Minimization with the Affine Normal Direction

Hsiao-Bing Cheng *
Li-Tien Cheng †
Shing-Tung Yau ‡

September 6, 2005

Abstract

In this paper, we consider minimization of a real-valued function f over $\mathbf{R}^{n+1}$ and study the choice of the affine normal of the level set hypersurfaces of f as a direction for minimization. The affine normal vector arises in affine differential geometry when answering the question of what hypersurfaces are invariant under unimodular affine transformations. It can be computed at points of a hypersurface from local geometry or, in an alternate description, centers of gravity of slices. In the case where f is quadratic, the line passing through any chosen point parallel to its affine normal will pass through the critical point of f. We study numerical techniques for calculating affine normal directions of level set surfaces of convex f for minimization algorithms.

1 Introduction

Affine transformations perform translations, rotations, and flips in $\mathbf{R}^{n+1}$, mapping $\vec{x} \mapsto A\vec{x} + \vec{b}$, for $A \in \mathbf{GL}(n+1, \mathbf{R})$, $\vec{b} \in \mathbf{R}^{n+1}$. In digital image processing, they have been used to handle changes in perspective, video images [16, 17], segmentation [1, 15], and smoothing [7], to reference a few. Similar operations also apply to problems in computer graphics and geographic information systems; however, these only scratch the surface of the subject of affine transformations and affine geometry.

The central question of affine differential geometry can be posed as: what hypersurfaces of $\mathbf{R}^{n+1}$ are invariant under the group of unimodular affine transformations? The most basic solutions are what are known as affine hyperspheres, or hypersurfaces whose affine normal lines through each point either intersect at exactly one point or are all parallel. In the simplest examples, these are ellipsoids and hyperboloids. In this paper, we are interested in how this applies to minimization.

*Department of Mathematics, Cornell University, Ithaca, New York 14853
†Deparment of Mathematics, University of California San Diego, La Jolla, California 92093
‡Department of Mathematics, Harvard University, Cambridge, Massachusetts 02138

2 Minimization

Many real-world problems can be stated as optimization problems. In business, one may be interested in minimizing the costs of production, while in biology, one may seek the minimum energy state of a complex molecule. When no physical quantity exists to be minimized, a non-physical quantity can be invented such that desired properties are present at the minimum. For example, in image processing, denoising of an image can be achieved by minimizing total variation in the presence of a fitting term [14]. Thus optimization not only arises from physical problems, but is an established computational tool as well. From this, it is not surprising that optimization has been and currently is a field of intense interest and study.

As a model problem, we consider the following basic optimization problem: find $\vec{x} \in \mathbf{R}^{n+1}$ that minimizes a given smooth, real-valued function $f : \mathbf{R}^{n+1} \to \mathbf{R}$. The function f can be chosen to represent production costs or a physical or artificially constructed energy. Our interest lies in determining the location of minima. Furthermore, we will be content with local minima since global minimization presents difficulties of a highly complex nature and of a different class. Thus, henceforth in this paper, we use the word "minima" to refer to local and global minima.

3 Iterative Methods

Algorithms for minimization typically use iterations to improve on approximations they generate. Starting from an initial guess, these algorithms calculate a direction that approximately points at the minimum. Sometimes both a direction and length is provided, and adding the guess to the vector with that length and direction leads to an improved approximation of the minimum. Otherwise, the improved approximation can be computed from a line search, a search for the minimum of the function along the line passing through the guess parallel to the direction. Such a search is not necessarily computationally expensive due to its one-dimensional nature. Repeating this process starting with the last approximation produces a sequence of approximations. What is desired of this sequence is, of course, fast convergence to the exact minimum.

We outline two example approaches for illustration that, in addition, can later both be viewed as starting points for our new approach to minimization. Variations of them also produce serious minimization algorithms.

3.1 Steepest Descent Method

Situated at any point, a natural direction to choose for minimization is the steepest descent direction. This is the direction along which the function is locally diminishing the most rapidly. Thus the use of this direction leads to a greedy algorithm where the best choice locally is used. Line searches in these directions and iterations produce the sequence of approximations associated to the steepest descent method.

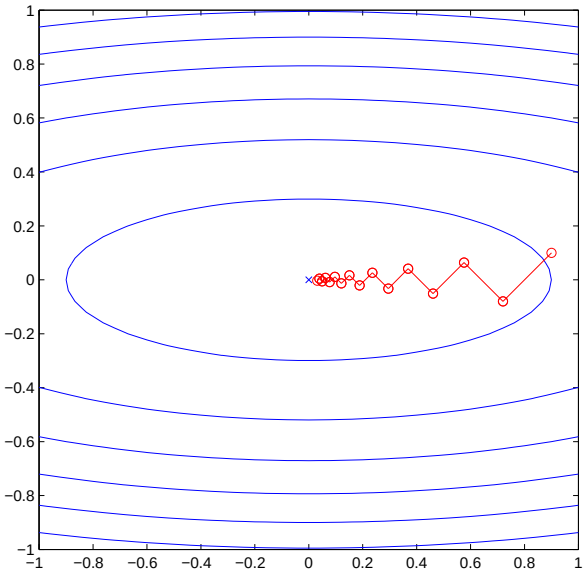

Figure 1: Gradient descent starting at $(0.9, 0.1)$ not yet close after 15 iterations due to a wasteful zig-zag behavior.

The steepest descent direction itself can be computed from derivative information of f through the form: $-\nabla f$. In terms of computational costs, this involves taking first derivatives of the function f. Unfortunately, while this direction is intuitively sound, it exhibits slow convergence for some very basic functions. This has lead to the introduction of the quadratic minimization problem as a basic test case for minimization algorithms. Consider the quadratic function

$$f(\vec{x}) = \vec{x}^t A\vec{x} + \vec{b}^t\vec{x} + c,$$

where A is an $(n+1) \times (n+1)$ positive definite matrix. We note this function has only one critical point, a global minimum. Furthermore, it can be viewed as arising from the quadratic approximation of a more general function. The steepest descent method, in general, performs poorly for quadratic functions (see Figure 1). However, note the one case where it performs perfectly is when $A = kI$, for k a constant. The level sets of f are all circles centered at the minimum and the steepest descent direction $-\nabla f$, which is normal to the level sets, points exactly towards that minimum. This viewpoint on the level sets of the function will fuel our minimization approach. Furthermore, it may be especially useful and natural when level sets of f have special meaning, such as when f represents an image.

We mention here that variations of this approach lead to very competitive and successful minimization algorithms such as the conjugate gradient method. In this method, in addition to the steepest descent direction, conjugate directions are also included. The main points of the algorithm involve efficient construction of the conjugate directions and fast convergence in terms of these sets of directions. Examples of conjugate gradient methods include the Fletcher-Reeves and Polak-Ribiere methods. See [8, 12] for more on the subject.

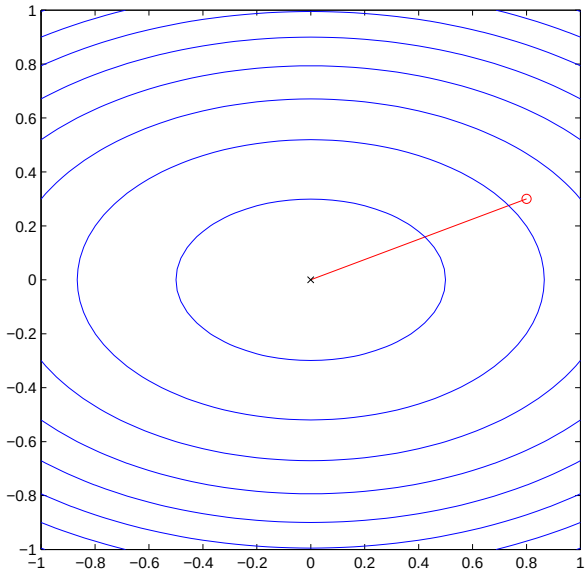

Figure 2: Finding the minimum of the quadratic minimization problem in one iteration starting from $(0.8, 0.3)$.

3.2 Newton's Method

Consider the quadratic approximation of the function f at the origin:

$$f(\vec{x}) \approx \vec{x}^t \nabla^2 f(0)\vec{x} + \nabla f(0)^t \vec{x} + c.$$

The critical point of this quadratic satisfies

$$\nabla^2 f(0)\vec{x} + \nabla f(0) = 0,$$

and so is located at $\vec{x} = -(\nabla^2 f(0))^{-1}\nabla f(0)$. Generalizing, when the quadratic approximation is taken at a point $\vec{y}$, the critical point is at $\vec{x} = \vec{y} - (\nabla^2 f(\vec{y}))^{-1}\nabla f(\vec{y})$. Thus, given an initial guess, this critical point, which corrects the guess by $-(\nabla^2 f)^{-1}\nabla f$, evaluated at the guess, can serve as the next approximation in an iterative method. This is known as Newton's method, whose sequence of approximations exhibit fast convergence. In terms of computational costs, this approach involves calculating first and second derivatives and inverting the $(n+1) \times (n+1)$ Hessian matrix. However, it performs perfectly on our test case, the quadratic minimization problem, with the vector $-(\nabla^2 f)^{-1}\nabla f$ sending any initial guess straight to the minimum (see Figure 2). Furthermore, it is able to capture maxima and unstable critical points as well as minima.

Improvements on Newton's method focus on improvements in efficiency, leading to competitive and widely used minimization algorithms. The so-called quasi-Newton methods remove calculation of second derivatives of f by replacing the inverse of the Hessian with an approximation that doesn't involve them. Examples of quasi-Newton methods include the Davidon-Fletcher-Powell and Broyden-Fletcher-Goldfarb-Shanno methods. See [8, 12] for more on the subject.

4 Affine Normal

The algorithm we introduce can be seen as incorporating the affine normal direction into the steepest descent method or, alternatively, Newton's method. To understand the affine normal direction, we look at affine differential geometry. Let us first consider the Euclidean case, using the Euclidean metric on $\mathbf{R}^{n+1}$. Let M be a hypersurface in $\mathbf{R}^{n+1}$. We can then choose a vector field of normal vectors N to M. Now if X and Y are vector fields on M and $D_X Y$ is the flat connection on $\mathbf{R}^{n+1}$, then we can decompose

$$D_X Y = \nabla_X Y + h(X, Y)N,$$

where $\nabla_X Y$ is the tangential part of $D_X Y$ and $h(X, Y)$ the normal part, also known as the second fundamental form. Furthermore, in this case, $\nabla_X Y$ is the Levi-Civita connection of the Riemannian metric induced by $\mathbf{R}^{n+1}$ on M.

In the affine situation, we consider any transversal vector field ξ on M and define $\nabla_X Y$ and $h(X, Y)$ as above, but with ξ in place of N. M is called non-degenerate if h has rank n, and if this is true, then there is a unique ξ, up to sign, so that the induced volume element $\theta(x_1, \ldots, x_n) = \det(x_1, \ldots, x_n, \xi)$ is parallel relative to the induced connection ∇. Moreover, it is equal to the volume element of the metric h. This ξ is called the affine normal field and II the affine metric. Note ξ is invariant under the unimodular affine group.

Alternatively, if we choose an arbitrary local frame field $e_1, \ldots, e_{n+1}$ with $e_1, \ldots, e_n$ tangent to M and $\det(e_1, \ldots, e_{n+1}) = 1$, we may define h as above, with $D_X Y = \nabla_X Y + h(X, Y)e_{n+1}$, to arrive at the affine metric

$$II_{i,k} = H^{-\frac{1}{n+2}} h_{i,k},$$

where H is mean curvature and the indices are in the $\{e_k\}$. In this case, it can be proven that the affine normal field is given by ΔX, where the laplacian is with respect to the affine metric II and X is the position vector of M.

When M is a level set surface of the function f, we can derive the affine normal field to take the form

$$H^{\frac{1}{n+2}} \begin{pmatrix} f^{ij}\left(-\frac{n}{n+2} f^{pq} f_{pqi} + n\frac{f_{n+1,i}}{|\nabla f|}\right) \\ -\frac{n}{|\nabla f|} \end{pmatrix},$$

where the coordinates x_i used are rotated so that x_{n+1} is in the normal direction; i.e., in the direction of ∇f. If only the direction of the affine normal is of importance we may simplify this expression, such as dropping the mean curvature H, to arrive at

$$\begin{pmatrix} f^{ij}\left(-\frac{1}{n+2}|\nabla f| f^{pq} f_{pqi} + f_{n+1,i}\right) \\ -1 \end{pmatrix}.$$

Rotating back to standard coordinates just involves treating each of the components in the above vector as the corresponding coefficient of a linear combination of the x_k in standard coordinates. This gives the affine normal direction of the level set surface of f passing through a point of interest.

We note here some details of the affine normal vector. One is that the length of the vector itself seems to be meaningless in terms of minimization. Another is that the vector degenerates when the Hessian is non-invertible. Finally, as mentioned before, when the hypersurface is an ellipsoid, all affine normals point towards its center. In fact, the affine normals of the level sets of a quadratic polynomial will point toward the unique critical point, even if that critical point is unstable. We will not take advantage of this latter property in our work here. See [2, 3, 4, 6, 11] for more on the affine normal and affine differential geometry.

5 Affine Normal Descent Algorithm

Using this affine normal field, we can summarize our algorithm in the following steps, iterated to convergence:

1. Compute the affine normal direction to the level set of the function at the current approximation location.

2. Use a line search to find the minimum of the function along that direction. This location serves as the new approximation.

We call this the affine normal descent algorithm.

For the quadratic minimization problem, due to the nature of the affine normal and the ellipsoidal level sets of f, the approximations of this algorithm will take on the value of the exact minimum after one iteration (see Figure 3). Thus, the affine normal and the vector $-(\nabla^2 f)^{-1}\nabla f$ used in Newton's method are parallel to each other in this case, though, in general, not for other choices of f.

This means we may view this algorithm as an extension of the steepest descent method, using the affine normal direction, which points at the center of ellipsoids, instead of the steepest descent direction, which points at the center of spheres. On the other hand, we may view it as a relative of Newton's method, both exact for the quadratic minimization problem but with different higher order terms.

Use of the affine normal has an immediate consequence: the algorithm is affine scale invariant. This is a useful property that implies the existence of rigorous testing programs that are easy to run and interpret [10] and can even mean greater numerical stability in certain circumstances [13]. Furthermore it is particularly adapted to scale-space [5] and natural scenes [1] in two and three-dimensional image processing and analysis. Another useful property is that the affine normal depends just on the shape of the level sets of f, not their values. This implies the algorithm is invariant under contrast changes, borrowing from terminology used in image processing.

What we have listed so far are certainly advantages, however, in order for the affine normal descent algorithm to truly be of merit, we must illuminate two other major aspects. One concerns the computational costs of the approach, especially in computing the affine normal direction. The other involves the generality of its application, namely to problems outside of the class of quadratic minimization problems. For the former issue, we consider

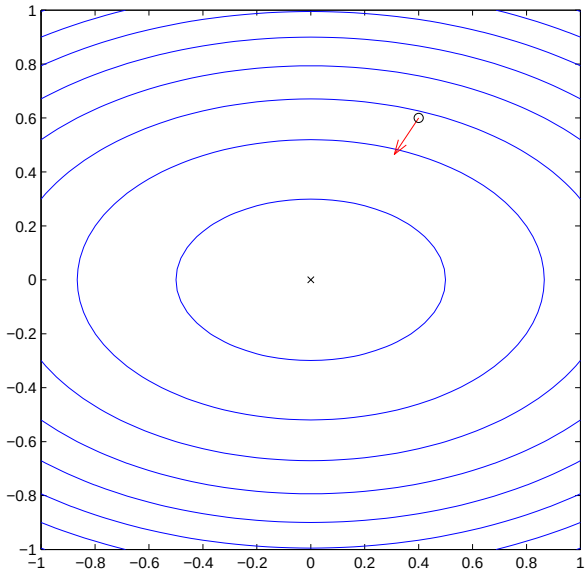

Figure 3: In this two-dimensional example, we illustrate the fact that the affine normal points toward the minimum of quadratic functions with ellipsoidal level sets.

the computational costs of the geometric construction of the affine normal direction and propose alternatives arising from different viewpoints afforded by affine geometry. For the latter issue, we perform numerical experiments on the minimization of a wide range of convex functions f and investigate the results.

6 Efficiency

In terms of computational costs, we note that the previously derived formula for the affine normal direction requires first, second, and third derivatives of f, as well as inversion of an $n \times n$ matrix of second derivatives. While it may be possible to generate other forms or approximations of the affine normal direction that simplify the inversion or diminish the need of derivative information, perhaps following how the quasi-Newton methods simplify Newton's method, we are currently not able to derive the correct expressions.

Instead, we consider a different viewpoint of the affine normal to bypass the need for such information. Consider a convex hypersurface, a point on that surface, and the tangent plane located there. Furthermore, consider the class of planes intersecting with the surface and parallel to the tangent plane. On each of these planes, we look at the center of gravity of the region enclosed by the intersection of the plane with the surface. The union of these centers of gravity forms a curve. It turns out that the one-sided tangent direction of this curve at the point of interest is the affine normal vector (see Figure 4). Thus, an alternate approach for calculating the affine normal vector involves calculating centers of gravity, completely bypassing the need for derivative information higher than that of the first derivative which is required for tangent planes. More accurately, it is not actually the centers of gravity that

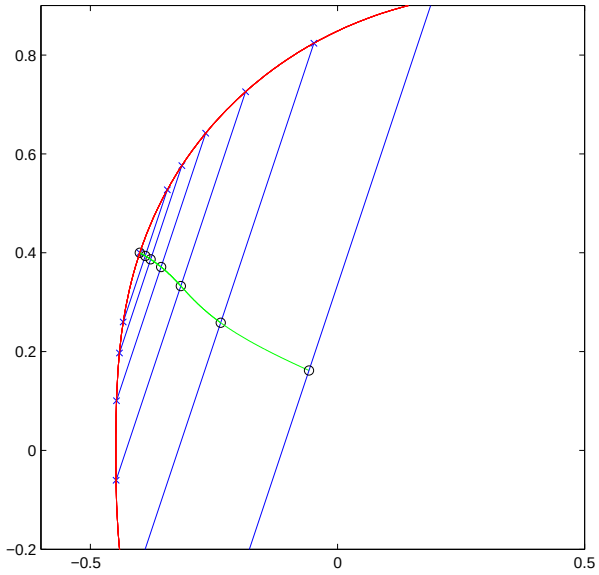

Figure 4: In this two-dimensional example, we show a point lying on the surface of a convex object, lines parallel to the tangent line at that point, the centers of gravity of the part of those lines lying in the convex object, and the curve that is created by a union of those centers of gravity. The tangent of that curve at the point of interest on the surface gives the affine normal direction.

are needed, just the one-sided tangent of the curve formed by them.

In detail, at an approximation location p, to find the equation of the plane passing through that location and tangent to the level set surface $f(x) = f(p)$, we note that ∇f is normal to that surface. Thus the plane satisfies $P(x) = \nabla f \cdot (x - p) = 0$ and planes parallel to it satisfy $P(x) = C$, for $C \in \mathbf{R}$. The regions of interest on these planes can then be implicitly identified as $\{P(x) = C\} \cap \{f(x) \leq f(p)\}$ and algorithms for finding centers of gravity of convex objects can be brought to bear on this description. Thus, we can calculate the center of gravity of such a region for a small value of $C < 0$ to form a divided difference approximating the tangent of the curve of centers of gravity. This gives an approximation of the affine normal vector.

This approach to an efficient algorithm seems promising, though it must be pointed out that current methods for finding centers of gravity are not fast enough in high dimensions to achieve a speed-up for our approach (see [9] for related but more difficult volume techniques). It is certainly beyond the scope of this paper to produce such methods. However, we fully expect such methods to be advanced in the near future.

Figure 5 visually compares two-dimensional minimizations of the function

$$f(x) = 2(x_1^2/2 + x_1^4/12) + (x_2^2/2 + x_2^4/12) + \sin x_1 \sin x_2 - 1$$

using derivatives, on the one hand, and centers of gravity, on the other, to calculate the affine normal. The results are identical. We note, however, that the method employed here

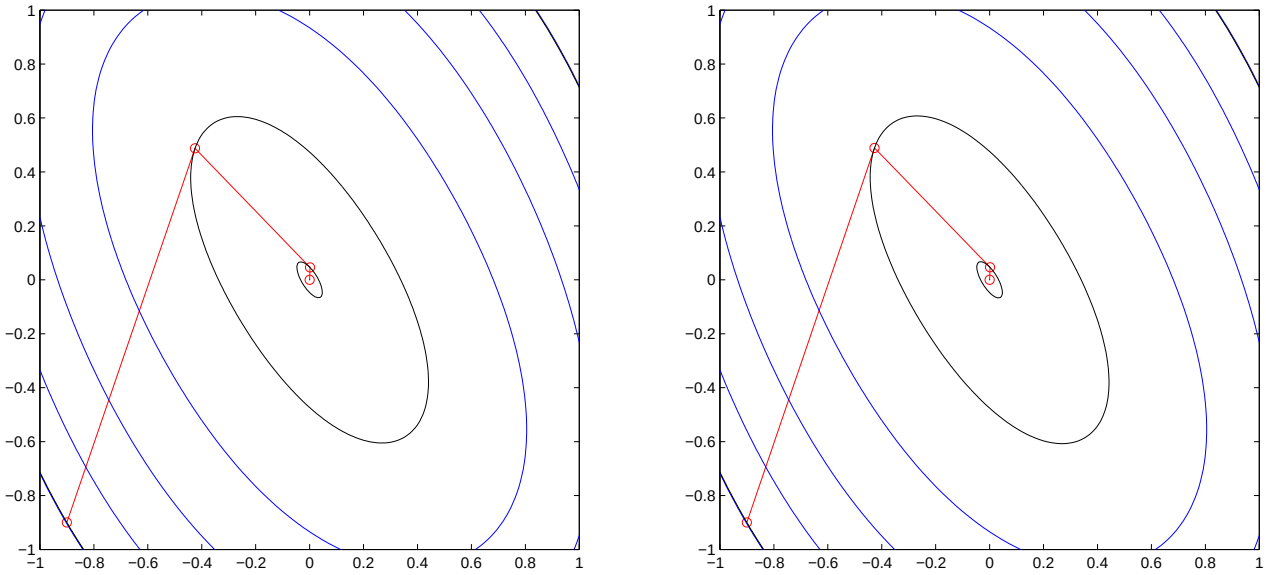

Figure 5: The figure on the left shows a numerical result of minimization using the affine normal computed from centers of gravity. The figure is identical to the one shown on the right, where the affine normal is computed through the geometric definition that uses derivatives.

for computing the centers of gravity is dimension-specific and not generalizable for efficient high-dimensional computations.

Additionally, there is another viewpoint for the construction of centers of gravity of convex objects. Consider the ellipsoid of minimum volume that circumscribes the convex object. The center of this ellipsoid is the center of gravity of the object and, in fact, dilation by $1/(n+1)$, where $n+1$ is the dimension of the object's ambient space, using origin at the ellipsoid center leads to a smaller ellipsoid that lies completely inside the object. Though there are currently no fast methods for finding any of these ellipsoids, we state its connection with the affine normal to illustrate other possibilities for affine normal construction in high dimensions. For more on these alternative viewpoints of the affine normal, see [2, 3, 4, 6, 11].

7 Convex Minimization

For more general cases, consider a convex f where the Hessian is always positive definite. Also suppose that the function's critical point, the global minimum, is attained away from infinity. Note, all level sets of f, which are what is important in our algorithm, are convex, and the affine normals of these surfaces will not, in general, point directly at the minimum. This means our affine normal descent algorithm in general generates an infinite sequence of approximations. We investigate the speed of convergence and approximation errors of this sequence by performing a series of numerical tests. Since the iterations of our algorithm are more similar to the quasi-Newton methods than the conjugate graidient methods, and

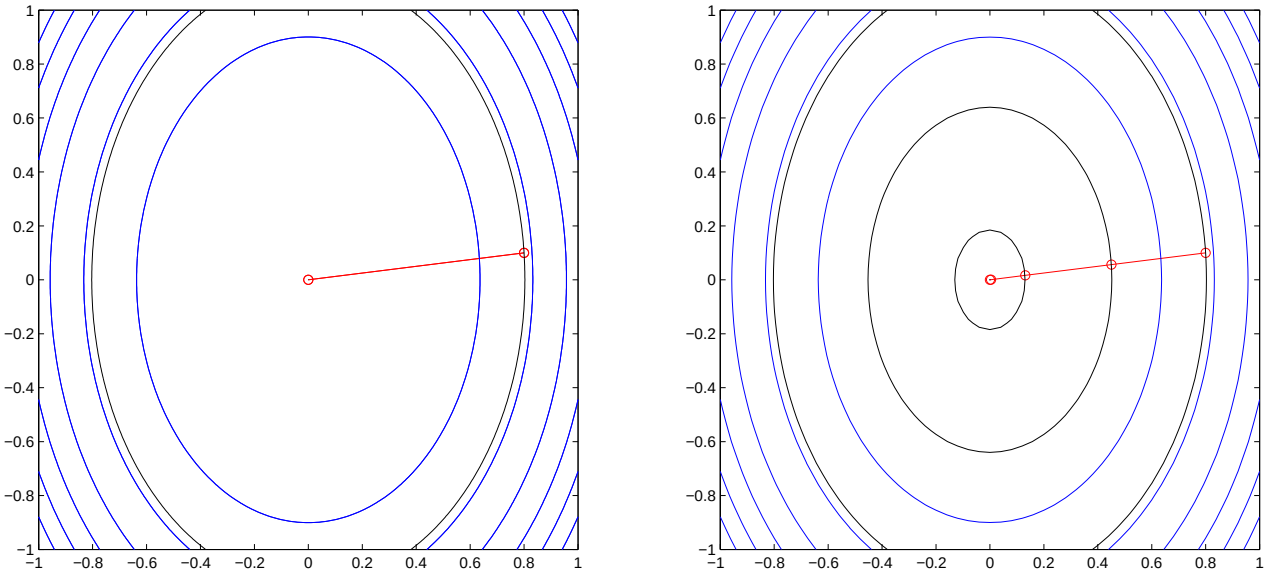

Figure 6: On the left, our algorithm exhibits convergence in one step to the minimum of a non-polynomial function whose level sets are ellipses. On the right, Newton's method requires 3 to get close to the minimum and 4 to reach a comparable accuracy.

since quasi-Newton methods are simply efficient forms of Newton's method, we mainly make comparisons of our results with those of Newton's method.

Our first series of tests considers convex functions whose level sets are ellipses centered at the minimum, such as

$$f(\vec{x}) = \exp\left(\sum_{i=1}^{n+1} \frac{x_i^2}{i}\right).$$

Note such a function is not a quadratic polynomial, but is a contrast change of one. Our algorithm converges in one step to the minimum in every case and in every dimension, as expected. Figure 6 shows the convergence in two dimensions along with that of Newton's method, which requires up to 4 iterations. We note, since it is hard to determine from the figure, that the direction of minimization that Newton's method uses in general does not point exactly at the minimum.

Our second series of tests takes place in three dimensions and studies the convergence of the affine normal descent algorithm. Let

$$f(\vec{x}) = 3\sum_{i=1}^{3} x_i^2 + \sum_{i=1}^{2} \sin x_i \sin x_{i+1}$$

be the convex function whose minimum p is at the origin. Table 1 collects information on iterations p_j of our algorithm with initial guess at $p_0 = (2, -1, -2)$. Convergence is fast and after 4 iterations, the minimum is captured. Newton's method exhibits similar speed in this case.

j	$f(p_j)$	$\lvert p_j - p_{j-1}\rvert$	$\lvert p_j - p\rvert$
0	27		3
1	1.40210926439103	2.87339142153736	0.69783378511265
2	0.00043034937731	0.70047441690861	0.01251645103971
3	0.00000000000001	0.01251643844478	0.00000005727856
4	0.00000000000000	0.00000005727852	0.00000000000005

Table 1: **Three Dimesional Result**: In this three-dimensional example, the minimum is captured after 4 iterations.

j	$f(p_j)$	$\lvert p_j - p_{j-1}\rvert$	$\lvert p_j - p\rvert$
0	1327.29166767858		13.60147050874
1	107.61710514695	13.34645179930	6.27181862814
2	100.05967077939	2.29760334273	6.30056266636
3	49.45367585162	4.43737133243	5.20714147442
4	5.57628342138	5.22989032262	1.91111608274
5	3.39959363498	1.76639408750	0.85090933891
6	3.01528632271	0.83300818402	0.17461685512
7	3.00000027963	0.17461525378	0.00074783310
8	3.00000000000	0.00074783310	0.00000000012

Table 2: **Starting Far in Three Dimensions**: Starting far from the minimum, convergence is at first slow, then gains speed as the approximation gets closer to the minimum.

Alternatively, let

$$f(\vec{x}) = \sum_{i=1}^{3} \left(x_i^2 + \frac{x_i^4}{12} + \cos x_i \right)$$

be the convex function whose minimum p is at the origin. In this case, we choose a starting point that is further from the origin, $p_0 = (10, 6, 7)$. Table 2 collects information on iterations p_j of our algorithm. As expected, the iterations are slow to converge at first, due to the distance of the approximations away from the minimum. However, as the approximations get better, so does the speed of convergence. The minimum is captured after 8 iterations. These examples are representative of the affine normal descent algorithm in three dimensions. Newton's method has similar convergence speeds in these tests.

Testing higher dimensions, let

$$f(\vec{x}) = 6 \sum_{i=1}^{6} x_i^2 + \sum_{i=1}^{5} \sin x_i \sin x_{i+1}$$

be the convex function with $\vec{x} \in \mathbf{R}^6$ and minimum at the origin. This is a higher-dimensional version of one of our previous three-dimensional tests. Using the initial guess $(0.1, -2, 0.2, 0, -0.3, 0.8)$, we show the iterations of our algorithm in Table 3. The behavior in terms of convergence is the same as in the three-dimensional case, with convergence occurring after 3 iterations. Newton's method exhibits similar speed for these tests.

For a five-dimensional convex function, let

$$f(\vec{x}) = \sum_{i=1}^{5}(x_i^2 + \sin x_i).$$

Let $(-0.2, 0, 0.4, 1, -0.3)$ be the starting point. The iterations of our algorithm are shown in Table 4. Note, we do not assume the minimum to be known in this case. Convergence to the point

$$\begin{pmatrix} -0.45018354967147 \\ -0.45018354967140 \\ -0.45018354967139 \\ -0.45018354967125 \\ -0.45018354967129 \end{pmatrix}$$

is achieved in a few iterations. Newton's method requires a similar number of iterations for convergence.

For a more nonlinear example, consider the function

$$f(\vec{x}) = \exp\left(\sum_{i=1}^{5} \frac{x_i^2}{i} + \frac{x_i^4}{12}\right) + \frac{1}{10}\sum_{i=1}^{5} \cos x$$

with minimum at the origin. Table 5 shows our affine normal descent algorithm in action with starting point $(0.2, 0.2, -0.4, 0.1, 1)$. In this example, convergence was achieved in 4 iterations while Newton's method required 6. This gap is retained even for other starting points and on other similar functions. This leads us to advance the notion that our algorithm performs better for highly nonlinear functions.

For ten dimensions, with the convex function

$$f(\vec{x}) = \sum_{i=1}^{10}\left(\frac{x_i^2}{i} + \frac{ix_i^4}{120}\right),$$

we once again see fast convergence to the minimum located at the origin in Table 6 using the initial guess $(1, 1, 1, 1, 1, 1, 1, 1, 1, 1)$. These examples are representative of the affine normal descent algorithm in higher dimensions. Newton's method, in this case, performed just as fast.

From these studies in higher dimensions, we see the convergence of our approach is not strongly affected by the dimension involved. Newton's method also has this property. Many other minimization algorithms such as the steepest descent method, however, can require significantly more work for convergence in higher dimensions.

For a majority of our tests so far, Newton's method has performed as well as our approach, though in general leading to different sequences of approximations. However, consider the convex function

$$f(\vec{x}) = \sum_{i=1}^{n+1}\left(\frac{x_i^2}{i} + \frac{1 - \cos(ix_i)}{i^3}\right)$$

with minimum at the origin. Our algorithm converges on average in 3 iterations for a variety of starting points, with one case shown in Table 7. Newton's method, on the other

j	$f(p_j)$	$\lvert p_j - p_{j-1}\rvert$	$\lvert p_j - p\rvert$
0	28.19657899961829		2.18632111989754
1	0.10820380719931	2.17147295853185	0.13959871710538
2	0.00000000732600	0.13960261665982	0.00003483822916
3	0.00000000000000	0.00003483822789	0.00000000000152

Table 3: **Six-Dimensional Result**: In this six-dimensional example, the minimum is captured after 3 iterations.

j	$f(p_j)$	$\lvert p_j - p_{j-1}\rvert$
0	2.02669978966015	
1	−0.87543513107826	2.17147295853185
2	−1.16211480429203	0.13960261665982
3	−1.16232787552543	0.00003483822789
4	−1.16232787579106	0.00001466875411
5	−1.16232787579106	0.00000000001789

Table 4: **Five-Dimensional Result**: In this five-dimensional example, convergence is achieved after 5 iterations.

j	$f(p_j)$	$\lvert p_j - p_{j-1}\rvert$	$\lvert p_j - p\rvert$
0	1.93582623492383		1.11803398874989
1	1.53696009070222	0.91576389258490	0.43531524358145
2	1.50052639212305	0.41811574484086	0.05228924261910
3	1.50000000048994	0.05227649798143	0.00005180928984
4	1.50000000000000	0.00005180928710	0.00000000000906

Table 5: **Highly Nonlinear Result in Five Dimensions**: In this five-dimensional example on a highly nonlinear function, convergence is achieved after 4 iterations.

j	$f(p_j)$	$\lvert p_j - p_{j-1}\rvert$	$\lvert p_j - p\rvert$
0	3.38730158730159		3.16227709208757
1	0.20345967861706	3.57024309129289	0.78295790802333
2	0.01001074240275	0.91450443097023	0.24957535992445
3	0.00001004198928	0.24673975195323	0.00934204949602
4	0.00000000000009	0.00934204949602	0.00000000000000

Table 6: **Ten-Dimensional Result**: In this ten-dimensional example, the minimum is captured after 4 iterations.

j	$f(p_j)$	$\lvert p_j - p_{j-1}\rvert$	$\lvert p_j - p\rvert$
0	1.21871862457614		1.445688322948010
1	0.04157512870688	1.49053754544770	0.24901763711108
2	0.00000037826466	0.24918409356303	0.00078674976823
3	0.00000000000000	0.00078674979188	0.00000000011379

Table 7: **Special Example in Three Dimensions**: Three-dimensional example converging in 3 iterations with initial guess at $(0.3, -1, 1)$. Newton's method, on the other hand, does not converge.

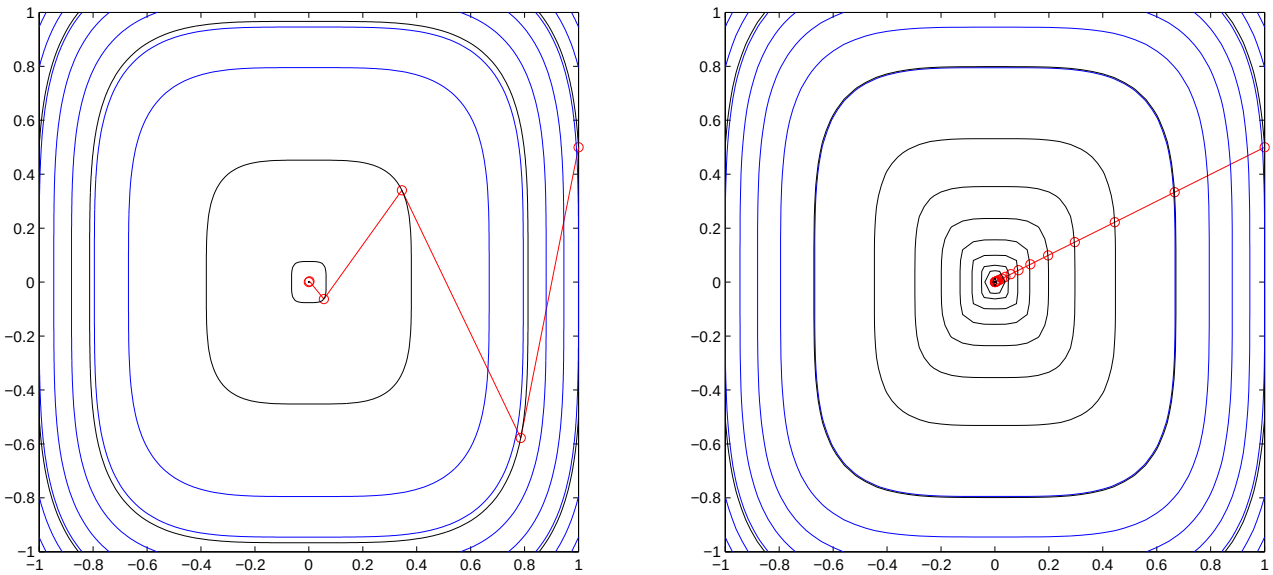

Figure 7: Our algorithm, on the left, and Newton's method, on the right, converge slowly and differently to the minimum at the origin.

hand, in general does not converge, oscillating for example between $(0, 0, -1.047191267)$ and $(0, 0, 1.047191267)$ in the three-dimensional case with initial guess $(0, 0, 1)$.

Finally, we consider a degenerate case. Let

$$f(\vec{x}) = \sum_{i=1}^{2} \frac{x_i^4}{i}.$$

In this case, convergence is slow near the minimum at the origin, due to a singular Hessian there, among other places. Starting from $(1, 0.5)$, our algorithm after 20 iterations reaches $(0.00001514822871, -0.00002512372278)$. Newton's method after the same number of iterations reaches $(0.00015233661514, 0.00004721666799)$. However, their behaviors are different, as seen in Figure 7. Nevertheless, degenerate cases should be avoided.

From these studies, we see that our affine normal descent algorithm exhibits fast convergence in a variety of situations in a number of dimensions. In fact, higher dimensions do not greatly affect the number of iterations needed. The speed of convergence in general is

on the scale of that of Newton's method, meaning our algorithm can be competitive with other minimization algorithms that are around. In addition, it is especially fast for certain, and perhaps most, highly nonlinear functions. On the other hand, as expected, convergence is slower when far from the minimum or when the function is degenerate at the minimum. These cases should be avoided as much as possible. Finally, we would like to note that the studies we have carried out here are mainly numerical in nature. Analytical studies to verify our results need to be the subject of future work.

8 Conclusion

Summarizing, our studies reveal three advantageous characteristics of our affine normal descent algorithm. The first is affine scale invariance, which can be at the same time natural, simplifying, and essential. The second is that it is exact for quadratic minimization due to the direction the affine normal points. Numerical studies show this translates to fast convergence when more general convex functions in any dimension are considered, especially a class of highly nonlinear ones. The third is that alternate viewpoints for constructing the affine normal allow for efficiency. Though we are unable currently to take advantage of this due property, we believe the desired advances will be forthcoming in the future. Altogether, we conclude that use of the affine normal as a descent direction in minimization, with the abilities revealed in our studies, can make an impact as a new algorithm in the field of minimization.

9 Acknowledgements

The first author was partially supported by an NSF VIGRE Postdoctoral Fellowship. The second author was partially supported by NSF grant #0208449.

References

[1] C. Ballester, V. Caselles, and M. Gonzalez. Affine Invariant Segmentation by Variational Method. *SIAM J. Appl. Math.*, 56(1):294–325, 1996.

[2] W. Blaschke. *Vorlesungen über Differentialgeometrie II, Affine Differentialgeometrie*. Springer-Verlag, 1923.

[3] E. Calabi. Complete Affine Hypersurfaces I. *Instituto Nazionale di Alta Matematica Symposia Mathematica*, 10:19–38, 1972.

[4] E. Calabi. Géometrie Différentielle Affine des Hypersurfaces. *Séminaire Bourbaki*, 573:1–16, 1980/81.

[5] V. Caselles and C. Sbert. What is the Best Causal Scale-Space for 3D Images. Tech Report, Univ. of Illes Balears, 1994.

[6] S.Y. Cheng and S.T. Yau. Complete Affine Hypersurfaces. Part I. The Completeness of Affine Metrics. *Comm. Pure Appl. Math.*, 39:839–866, 1986.

[7] M. Descoteaux. Affine and Euclidean Geometric Heat Equation for Anisotropic Smoothing. Tech Report, McGill University, 2003.

[8] D.A.H. Jacobs. *The State of the Art in Numerical Analysis*. Academic Press, 1977.

[9] L. Lovász and S. Vempala. Simulated Annealing in Convex Bodies and an $\mathcal{O}^*(n^4)$ Volume Algorithm. *Proc. of the 44th IEEE Foundations of Computer Science*, page 650, 2003.

[10] J.N. Lyness. The Affine Scale Invariance of Minimization Algorithms. *Math. Comp.*, 33:265–287, 1979.

[11] K. Nomizu and T. Sasaki. *Affine Differential Geometry: Geometry of Affine Immersions*. Cambridge University Press, 1994.

[12] E. Polak. *Computational Methods in Optimization*. Academic Press, 1971.

[13] Y. Rathi, P.J. Olver, G. Sapiro, and A. Tannenbaum. Affine Invariant Surface Evolutions for 3D Image Segmentation. To appear in ICIP, 2005.

[14] L. Rudin, S. Osher, and E. Fatemi. Nonlinear Total Variation Based Noise Removal Algorithms. *Physica D*, 60:259–268, 1992.

[15] G. Sapiro, P. Olver, and A. Tannenbaum. Affine Invariant Edge Detection: Edge Maps, Anisotropic Diffusion and Active Contours. *Acta. Math. Appl.*, 59, 1999.

[16] J. Shi and C. Tomasi. Good Features to Track. *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, pages 593–600, 1994.

[17] C. Tomasi and T. Kanade. Shape and Motion from Image Streams: a Factorization Method. *International Journal of Computer Vision*, 9(2):137–154, 1992.