A Reordering Approach to Numerical Analysis of Boundary Value Problems

Carl Boettiger

August 18, 2006

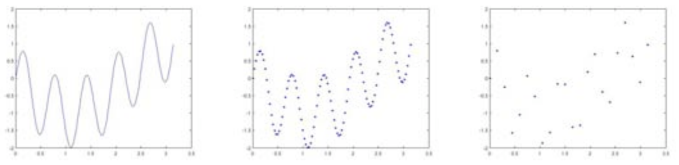
1 Statement of problem

Boundary value problems (BVP) are encountered in nearly every application of mathematical modeling, from plasma physics to biochemistry to socio-economic models. In complicated systems, finding analytic solutions often becomes impractical or impossible. Consequently, a vast array of powerful numerical techniques have been developed over the past century to determine approximate solutions using a computer. Despite this development, modern problems remain very demanding, and can not only face limitations of computing power but also limitations inherent in the equations themselves. We propose and develop an entirely novel approach to sidestep such inherent problems.

This class of BVP's is composed of those whose solutions have a high degree of fine structure, (figure 1). Numerical problems must be discretized on a finite mesh or grid of points, and highly structured functions require a very fine mesh to resolve that structure. Not only is such a fine mesh computationally expensive, but will also force many traditional approaches to become ill-conditioned to produce inaccurate solutions. In order to escape this difficulty, we transform the problem into a space where it can be solved on a much coarser mesh.

2 Approach

Imagine for a moment that solution to the differential equation is a highly structured function we will call $f(x)$, Figure 1a. We discretize this function



(a) (b) (c)

Figure 1: A highly structured function, discretized on fine and coarse grids

into a series of points, Figure 1b. We can represent this discretization by listing the values at each of these points, [0, .26. 50, . . .] as the components of a vector $\vec{f}$. (Note that if we do not use a fine enough mesh, we cannot resolve the structure, figure 1c.) We represent the grid points where these points occur by a similar vector list denoted $\vec{x}$. Plotting each component of $\vec{x}$ against its corresponding component in $\vec{f}$ gives us figure 1b. The essence of our approach is to find a new space $\vec{y}$ where $\vec{f}$ looks less structured. This space $\vec{y}$ will simply be some reordering of the components in $\vec{x}$ such that when we plot each $\vec{y}$ component against $\vec{f}$ to get a plot or reduced structure, as in figure 2. In this new space we can use a much coarser grid and still resolve the essential structural features. Now the problem can be solved on this coarse grid with much less difficulty than on a fine grid. Once we have the solved the problem in this space, we simply transform back to the $\vec{x}$ space to recover our final answer. The challenge, then, is to find the appropriate choice for a $\vec{y}$ space.

The implementation of this approach is as follows. We begin with a discritized differential equation of the form

$$\mathbb{L}\vec{f} = \vec{g} \tag{1}$$

Here $\mathbb{L}$ represents a linear combination of differential operators, perhaps $a\frac{\mathrm{d}^2}{\mathrm{d}x^2} + b\frac{\mathrm{d}}{\mathrm{d}x} + c$, and $\vec{g}$ represents the given inhomogeneity and specified boundary conditions. The task is to determine values for $\vec{f}$. We assume that $\vec{f}$ is highly structured, and consequently the discretized problem has been defined on a suitable fine grid $\vec{x}$ with a large number of N points (that is, the vectors are of dimension N). We postulate the existence of an operator (permutation matrix) $\mathbb{P}$ such that $\mathbb{P}\vec{f}$ is much less structured as discussed above. Since

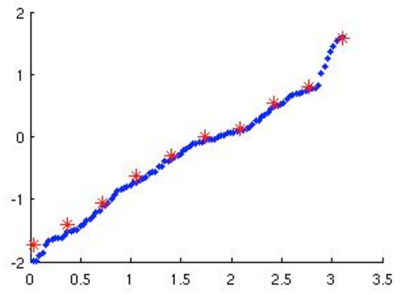


Figure 2: Reordering the structured function in Figure 1 allows a coarse grid approximation, indicated by the red stars.

$\mathbb{P}^T\mathbb{P} = \mathbb{I}$ we can rewrite the equation,

$$\mathbb{L}\mathbb{P}^T\mathbb{P}\vec{f} = \vec{g} \tag{2}$$

With $\mathbb{P}\vec{f}$ being less structured by assumption, we need not specify $\mathbb{P}\vec{f}$ (the image of $\vec{f}$ in the $\vec{y}$ space) at every point on the fine mesh. We instead approximate this by taking its values at only a few M points, $\vec{f'}$ (such as the highlighted points in Figure 2) and then smoothly interpolating the points in between by means of the operator Ω, that is, $\Omega\vec{f'} \approx \mathbb{P}\vec{f}$. Introducing this approximation into equation (2) gives us

$$\mathbb{L}\mathbb{P}^T\Omega\vec{f'} = \vec{g} \tag{3}$$

Now the only unknowns are the few M points in $\vec{f'}$, rather than the many N points in $\vec{f}$, and consequently the problem is much easier to solve. Several methods are possible, perhaps the most obvious being a least squares solution. The advantage in this is two-fold. Not only are there now many fewer equations to solve (at the cost of three sparse matrix multiplications), but although the original operator $\mathbb{L}$ will tend to be nearly singular for large N, the new operator $\mathbb{A} := \mathbb{L}\mathbb{P}^T\Omega$ should be well-conditioned.

Thus far we have proceeded under the assumption of an appropriate $\mathbb{P}$ such that the approximation made above will be satisfactory for small M. The real task now is to find such a $\mathbb{P}$. This is accomplished iteratively by means of a genetic algorithm. The process begins with a “population” of 100 random guesses for permutations $\mathbb{P}$. Each permutation is tested in solving equation (3), giving a corresponding guess for $\vec{f'}$. Then both the permutation and its corresponding guess for $\vec{f'}$ are inserted into the right hand side of equation 3, which determines an estimate for the values of $\vec{g}$. Since the vector $\vec{g}$ is already known, we can it to this estimate to give a metric on which to assess how good was the initial guess $\mathbb{P}$. This process is repeated for each of the 100 random guesses of $\mathbb{P}$, and each are assigned a score based on the metric described. The score functions as a "fitness" for each member of the population, determining the probability that it will "survive" (reproduce) in the next "generation." Those that survive to the next generation have some probability of having a "mutation" occur, slightly modifying the permutation, as well as the chance of a "crossover," where a section of the permutation is replaced with the sequence of another permutation (duplicate values are eliminated and missed values added to the end). In this manner, the population changes stochastically, selecting for better and better permutations that emerge as a result of these mutations and crossovers. The algorithm terminates after a fixed number of generations or when several successive generations no longer improve the population fitness.

3 Advances and Current Status

We have demonstrated that the genetic algorithm is capable of finding acceptable permutations on a variety of model problems that are difficult or impossible to solve using existing algorithms, such as MATLAB's boundary value problem solver, `bvp4c`. Furthermore we determined that any existing algorithms relying on standard approaches such as finite elements or finite differencing will result in near-singular matrices on the problems considered, leaving shooting techniques as the only commonly employed viable alternative. Like each of these traditional approaches, the computational time required for to solve the equations should scale linearly with N, although the convergence of the genetic algorithm appears to have a steeper dependence.

The technique outlined here and the crucial result that a relatively simple search algorithm (the basic genetic algorithm) can find adequate permutations provides many possible directions for further development. Several of these have been explored with varying success and the work is currently proceeding towards publication.