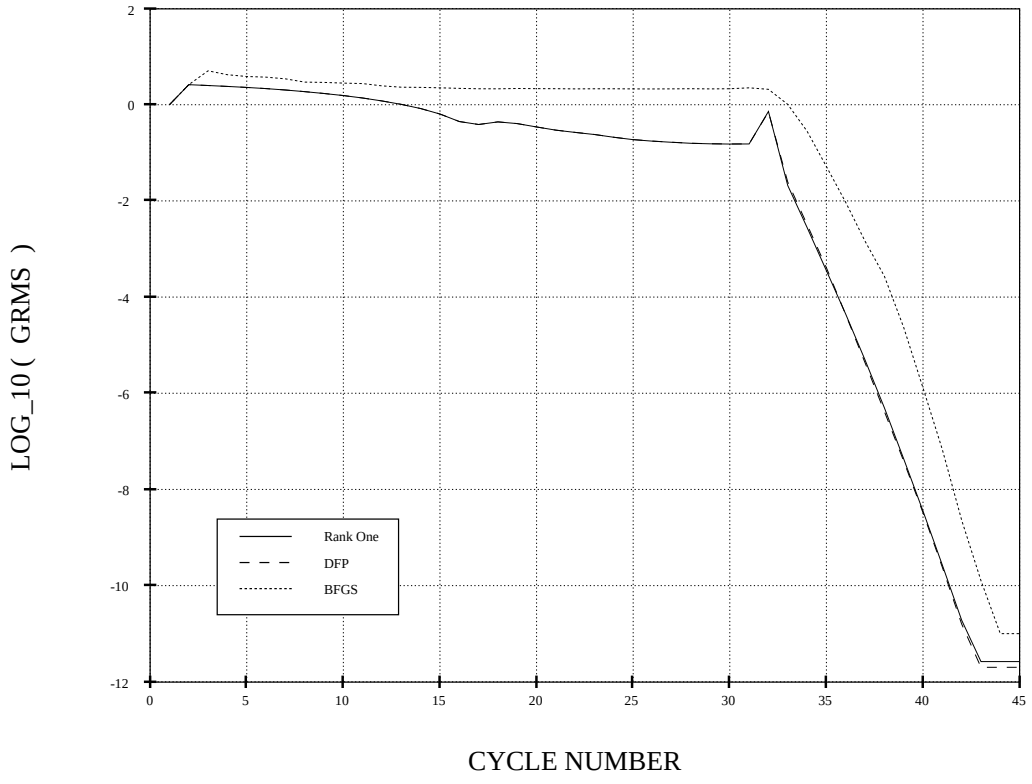


Figure 27: Comparison of quasi-Newton convergence histories with N=31.

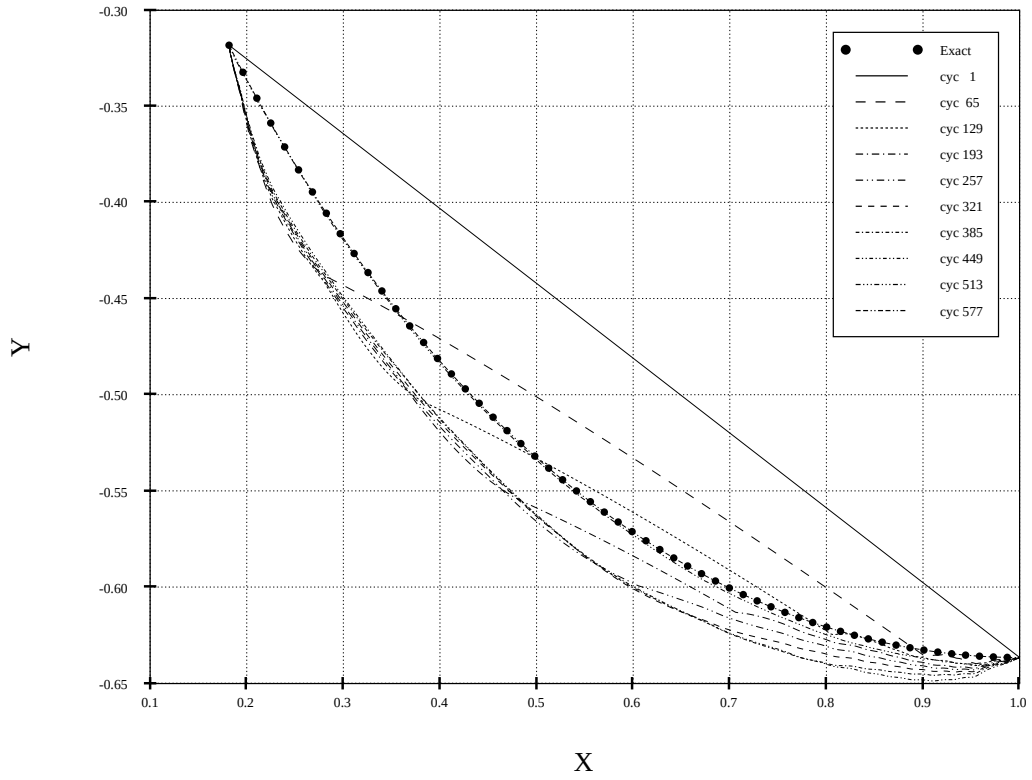


Figure 28: History of paths for Rank-1 quasi-Newton with N=511.

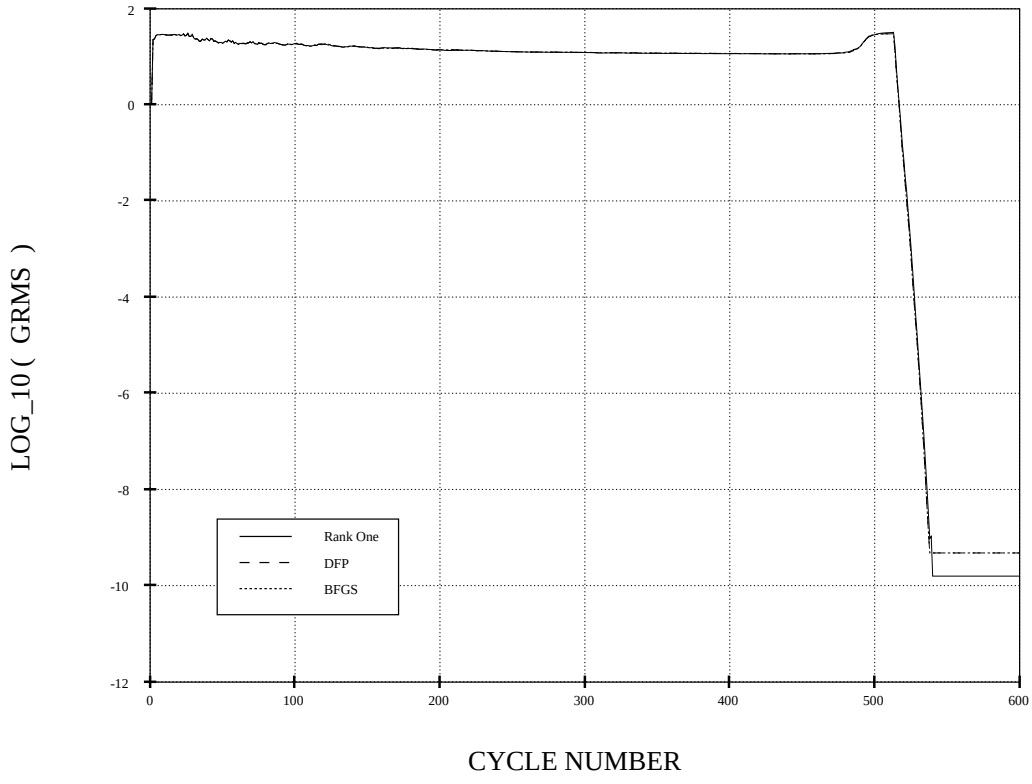


Figure 29: Comparison of quasi-Newton convergence histories with N=511.

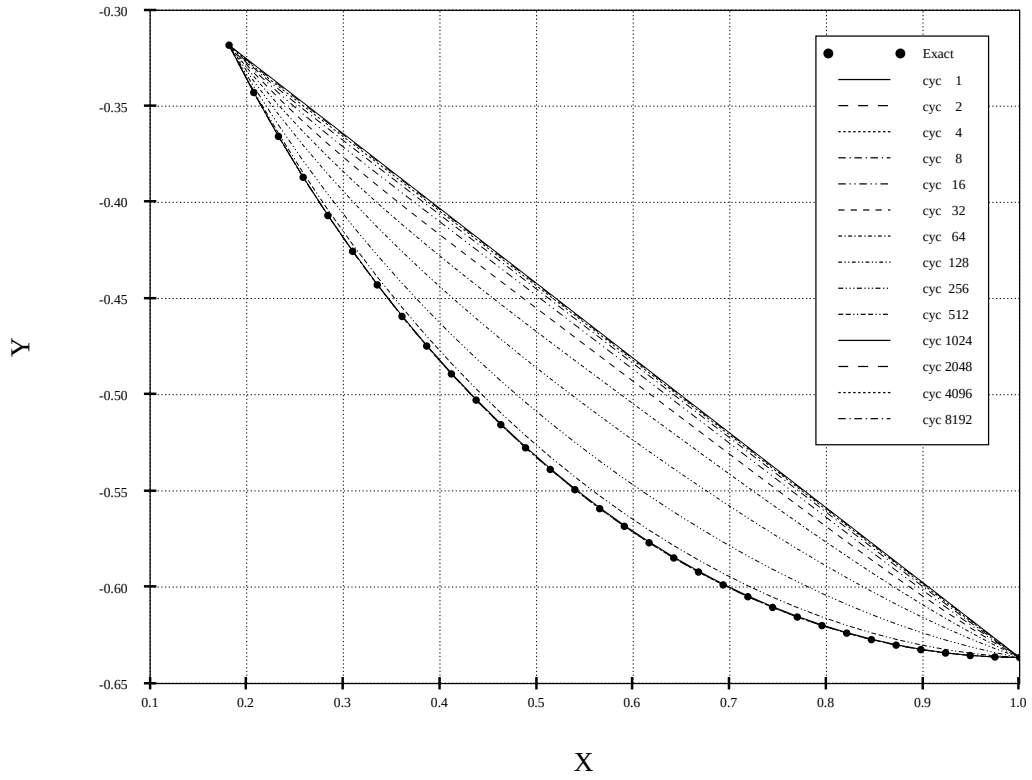


Figure 30: History of paths of steepest descent with N=31.

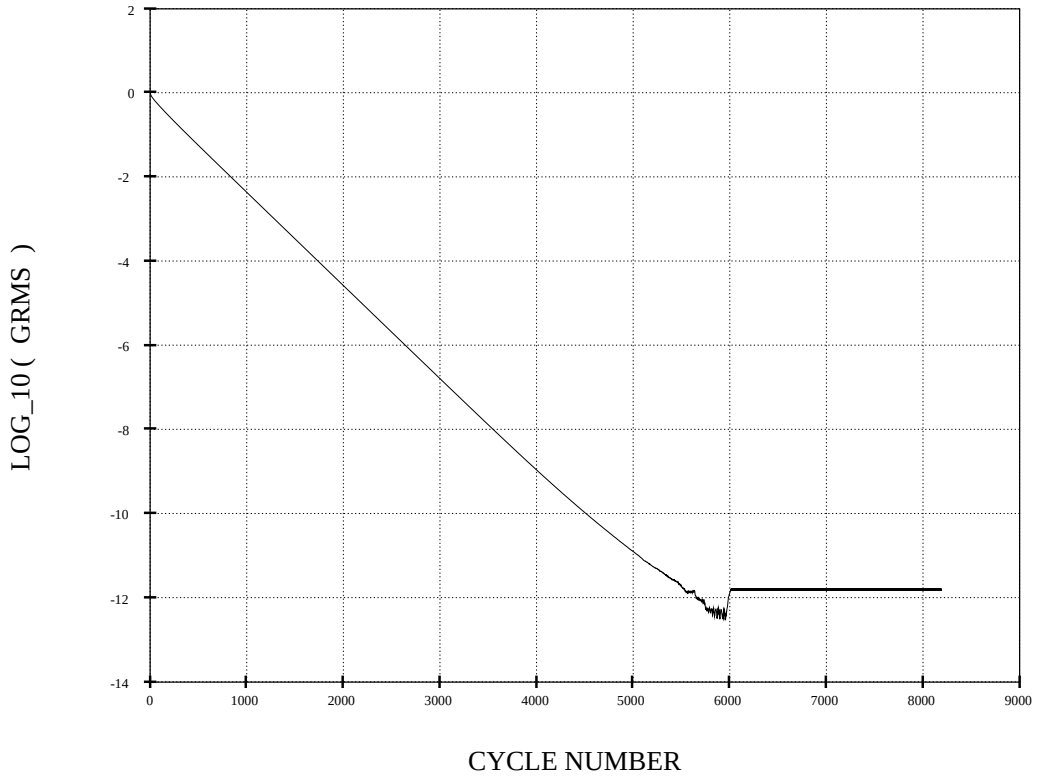


Figure 31: Convergence history of steepest descent with N=31.

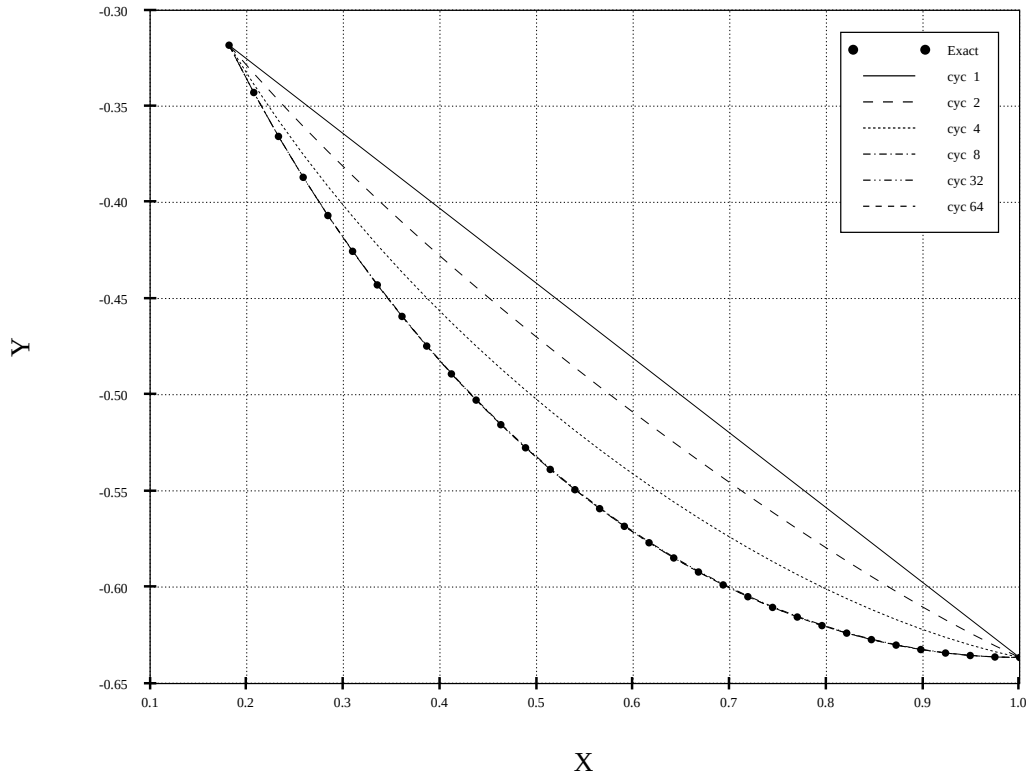


Figure 32: History of paths of smoothed descent with N=31 & STEP=100.

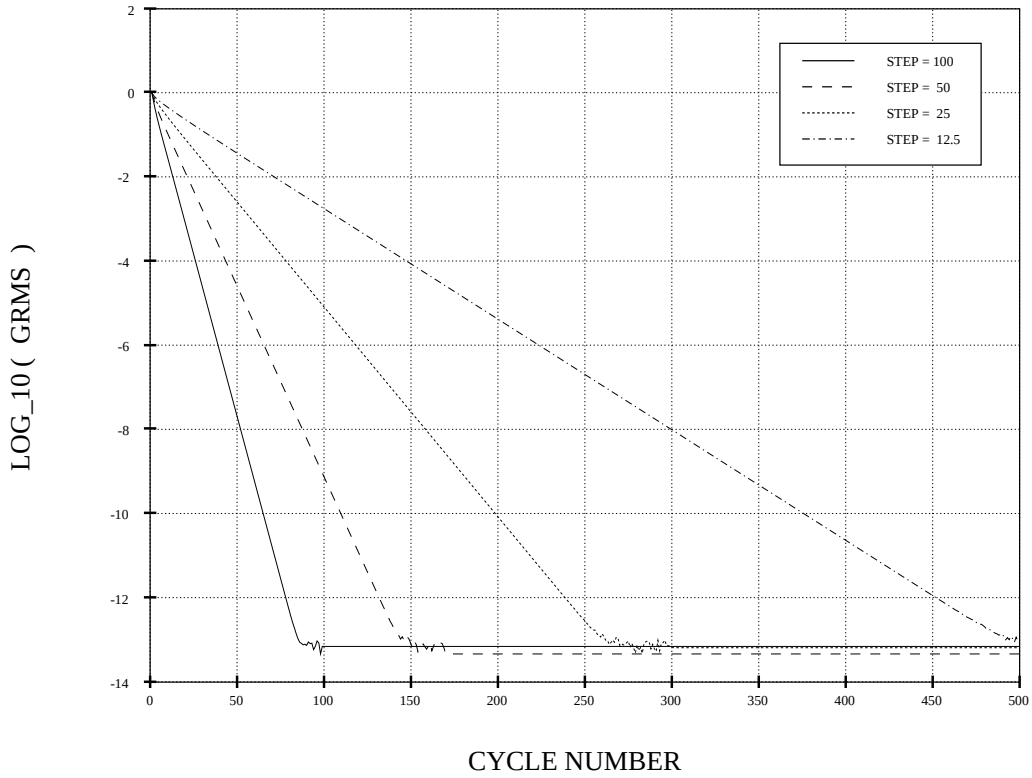


Figure 33: Convergence history of smoothed descent with N=31.

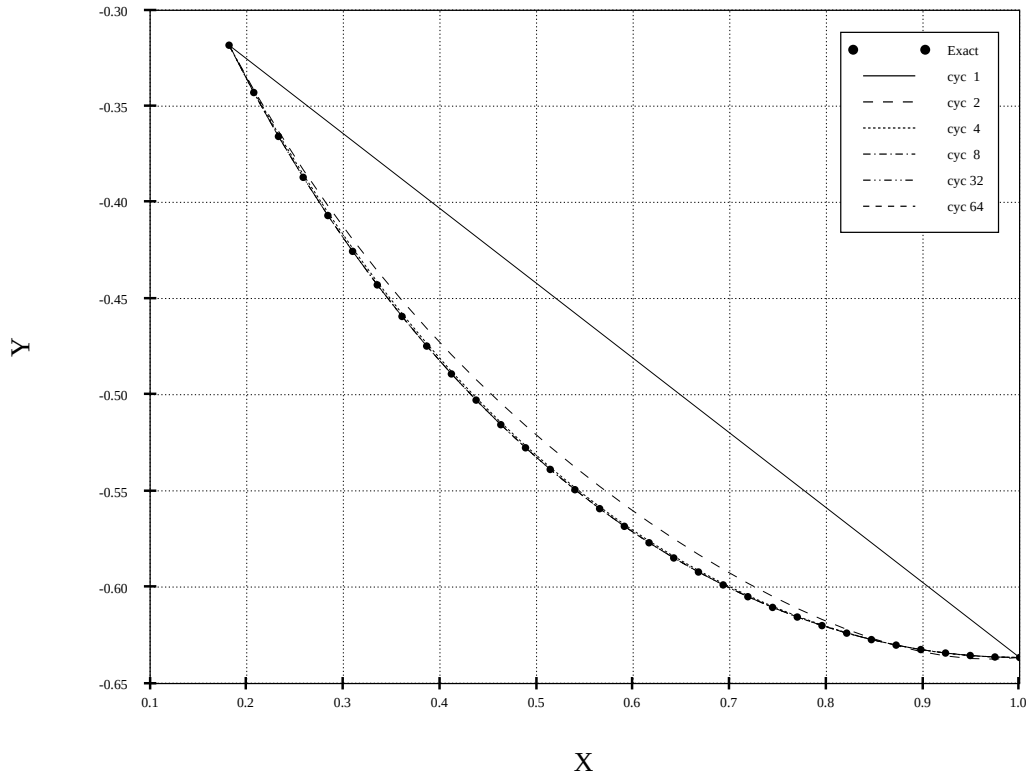


Figure 34: History of paths for Krylov acceleration with N=31 & STEP=100.

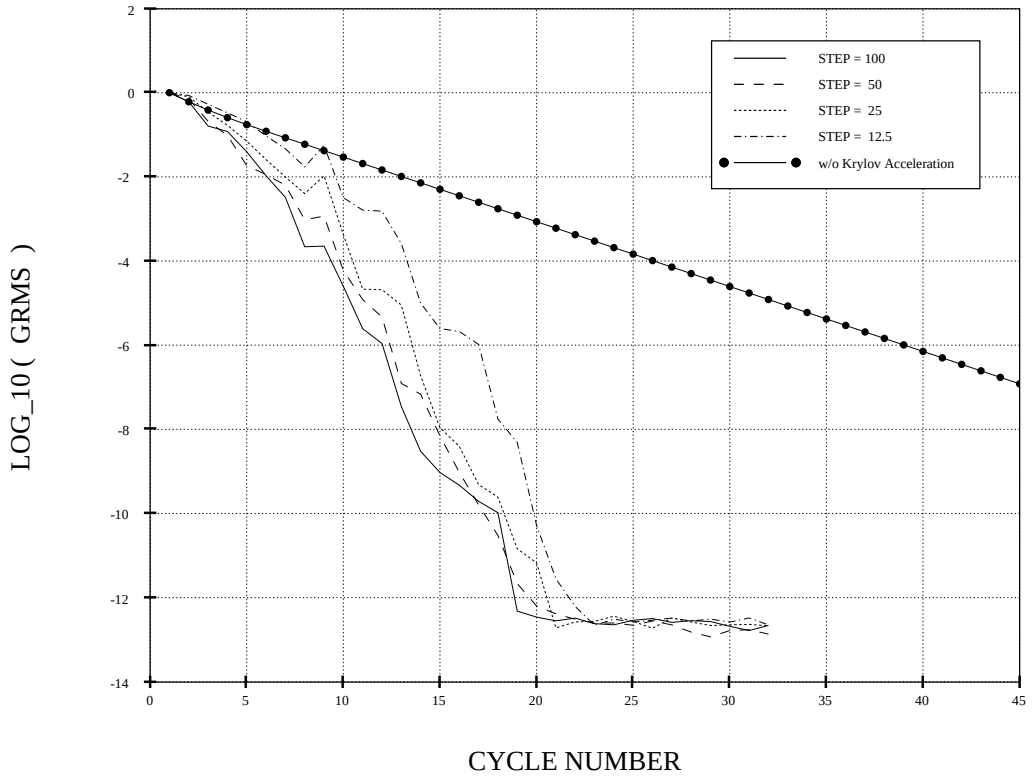


Figure 35: Convergence history of Krylov acceleration with N=31.

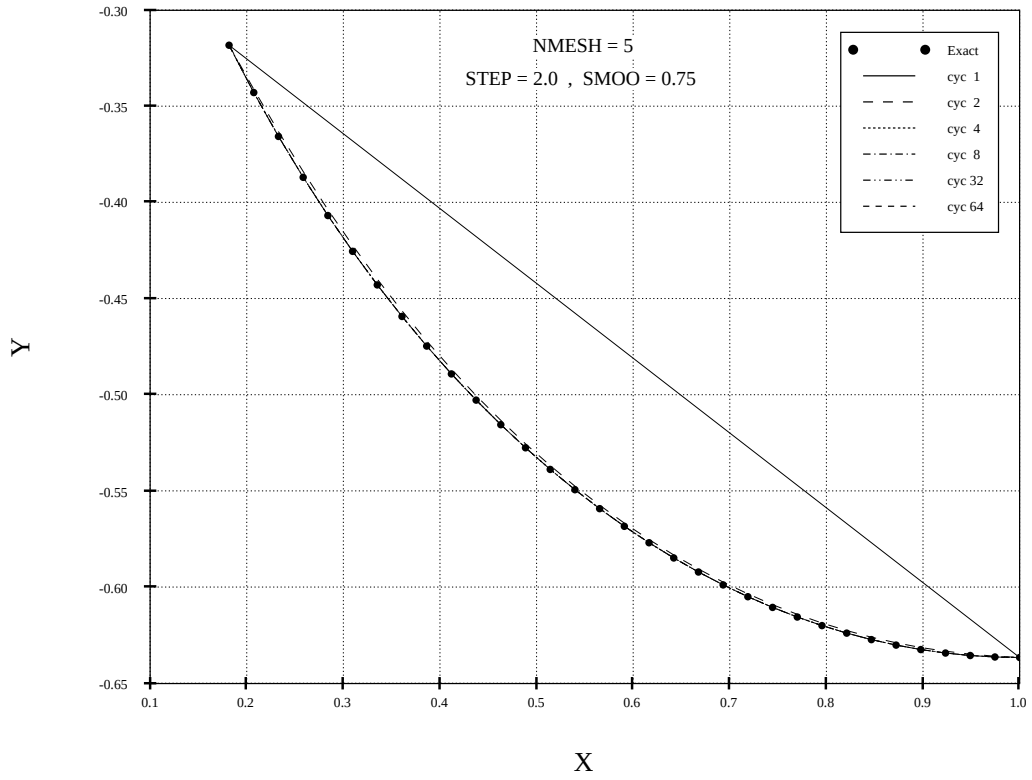


Figure 36: History of paths for multigrid acceleration with N=31.

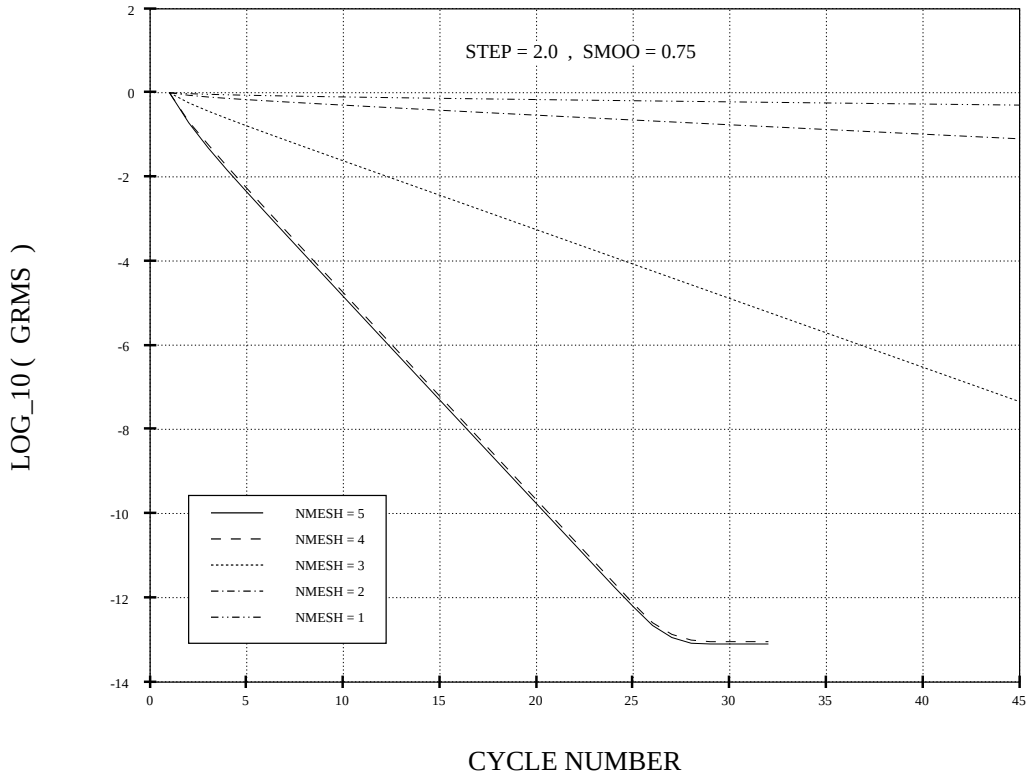


Figure 37: Convergence history of multigrid acceleration with N=31.

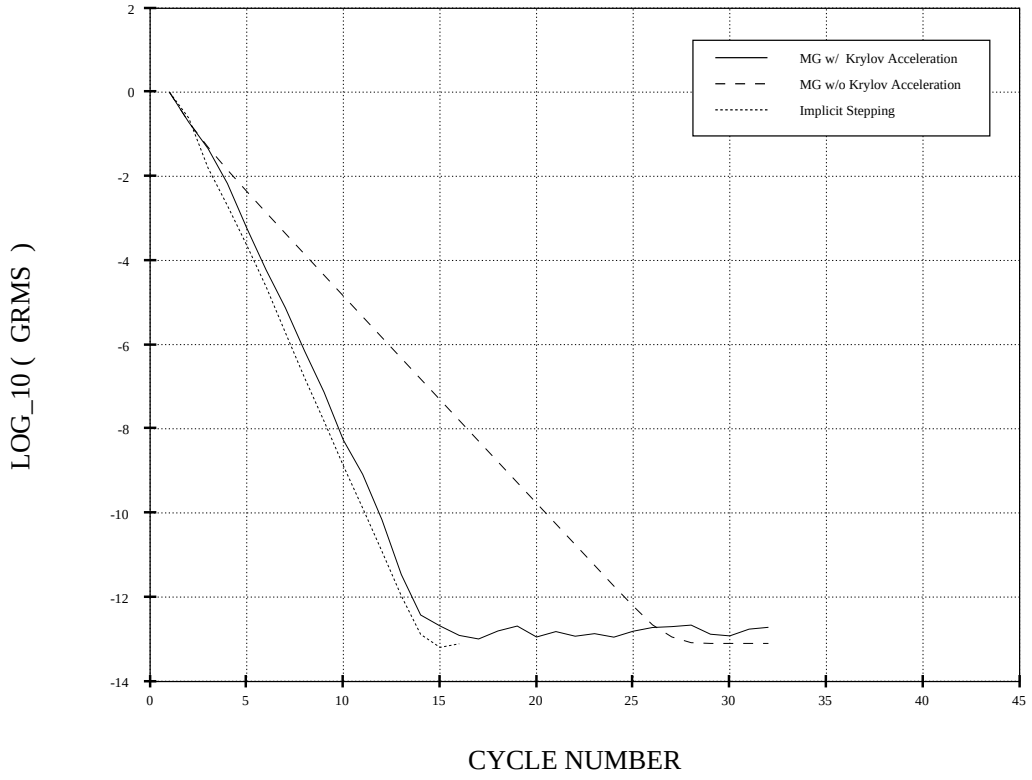


Figure 38: Comparison of grid-independent convergence histories.

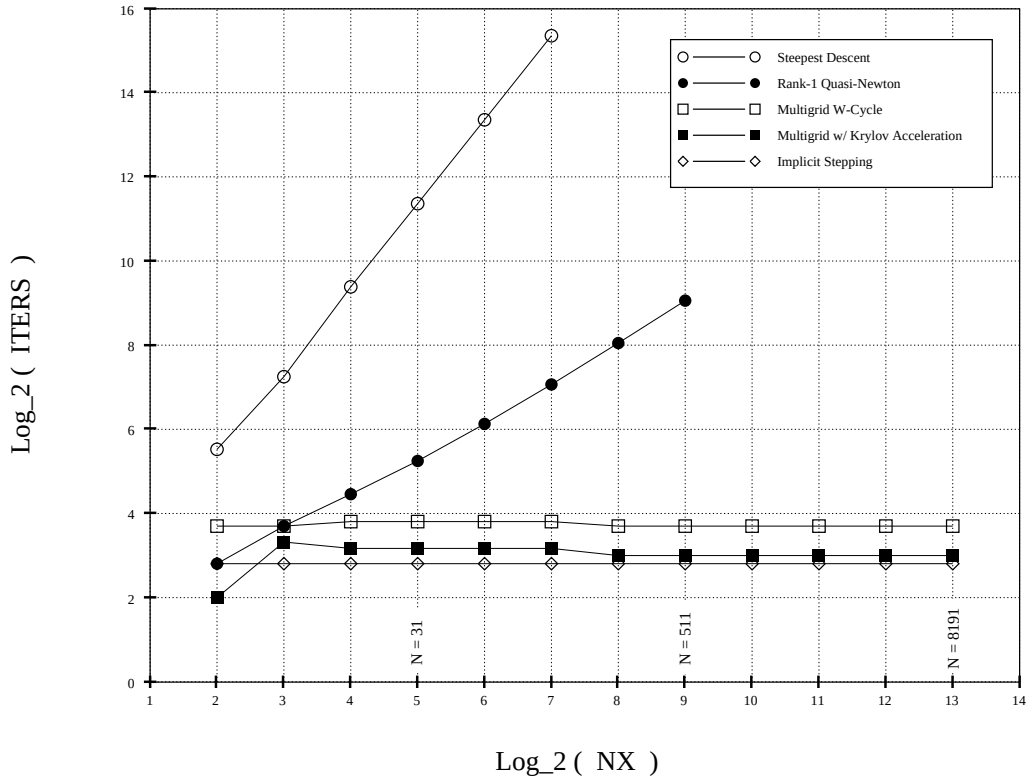


Figure 39: Comparison of convergence properties with dependence on dimensionality.