SOLVING BOUNDARY VALUE PROBLEMS NUMERICALLY USING STEEPEST DESCENT IN SOBOLEV SPACES

W. T. Mahavier

Abstract. Elementary boundary value problems are used as a vehicle to introduce upper level undergraduate students or first year graduate students to descent algorithms. The paper is expository in nature and includes references for Euclidean and Sobolev steepest descent while serving as an introduction to optimization techniques such as variable metric and conjugate gradient methods. Numerical algorithms for solving boundary value problems with both Euclidean and Sobolev descent are developed. Results for constrained, unconstrained, and singular problems are displayed and properties of descent algorithms are outlined.

1. Introduction. Let k_1, k_2 , k_3, a, $b \in \mathbb{R}$ with $a < b$. Let $f\colon [a,b] \times \mathbb{R} \to \mathbb{R}$ be differentiable with respect to the second variable. This paper illustrates a numerical method for solving the class of first order, boundary value problems with linear inhomogeneous boundary conditions,

$$\begin{gathered} y'(t) = f(t, y(t)) \\ k_1 * y(a) + k_2 * y(b) = k_3. \end{gathered} \tag{1}$$

This paper illustrates a simple application of a universal procedure. Written while the author was a graduate student, it is accessible to upper–level undergraduate and first year graduate students and was used by the author to supplement traditional topics in a two semester first year graduate numerical methods course. By incorporating the material of this paper, students were introduced to finite dimensional vector space theory in such a way that the concepts generalized to infinite dimensional settings [13]. Because this paper rests heavily on results from linear algebra and advanced calculus, we use bullets to highlight these results as they are needed.

Steepest descent is a highly versatile technique. It is used for finding a relative minimum (or maximum) of a function and is often used to find a starting point for Newton's method [2]. Traditional texts on differential equations and numerical methods consider the methods of Euler, Taylor, and Runga–Kutta for initial value problems and two point boundary value problems. To demonstrate the versatility of descent methods, we first consider the class of problems, Equation 1, which incorporates not only initial and final value problems but also problems with mixed boundary conditions that previously mentioned methods are not able to handle. Section 4 includes examples of unconstrained, constrained, non–linear, and singular problems all of which utilize the same algorithm.

Given a function, Φ, and a point in the domain of the function, x_0, the process is to generate a sequence of points in the domain of the function which converge to a point at which Φ attains a relative minimum. The sequence is generated by setting $x_{k+1} = x_k - \delta_k \nabla\Phi(x_k)$ for $k = 0, 1, 2, \ldots$ where δ_k is some small (perhaps optimally chosen) positive number. More thorough introductory information on steepest descent may be found in [10, 20, 21]. Using steepest descent to solve differential equations requires that we determine a function whose minima represent solutions to the differential equation. Such methods were first introduced in Cauchy in [3], however, with the advent of computers it was discovered that such methods were numerically inefficient. Modifications such as conjugate gradient and variable metric methods were introduced to speed up the convergence. In the introduction to Hestenes' book, [7], he states, "Variable metric methods are considered by many to be the most effective technique for optimizing a nonquadratic function." Sobolev steepest descent is an example of this technique where gradients are not based on Euclidean space but on Sobolev spaces which will be defined in this paper. In Section 4 we will see that the choice of the gradient has considerable impact on the speed of convergence of the algorithm.

Sobolev steepest descent originates with J. W. Neuberger and this paper rests firmly on his work. In [16] sufficient conditions are given for the convergence of constrained steepest descent utilizing the Sobolev gradients in continuous spaces. In [12] a convergence proof is given for discrete spaces such as those in this paper and in [13] results for first and second order boundary value problems with linear singularities are considered. Additional theory of Sobolev steepest descent for systems of ordinary differential equations may be found in [17] and references for the application of the method to partial differential equations include [14, 15, 18]. Pressing applied problems have been addressed using these methods in [4, 5, 8]. A general reference for Sobolev spaces is [1].

2. Euclidean Descent. Our space is $(\mathbb{R}^{n+1}, \langle\cdot,\cdot\rangle)$ where $\langle\cdot,\cdot\rangle$ denotes the Euclidean inner product (the "dot" product). Let $\|\cdot\|$ denote the Euclidean norm and denote $x \in \mathbb{R}^m$ by $x = (x_1, \ldots, x_m)$. Let k_1, k_2, k_3, a, $b \in \mathbb{R}$ with $a < b$. Let

$f\colon [a,b]\times\mathbb{R}\to\mathbb{R}$ be differentiable with respect to the second variable. Suppose n is the number of divisions we wish to partition the interval $[a,b]$ into and $\delta=(b-a)/n$.

- If U and V are finite dimensional vector spaces and $L(U,V)$ denotes the set of all linear operators from U to V then every $M\in L(U,V)$ has a matrix representation.

To simplify our notation, we define discretized versions of the identity and derivative operators. It will be simplest to think of D_0 and D_1 as their matrix representations. Let D_0 and $D_1\in L(\mathbb{R}^{n+1},\mathbb{R}^n)$ be defined by

$$D_0(x)=\begin{pmatrix}\frac{x_1+x_2}{2}\\ \vdots\\ \frac{x_n+x_{n+1}}{2}\end{pmatrix}\quad\text{and}\quad D_1(x)=\begin{pmatrix}\frac{x_2-x_1}{\delta}\\ \vdots\\ \frac{x_{n+1}-x_n}{\delta}\end{pmatrix}.$$

In order to solve the differential equation $y'(t)=f(t,y(t))$ we construct a function whose minimum is a solution to the equation. Let $y\in\mathbb{R}^{n+1}$ and for all $k=1,2,\ldots,n+1$, let $t_k=a+(k-1)*\delta$ and $f_k=f(t_k,y_k)$. Define $\Phi\colon(\mathbb{R}^{n+1},\|\cdot\|)\to\mathbb{R}$ by

$$\Phi(y)=\|D_1y-D_0f\|^2/2=1/2\sum_{k=1}^{n}\left(\frac{y_{k+1}-y_k}{\delta}-\frac{f_k+f_{k+1}}{2}\right)^2.$$

If we are able to determine a point, y, in the domain of Φ that satisfies $\Phi(y)=0$ then we have $D_1y=D_0f$. This equation is the discrete version of the differential equation, $y'(t)=f(t,y(t))$, that we desired to solve. As we outlined in the introduction, we will minimize Φ via successive approximations. The function Φ will remain the same for both the Euclidean and Sobolev descent.

<u>Algorithm 1</u>.

1. Choose $y\in\mathbb{R}^{n+1}$.
2. Compute $\nabla\Phi(y)=\left(\frac{\partial\Phi}{\partial y_1},\frac{\partial\Phi}{\partial y_2},\ldots,\frac{\partial\Phi}{\partial y_{n+1}}\right)$.
3. Determine h which minimizes $\Phi(y-h\nabla\Phi(y))$ (or try $h=.01$).
4. Let $y^{\text{new}}=y-h\nabla\Phi(y)$.

5. If $\|y^{\text{new}} - y\| < \epsilon$, we have a solution; else, put $y = y^{\text{new}}$ and repeat steps 2 through 5.

3. Sobolev Descent. The heart of this paper is that $\mathbb{R}^{n+1}$ may be considered as two distinct spaces if we place two different inner products on $\mathbb{R}^{n+1}$. Thus, you should think of the difference between this section and the last one as follows. In the previous section we introduced a pair which were intimately related. The space, $(\mathbb{R}^{n+1}, \langle\cdot,\cdot\rangle)$, and the gradient of Φ based on the space, $\nabla\Phi$. In this section we introduce a second intimately related pair. The space, $(\mathbb{R}^{n+1}, \langle\cdot,\cdot\rangle_s)$, and the gradient of Φ based on this space, $\nabla_s\Phi$. In section 4 you will see that descent based on these two different spaces yields quite different results in terms of the necessary number of iterations and the amount of time required to solve a problem.

We first introduce the notation and theory needed to perform descent in the simplest case; the case where we have no boundary conditions. Introducing the material in this way lays the foundation for the transition to the case with boundary conditions. Our space is $(\mathbb{R}^{n+1}, \langle\cdot,\cdot\rangle_s)$ where $\langle\cdot,\cdot\rangle_s$ denotes the discretized Sobolev inner product defined by

$$\begin{aligned}\langle u,v\rangle_s &= \langle D_0(u), D_0(v)\rangle + \langle D_1(u), D_1(v)\rangle\\ &= \sum_{k=1}^{n}\left(\frac{u_{k+1}+u_k}{2}\right)\left(\frac{v_{k+1}+v_k}{2}\right) + \left(\frac{u_{k+1}-u_k}{2}\right)\left(\frac{v_{k+1}-v_k}{2}\right)\end{aligned}$$

for all $u,v \in \mathbb{R}^{n+1}$. Define $D \in L(\mathbb{R}^{n+1}, \mathbb{R}^{2n})$ by $D(x) = \begin{pmatrix} D_0(x)\\ D_1(x)\end{pmatrix}$ and $\|\cdot\|_s = \sqrt{\langle\cdot,\cdot\rangle_s}$. Observe that D relates the Euclidean and Sobolev norms by $\|\cdot\|_s = \|D(\cdot)\|$. The fact that our new inner product (and norm) take into account "derivatives" provides some intuition as to why they are better suited to solving differential equations via descent methods.

• (Halmos, [6]). If F is a linear functional on the finite dimensional inner product space, $(V, \langle\cdot,\cdot\rangle)$ then there exists a unique element $z \in V$ which satisfies, $F(x) = \langle z, x\rangle$ for all $x \in V$.

• If F is a differentiable function on $\mathbb{R}^n$ and $x \in \mathbb{R}^n$ then $F'(x)$ is a linear functional.

We use these two facts to construct our gradient based on the new inner product space, $(\mathbb{R}^{n+1}, \langle \cdot, \cdot \rangle_s)$. Since $(\mathbb{R}^{n+1}, \langle \cdot, \cdot \rangle_s)$ is a finite dimensional inner product space and for each $u \in \mathbb{R}^{n+1}$, $\Phi'(u)$ is a linear functional, there exists a unique element, z, depending on Φ and u which satisfies $\Phi'(u)(h) = \langle z, h \rangle_s$ for all $h \in \mathbb{R}^{n+1}$. We define $\nabla_s \Phi(u)$ to be this element. In order to compute this new gradient, we prove a theorem which relates the Sobolev gradient to the Euclidean gradient.

Theorem 1. If $\langle \cdot, \cdot \rangle_s$ denotes the discretized Sobolev inner product on $\mathbb{R}^{n+1}$ and $\langle \cdot, \cdot \rangle$ represents the standard inner product on $\mathbb{R}^{n+1}$ then there exists an $A \in L(\mathbb{R}^{n+1}, \mathbb{R}^{n+1})$ such that $\langle x, y \rangle_s = \langle Ax, y \rangle = \langle x, Ay \rangle$ for all $x, y \in \mathbb{R}^{n+1}$. Moreover, $A = D^t D$ and $A \nabla_s \Phi(x) = \nabla \Phi(x)$ for all $x \in \mathbb{R}^{n+1}$.

Proof. Two facts from linear algebra will be useful.

• If $(V, \langle \cdot, \cdot \rangle)$ is a finite dimensional vector space, then $\langle Mx, y \rangle = \langle x, M^t y \rangle$ for every linear operator, M on V.

• If $(V, \langle \cdot, \cdot \rangle)$ is a finite dimensional vector space and $x, y \in V$ such that $\langle x, v \rangle = \langle y, v \rangle$ for every $v \in V$ then $x = y$.

Using these facts and the linearity of the inner product,

$$\begin{aligned}
\langle x, y \rangle_s &= \langle D_0 x, D_0 y \rangle + \langle D_1 x, D_1 y \rangle \\
&= \langle D_0^t D_0 x, y \rangle + \langle D_1^t D_1 x, y \rangle \\
&= \langle (D_0^t D_0 + D_1^t D_1) x, y \rangle
\end{aligned}$$

for all $x, y \in \mathbb{R}^{n+1}$. Therefore, $A = D^t D$. Given $u \in \mathbb{R}^{n+1}$ we have

$$\begin{aligned}
\langle \nabla \Phi(u), h \rangle &= \Phi'(u)(h) \\
&= \langle \nabla_s \Phi(u), h \rangle_s \\
&= \langle A \nabla_s \Phi(u), h \rangle
\end{aligned}$$

for all $h \in \mathbb{R}^{n+1}$. Consequently, $A \nabla_s \Phi(x) = \nabla \Phi(x)$ for all $x \in \mathbb{R}^{n+1}$ and the theorem is proved.

Now we outline the method. Compute the matrix, A, which is diagonally dominant and tridiagonal. Choose an initial guess, y. Compute the standard gradient, $\nabla \Phi(y)$, and solve the linear system from Theorem 1, $A \nabla_s \Phi(y) = \nabla \Phi(y)$, for the Sobolev gradient, $\nabla_s\Phi(y)$. Follow the negative of this direction an 'optimal' distance, h. Since we seek a zero of Φ, 'optimal' implies that $\Phi(y - h\nabla_s\Phi(y))$ is minimized. Consider the distance between the new point, y^{new}, and y. If this distance is less than ϵ, consider y^{new} a solution, else repeat the process with y^{new} as an initial guess.

Algorithm 2.

1. Compute the matrix A.
2. Choose $y \in \mathbb{R}^{n+1}$.
3. Compute the gradient of Φ at $y, \nabla\Phi(y)$.
4. Solve $A\nabla_s\Phi(y) = \nabla\Phi(y)$ for $\nabla_s\Phi(y)$.
5. Determine h which minimizes $\Phi(y - h\nabla_s\Phi(y))$.
6. Let $y^{\text{new}} = y - h\nabla_s\Phi(y)$.
7. If $\|y^{\text{new}} - y\| < \epsilon$, we have a solution; else, put $y = y^{\text{new}}$ and repeat steps 3 through 7.

We have now outlined both Euclidean and Sobolev descent for Equation 1 in the simplest case; the case with no boundary conditions. At this point the reader should be able to write a short code which verifies the results for Sobolev descent in Table 1 of Section 4. Having developed the notation and the spaces, we may now add the boundary conditions, $k_1 * y(a) + k_2 * y(b) = k_3$.

• If $(V, \langle\cdot,\cdot\rangle)$ is a finite dimensional inner product space and $S \subset V$ is a subspace then there exists a function $\pi\colon V \to S$ called the orthogonal projection of V onto S that satisfies, $\|\pi v - v\| \leq \|v - s\|$ for all $s \in S$.

Let $\mathbb{R}_0^{n+1} = \{x \in \mathbb{R}^{n+1} : k_1 * x_1 + k_2 * x_{n+1} = 0\}$ and let π_s denote the orthogonal projection of $\mathbb{R}^{n+1}$ onto $\mathbb{R}_0^{n+1}$ under the Sobolev inner product. A generalization of the case without boundary conditions follows. Start with an initial guess, y. Compute the sobolev gradient, $\nabla_s\Phi(y)$, and the projection, π_s. Project $\nabla_s\Phi(y)$ onto $\mathbb{R}_0^{n+1}$ under π_s and follow the negative of this direction an optimal distance, h. Consider the distance between the new point, y^{new}, and y. If this distance is less than ϵ, consider y^{new} a solution, else repeat the process with y^{new} as our initial guess.

Algorithm 3.

1. Compute the matrix A.
2. Choose $y \in \mathbb{R}^{n+1}$ such that y satisfies the boundary conditions.
3. Compute the gradient of Φ at y, $\nabla\Phi(y)$.

4. Solve $A\nabla_s\Phi(y) = \nabla\Phi(y)$ for $\nabla_s\Phi(y)$.
5. Compute the Sobolev projection, π_s, and project $\nabla_s\Phi(y)$ onto $\mathbb{R}_0^{n+1}$.
6. Determine h which minimizes $\Phi(y - h\pi_s\nabla_s\Phi(y))$.
7. Let $y^{\text{new}} = y - h\pi_s\nabla_s\Phi(y)$.
8. If $\|y^{\text{new}} - y\| < \epsilon$ we have a solution; else, put $y = y^{\text{new}}$ and repeat steps 3 through 8.

Since $\pi_s\nabla_s\Phi(y)$ is in the linear subspace, $\mathbb{R}_0^{n+1}$, boundary conditions are exactly maintained at each iteration. To implement this algorithm we would need an explicit definition for the projection π_s. We avoid this computation, by combining steps 4 and 5 in order to compute the quantity $\pi_s\nabla_s\Phi(y)$ by solving one system of $n+1$ equations. This allows us to avoid computing the projection π_s directly. We now exhibit the method used to compute $\pi_s\nabla_s\Phi(y)$. The next results come from linear algebra.

• If $(V, \langle\cdot,\cdot\rangle)$ is a finite dimensional inner product space and π is a projection on V then $\langle \pi x, y\rangle = \langle x, \pi y\rangle$ for all $x, y \in V$.

Let $g = \nabla_s\Phi(y)$ and define $\gamma\colon \mathbb{R}_0^{n+1} \to \mathbb{R}$ by $\gamma(x) = \|x - g\|_s^2/2$. Minimizing γ over $\mathbb{R}_0^{n+1}$ corresponds to determining $x \in \mathbb{R}_0^{n+1}$ such that $x = \pi_s\nabla_s\Phi(y)$. Let π_e denote the orthogonal Euclidean projection onto $\mathbb{R}_0^{n+1}$.

$$\begin{aligned}\gamma(x) &= \|x - g\|_s^2/2 \\ &= \|D(x-g)\|^2/2\end{aligned}$$

and

$$\begin{aligned}\gamma'(x)(z) &= \langle D(x-g), D(z)\rangle \\ &= \langle D^tD(x-g), z\rangle \\ &= \langle D^tD(x-g), \pi_e(z)\rangle \\ &= \langle \pi_e D^tD(x-g), z\rangle.\end{aligned}$$

Therefore $\gamma'(x)(z) = 0$ for all $z \in \mathbb{R}_0^{n+1}$ if and only if $\pi_e D^tD(x) = \pi_e D^tD(g)$. Substituting $A = D^tD$, $g = \nabla_s\Phi(y)$, and $A\nabla_s\Phi(y) = \nabla\Phi(y)$ into this equation yields $\pi_e Ax = \pi_e\nabla\Phi(y)$. The solution to this equation is the desired quantity, $x = \pi_s\nabla_s\Phi(y)$.

Having avoided direct computation of the projection, π_s, we must still determine the projection, π_e by defining $\psi(x) = \|x-u\|^2/2$ and minimizing ψ over $\mathbb{R}_0^{n+1}$ to obtain

$$\pi_e(x) = \left(\frac{k_2(k_2x_1 - k_1x_{n+1})}{k_1^2 + k_2^2}, x_2, x_3, \ldots, x_n, \frac{-k_1(k_2x_1 - k_1x_{n+1})}{k_1^2 + k_2^2} \right).$$

Certain values of k_1 and k_2 cause numerical difficulties. For example, π_e is not defined if $k_1 = k_2 = 0$. Let us look at each of the four possible cases. If $k_1 = k_2 = 0$, we do not wish to use the projection as we have no boundary conditions; we merely apply Algorithm 2 where π_e was neither defined nor needed. If $k_1 = 0$ and $k_2 \neq 0$ we are considering the final value problem, $y(b) = k_3/k_2$. Here, π_e zeroes out the last row of the matrix A. Therefore, we replace this row by the data, $(0, \ldots, 0, k_2)$, and zero out the final entry of the gradient vector, $\nabla\Phi(y)$, before solving the system. If $k_1 \neq 0$ and $k_2 = 0$, we are considering the initial value problem, $y(a) = k_3/k_1$. In this case π_e zeroes out the first row of the matrix A and we replace this row by the data, $(k_1, 0, \ldots, 0)$ and zero out the first entry of the gradient vector $\nabla\Phi(y)$, before solving the system. If k_1 and k_2 are both non-zero, π_e is well defined, yet when we apply the projection to the matrix A, the first and last rows of the projected matrix are linearly dependent. In this case, we replace the last row by the boundary data, $(k_1, 0, \ldots, 0, k_2)$ and zero out the last entry of the gradient vector, $\nabla\Phi(y)$, before solving the system. Notice that in each of these cases, replacing the appropriate rows of the matrix assures that $\pi_e A$ is non–singular and that the solution to the system will be an element of $\mathbb{R}_0^{n+1}$ so that boundary conditions are maintained at each iteration.

A revised algorithm follows.

1. Compute the matrix A and the projection π_e.
2. Choose $y \in \mathbb{R}^{n+1}$ such that y satisfies the boundary conditions.
3. Compute the gradient of Φ at y, $\nabla\Phi(y)$.
4. Apply π_e to the matrix A and the gradient $\nabla\Phi(y)$.
5. Make $\pi_e A$ nonsingular by replacing the necessary rows.
6. Solve $\pi_e A(\pi_s \nabla_s \Phi(y)) = \pi_e \nabla\Phi(y)$ for $\pi_s \nabla_s \Phi(y)$.
7. Determine h which minimizes $\Phi(y - h\pi_s \nabla_s \Phi(y))$.
8. Let $y^{\text{new}} = y - h\pi_s \nabla_s \Phi(y)$.

9. If $\|y^{\text{new}} - y\| < \epsilon$, we have a solution; else, put $y = y^{\text{new}}$ and repeat steps 3 through 9.

4. Examples. In this section we consider three problems. In Section 5 we illustrate the algorithm on the differential equation $y' = y$. We consider the unconstrained problem and the initial value problem. In Section 6 we consider a nonlinear equation with mixed boundary conditions and in Section 7 we address Legendre's equation which is singular at $x = 1$. Neither of the problems considered in Sections 6 and 7 could be solved by methods traditionally taught in undergraduate differential equations or numerical methods courses.

A desired accuracy of ϵ requires that $\|y^{new} - y\| < \epsilon$ and average absolute error is defined by

$$\text{average absolute error} = \frac{1}{n+1} \sum_{i=1}^{n+1} |y_k^{\text{true}} - y_k^{\text{approx}}|,$$

while maximum absolute error is given by

$$\text{maximum absolute error} = \max\{|y_k^{\text{true}} - y_k^{\text{approx}}| : k = 1, 2, \ldots, n+1\}.$$

$y' = y$	$\epsilon = 10^{-2}$		$N = 100$	
Gradient	Iterations	Seconds	Aver. Abs. Error	Max. Abs. Error
E	1845	3	5.9×10^{-1}	9.2×10^{-1}
S	4	1	5.6×10^{-4}	7.5×10^{-4}

$y' = y$	$\epsilon = 10^{-4}$		$N = 1000$	
Gradient	Iterations	Seconds	Aver. Abs. Error	Max. Abs. Error
E	14,777	27	6.6×10^{-3}	9.2×10^{-3}
S	6	1	8.7×10^{-6}	1.0×10^{-5}

$y' = y$	$\epsilon = 10^{-6}$		$N = 10,000$	
Gradient	Iterations	Seconds	Aver. Abs. Error	Max. Abs. Error
S	8	22	1.9×10^{-7}	2.6×10^{-7}

Table 1: The simplest differential equation.

5. Results for $\mathbf{y' = y}$. All the results in this section correspond to the differential equation $y' = y$ on the interval $[0, 1]$ with an initial guess of $y_0 = 3$. The unconstrained case is included because the results in Table 1 may be obtained without reading beyond Algorithm 2. Since all solutions are of the form $c * E$ where $E(t) = e^t$ on $[0, 1]$, we may determine the solution to which each process will converge by minimizing the functions, $\gamma_E(c) = \|y_0 - cE\|_L^2$ for Euclidean descent and $\gamma_S(c) = \|y_0 - cE\|_H^2$ for Sobolev descent where

$$\|f\|_L = \sqrt{\int_0^1 f^2} \text{ and } \|f\|_H = \sqrt{\int_0^1 (f^2 + (f')^2)}.$$

These functions are quadratic in c; hence the minimum is easily obtained.

Table 1 indicates the difference between Euclidean (E) and Sobolev (S) descent for the unconstrained case. Several observations are in order. First, Sobolev descent outperforms Euclidean descent in terms of the time required to solve the problem, the accuracy achieved, and the number of iterations. Second, as we increase the number of divisions from 100 to 1,000, Euclidean descent requires approximately 9 times the number of iterations to converge. On the other hand, even as we increase the number of divisions from 100 to 10,000, Sobolev descent requires only twice the number of iterations. Third, the order of magnitude of the error in the solution obtained via Sobolev descent is comparable to the desired accuracy while for Euclidean descent the achieved accuracy is consistently lower than the desired accuracy. Finally, for $N = 10,000$ we have omitted Euclidean descent as convergence was not obtained in this case.

Table 2 contains the results for the initial value problem for Sobolev descent. Euclidean results are omitted, but are similar to those in Table 1.

$y' = y$ $\quad y(0) = 3$ $\quad y_0 = 3$ $\quad \epsilon = 10^{-5}$ $\quad N = 10,000$			
Gradient	Iterations	Seconds	Max. Abs. Error
S	21	5	2.3×10^{-5}

Table 2: The initial value problem.

$y' = (t+y)^2$ $y_0(t) = -t$ $\epsilon = 10^{-3}$ $N = 100$			
Gradient	Iterations	Seconds	Max. Abs. Error
S	6	1	3.0×10^{-7}

Table 3: Nonlinear with Mixed Boundary Conditions.

6. A Nonlinear Example with Mixed Boundary Conditions. The results in Table 3 apply to the differential equation, $y'(t) = (t+y(t))^2$ with boundary conditions, $y(-\pi/4)+y(\pi/4) = 0$ on the interval $[-\pi/4, \pi/4]$. The solution is given by $y(t) = \tan(t) - t$.

7. Legendre's equation. As a final example we present results for Legendre's equation which has a singularity at $t = 1$. Table 4 includes data for descent based on three distinct gradients. The first two gradients are the Euclidean (E) and Sobolev (S) gradients as introduced in previous sections. The last gradient is based on a weighted Sobolev (S_w) inner product, i.e., a gradient custom built for the singular nature of the problem at hand. Detailed information on the modifications made to the algorithm may be found in [13]. A general reference on weighted spaces is [9].

The problem is to solve $((1-t^2)u')' + 2u = 0$ on I with $u(0) = 0$, $u(1) = 1$, and $u \in \mathbb{C}_I^2$. General solutions are $u(t) = c_1 t + \frac{c_2}{2} t \ln\left(\frac{1+t}{1-t}\right)$ and only $u(t) = t$ satisfies the boundary conditions.

For the second and third tables we increase the number of divisions of the interval and decrease the desired accuracy. The second table omits the Euclidean descent results which appeared in the first table as Euclidean descent will not converge for the tighter stopping criteria. Likewise, the third table omits both the Euclidean and Sobolev descent because neither converge for the still tighter stopping criteria.

These tables demonstrate clearly the advantage of applying gradients for a given problem. We see that the weighted Sobolev descent outperforms Sobolev descent which in turn outperforms Euclidean descent in terms of the time and the number of iterations required to achieve a desired accuracy.

$((1-t^2)u')' + 2u = 0 \quad y(0) = 0 \quad y(1) = 1 \quad \epsilon = 10^{-6} \quad N = 100$				
Grad.	Iter.	Sec.	Av. Abs. Err.	Max. Abs. Error
E	5948	24	1.0×10^{-1}	6.6×10^{-1}
S	1998	7	1.0×10^{-6}	3.7×10^{-5}
S_w	64	1	1.0×10^{-7}	8.0×10^{-6}

$((1-t^2)u')' + 2u = 0 \quad y(0) = 0 \quad y(1) = 1 \quad \epsilon = 10^{-6} \quad N = 10,000$				
Grad.	Iter.	Sec.	Av. Abs. Err.	Max. Abs. Error
S	2142	82	1.0×10^{-6}	3.4×10^{-5}
S_w	85	3	1.0×10^{-6}	1.2×10^{-5}

$((1-t^2)u')' + 2u = 0 \quad y(0) = 0 \quad y(1) = 1 \quad \epsilon = 10^{-15} \quad N = 100,000$				
Grad.	Iter.	Sec.	Av. Abs. Err.	Max. Abs. Error
S_w	325	125	1.0×10^{-14}	1.7×10^{-14}

Table 4: Legendre's Equation.

8. Conclusions. In this section we comment on a few of the nicer properties of descent algorithms that we have observed. Tables 1 and 4 support these observations, but do not represent isolated incidences. The author has applied the method to some 50 or more problems and the following traits appear consistently.

1. For the problems presented weighted Sobolev descent outperforms Sobolev descent which in turn outperforms Euclidean descent.

2. Sobolev descent is not sensitive to the number of divisions of the interval (see Table 1). Given a desired accuracy, the number of iterations required for convergence of Sobolev descent does not increase significantly as the number of divisions is increased. Such properties are particularly significant when considering partial differential equations where the number of grid points is greatly increased and experimental runs may use courser grids.

3. The same statement cannot be made of Euclidean descent.

4. The average absolute error and maximum absolute error for both Sobolev and weighted Sobolev descent are on the order of magnitude of the residual. Restated, if the stopping criteria is $\|y^{\text{new}} - y\| < \epsilon$ for successive iterations y and y^{new} then we may expect that upon convergence to a solution, y^{approx}, we have

$$\frac{1}{n+1}\sum_{i=1}^{n+1}|y_k^{\text{true}} - y_k^{\text{approx}}| \cong \epsilon$$

and

$$\max\{|y_k^{\text{true}} - y_k^{\text{approx}}| : k = 1, 2, \dots, n+1\} \cong \epsilon.$$

5. Exact boundary conditions are maintained at each step of the descent process, guaranteeing exact boundary conditions for the solution. This is a consequence of the algorithm. Because we perturb our solutions by elements which satisfy zero boundary conditions, the initial boundary conditions are never changed.

6. The generalization of the code to handle singular multiple point boundary value problems such as $(t-1/4)(t-3/4)y' = y$ on $[0,1]$ with constraints, $y(1/4) = 1$, and $y(3/4) = 2$, requires only a slightly more difficult computation of the projection, π_e.

7. Generalization of the code to partial differential equations is type independent, [14, 19]. Many numerical methods which solve partial differential equations use theory which applies only to elliptic, parabolic, or hyperbolic equations while many pressing problems address differential equations which are of mixed type.

References

1. R. A. Adams, *Sobolev Spaces*, Academic Press, 1975.
2. R. L. Burden and J. D. Faires, *Numerical Analysis*, PWS Publishing Co., Boston, 1993.
3. P. L. Cauchy, *Méthode Générale pour la Résolution des Systemes D'équations Simultanées*, C. R. Acad. Sci. Paris, 25 (1847).
4. J. G. Dix and T. W. McCabe, "On Finding Equilibria for Isotropic Hyperelastic Materials," *Nonlinear Analysis*, 15 (1990), 437–444.
5. J. Garza, "Using Steepest Descent to Find Energy–Minimizing Maps Satisfying Nonlinear Constraints," Dissertation, University of North Texas (1994).
6. P. R. Halmos, *Finite Dimensional Vector Spaces*, D. Van Nostrand Co. Inc., New York, 1958.
7. M. R. Hestenes, *Conjugate Direction Methods in Optimization*, Springer–Verlag, New York, NY, 1980.

8. K. Kim, "Steepest Descent for Partial Differential Equations of Mixed Type," Dissertation, University of North Texas, 1992.

9. A. Kufner, *Weighted Sobolev Spaces*, John Wiley & Sons, New York, NY, 1985.

10. D. G. Luenberger, *Linear and Nonlinear Programming*, Addison–Wesley, Reading, MA, 1989.

11. W. T. Mahavier, "An Interactive Numerical Analysis Course," *Creative Math Teaching*, 3.2 (1996).

12. W. T. Mahavier, "A Convergence Result for Discrete Steepest Descent in Weighted Sobolev Spaces," *Journal of Abstract and Applied Mathematics*, 2.1–2 (1997).

13. W. T. Mahavier, "A Numerical Method Utilizing Weighted Sobolev Descent to Solve Singular Differential Equations," *Nonlinear World*, 4.4 (1997).

14. W. T. Mahavier, "A Numerical Method Utilizing Weighted Sobolev Steepest Descent for Singular Partial Differential Equation," (in preparation).

15. J. W. Neuberger, "A Numerical Method for Finding Sign–Changing Solutions of Super Linear Dirichlet Problems," *Nonlinear World*, 1996, (to appear).

16. J. W. Neuberger, "Steepest Descent and Differential Equations," *J. Math. Soc. Japan*, 37 (1985).

17. J. W. Neuberger, "Steepest Descent for General Systems of Linear Differential Equations in Hilbert Space," *Ordinary Differential Equations and Operators*, Springer–Verlag Lecture Notes in Mathematics, 1032, (1982), 390–406.

18. J. W. Neuberger, "Iteration for Systems of Nonlinear Partial Differential Equation," *Nonlinear Equations in Abstract Spaces*, Academic Press, Inc., 1978, 253–263.

19. J. W. Neuberger, *A Type–Independent Method for Systems of Nonlinear Partial Differential Equation*, Oak Ridge National Laboratory, CSD TM–161 (1981).

20. J. M. Ortega and W. C. Rheinboldt, *Iterative Solution of Nonlinear Equations in Several Variables*, Academic Press, New York, 1970.

21. A. Ralston and P. Rabinowitz, *A First Course in Numerical Analysis*, McGraw–Hill, New York, 1978.

W. T. Mahavier
Department of Mathematics
Nicholls State University
Thibodaux, LA 70310
email: math-wtm@nich-nsunet.nich.edu