Project 1

Josh Bradt

February 12, 2016

1 Introduction

Efficiently solving differential equations is essential to many problems in computational science. One particularly frequent class of differential equations are linear second-order differential equations, which can be written as

$$\frac{d^2y}{dx^2} + k(x)y = f(x) \tag{1}$$

for some source function $f(x)$ and a real function $k(x)$.

One example of an equation of this form is found in classical electrostatics. There, the electric field of a point charge can be found using Poisson's equation:

$$\nabla^2 \Phi(\mathbf{r}) = -4\pi\rho(\mathbf{r}) \tag{2}$$

where $\rho(\mathbf{r})$ is the charge distribution. Assuming spherical symmetry, this becomes a one-dimensional equation

$$\frac{1}{r^2}\frac{d}{dr}\left(r^2\frac{d\Phi}{dr}\right) = -4\pi\rho(r)$$

which can be written as

$$\frac{d^2\phi}{dr^2} = -4\pi r\rho(r)$$

by letting $\Phi(r) = \phi(r)/r$. This is now a linear second-order differential equation of the form shown in (1) where $k(r) = 0$ and $f(r) = -4\pi r\rho(r)$. To simplify things further, let $r \to x$ and $\phi \to u$, and then define $f(x) = -4\pi x\rho(x)$. Then our equation becomes

$$-u''(x) = f(x)$$

Equations of this form can occasionally be solved analytically, but in general they must be solved using numerical methods.

2 Numerical algorithm

To make the problem more concrete, we will be solving the equation

$$-u''(x) = f(x) \tag{3}$$

on the domain $x \in [0, 1]$ with Dirichlet boundary conditions $u(0) = u(1) = 0$.

2.1 Transformation to linear equations

The second derivative can be found using the second-order finite difference relation

$$u''(x) \approx \frac{u(x+h) + u(x-h) - 2u(x)}{h^2} + O(h^2) \tag{4}$$

for some small step size h. Plugging this relation into (3) produces the equation

$$-\frac{u(x+h) + u(x-h) - 2u(x)}{h^2} = f(x). \tag{5}$$

Next, we discretize the problem by creating a mesh of step size h between the lower and upper boundaries. This is conceptually the same as representing the functions $u(x)$ and $f(x)$ as vectors u_i and f_i. Thus, we can write

$$-\frac{u_{i+1} + u_{i-1} - 2u_i}{h^2} = f_i, \quad i = 1, \ldots, n. \tag{6}$$

Thinking of u and f as vectors, this can be interpreted as taking the $(i+1)$-th element of u, the $(i-1)$-th element of u, and so on. This leads to a natural interpretation of this equation in terms of a set of linear equations

$$\begin{pmatrix} 2 & -1 & & & \\ -1 & 2 & -1 & & \\ & -1 & \ddots & \ddots & \\ & & \ddots & \ddots & -1 \\ & & & -1 & 2 \end{pmatrix} \begin{pmatrix} u_1 \\ u_2 \\ u_3 \\ \vdots \\ u_n \end{pmatrix} = \begin{pmatrix} w_1 \\ w_2 \\ w_3 \\ \vdots \\ w_n \end{pmatrix} \tag{7}$$

where $w_i \equiv h^2 f_i$, and all elements not shown in the matrix are taken to be zero. This is a *tridiagonal* matrix, meaning it has elements only on the primary diagonal and on the diagonals above and below it.

Rewriting the problem in this way turns the relatively difficult task of solving a differential equation into the well-studied task of solving a system of linear equations.

2.2 General Gaussian elimination

One of the most basic and well-known methods for solving linear equations is the method of Gaussian elimination. Solving a system of equations in this method consists of two steps: forward substitution and backward substitution.

Starting with a general matrix

$$\begin{pmatrix} a_{11} & a_{12} & a_{13} & a_{14} & w_1 \\ a_{21} & a_{22} & a_{23} & a_{24} & w_2 \\ a_{31} & a_{32} & a_{33} & a_{34} & w_3 \\ a_{41} & a_{42} & a_{43} & a_{44} & w_4 \end{pmatrix}$$

with the solution vector $\mathbf{w}$ appended as the last column, the forward substitution step consists of adding multiples of each row to the rows below it until

the matrix becomes upper-triangular. Thus, after one substitution, the original matrix above becomes

$$\begin{pmatrix} a_{11} & a_{12} & a_{13} & a_{14} & w_1 \\ 0 & a'_{22} & a'_{23} & a'_{24} & w'_2 \\ 0 & a'_{32} & a'_{33} & a'_{34} & w'_3 \\ 0 & a'_{42} & a'_{43} & a'_{44} & w'_4 \end{pmatrix}, \text{ where } \begin{cases} a'_{ij} & = a_{ij} - \dfrac{a_{i1}}{a_{11}} a_{1j} \\ w'_i & = w_i - \dfrac{a_{i1}}{a_{11}} w_1 \end{cases}.$$

After a second substitution, it becomes

$$\begin{pmatrix} a_{11} & a_{12} & a_{13} & a_{14} & w_1 \\ 0 & a'_{22} & a'_{23} & a'_{24} & w'_2 \\ 0 & 0 & a''_{33} & a''_{34} & w''_3 \\ 0 & 0 & a''_{43} & a''_{44} & w''_4 \end{pmatrix}, \text{ where } \begin{cases} a''_{ij} & = a'_{ij} - \dfrac{a'_{i2}}{a'_{22}} a'_{2j} \\ w''_i & = w'_i - \dfrac{a'_{i2}}{a'_{22}} w'_2 \end{cases}.$$

Finally, after a third substitution, we have

$$\begin{pmatrix} a_{11} & a_{12} & a_{13} & a_{14} & w_1 \\ 0 & a'_{22} & a'_{23} & a'_{24} & w'_2 \\ 0 & 0 & a''_{33} & a''_{34} & w''_3 \\ 0 & 0 & 0 & a'''_{44} & w'''_4 \end{pmatrix}, \text{ where } \begin{cases} a'''_{ij} & = a''_{ij} - \dfrac{a''_{i3}}{a''_{33}} a''_{3j} \\ w'''_i & = w''_i - \dfrac{a''_{i3}}{a''_{33}} w''_3 \end{cases}.$$

This pattern is described in pseudocode in Algorithm 1. In this formation, the variable k represents the index of the row currently being subtracted from the other rows, and i and j represent the row and column being operated on. The multiplicative factor is computed once per row and stored in C for efficiency.

Algorithm 1 Gaussian elimination forward substitution

Require: a is an $N \times N$ array
Require: w is a length-N vector
  for $k = 0, 1, \ldots, N-1$ **do**
    for $i = k+1, k+2, \ldots, N$ **do**
      $C \leftarrow a_{ik}/a_{kk}$
      $w_i \leftarrow w_i - Cw_k$
      for $j = k, k+1, \ldots, N$ **do**
        $a_{ij} \leftarrow a_{ij} - Ca_{kj}$
      end for
    end for
  end for

Once the forward-substitution step is complete, we proceed with backward substitution to fully diagonalize the matrix. This process only affects the solution vector $\mathbf{w}$. In the following steps, assume the b_{ij} are the elements of the matrix after forward substitution is complete, and the v_i are the elements of the solution vector after forward substitution.

Algorithm 2 Gaussian elimination backward substitution

Require: b is an $N \times N$ array
Require: v is a length-N vector
  for $k = N, N-1, \ldots, 1$ **do**
    for $i = k-1, k-2, \ldots, 0$ **do**
      $C \leftarrow b_{ik}/b_{kk}$
      $v_i \leftarrow v_i - Cv_k$
    end for
  end for

First, after one step of backward substitution, we have

$$\begin{pmatrix} b_{11} & b_{12} & b_{13} & 0 & v_1' \\ 0 & b_{22} & b_{23} & 0 & v_2' \\ 0 & 0 & b_{33} & 0 & v_3' \\ 0 & 0 & 0 & b_{44} & v_4 \end{pmatrix}, \text{ where } v_i' = v_i - \frac{b_{i4}}{b_{44}} v_4.$$

After a second step, this becomes

$$\begin{pmatrix} b_{11} & b_{12} & 0 & 0 & v_1'' \\ 0 & b_{22} & 0 & 0 & v_2'' \\ 0 & 0 & b_{33} & 0 & v_3' \\ 0 & 0 & 0 & b_{44} & v_4 \end{pmatrix}, \text{ where } v_i'' = v_i' - \frac{b_{i3}}{b_{33}} v_3'.$$

And finally, the last step produces

$$\begin{pmatrix} b_{11} & 0 & 0 & 0 & v_1''' \\ 0 & b_{22} & 0 & 0 & v_2'' \\ 0 & 0 & b_{33} & 0 & v_3' \\ 0 & 0 & 0 & b_{44} & v_4 \end{pmatrix}, \text{ where } v_i''' = v_i'' - \frac{b_{i2}}{b_{22}} v_3''.$$

At this point, the system of equations is clearly solved.

This backward substitution process is described in Algorithm 2.

This is clearly not the most computationally efficient algorithm for solving a system of equations. By inspecting the algorithm, we can see that the forward substitution step scales as $O(n^3)$, while the backward substitution step scales as $O(n^2)$. A more efficient method for solving our particular problem will be outlined in the next section.

2.3 A specialized algorithm

In Equation 7, we showed that the equation that we're actually trying to solve consists of a tridiagonal matrix. This means that most of the elements are zeros, and the remaining elements only take one of two constant values. Therefore, if we use the general algorithm described in the previous section, it will mostly be wasting time repeating the same calculations. Additionally, it's a waste of

Algorithm 3 A specialized Gaussian solver

Require: a, c are constants equal to the upper and lower diagonals
Require: b is a length-N vector representing the diagonal elements
  for $i = 1, 2, \ldots, N$ **do** ▷ Forward substitution
    $M \leftarrow a/b_{i-1}$
    $b_i \leftarrow b_i - Mc$
    $w_i \leftarrow w_i - Mw_{i-1}$
  end for
  for $i = N-1, N-2, \ldots, 0$ **do** ▷ Backward substitution
    $w_i \leftarrow w_i - \frac{c}{b_{i+1}} w_{i+1}$
  end for

memory to represent the equation using a dense matrix, and it may become impossible to do so as the step size gets smaller.

A more efficient way to represent the matrix is as three