PROJECT 1 - TRIDIAGONAL MATRIX SOLVER

MARCO SANTIA

Abstract. Numerical methods for discretizing linear second-order differential equations are compared for the one-dimensional Poisson equation case. Finite-difference discretization in this case produces a tridiagonal matrix when using Dirichlet boundary conditions that is solved with a simplified Gauss-elimination algorithm know as the Thomas algorithm. The problem is then solved using a problem-specific variant of the Thomas algorithm as well as a LU-decomposition method requiring full matrix-construction. It was found that all three methods produce very high accuracy with respect to the exact solution for moderately sized grids. The LU decomposition method becomes memory restrictive without sparse matrix methods for large grid sizes. We find the problem specific algorithm to provide relative speedups compared to the general Thomas algorithm. Both of which scale linearly with grid size compared to the cubic scaling of the LU method.

1. Introduction

Many physical phenomena can be described by elliptical partial differential equations which contain second order derivatives. For example in semiconductor device modeling the charge density ρ the Poisson equation is used

$$\nabla^2\Phi = \frac{q}{e}(n - p + N_A - N_D) \tag{1}$$

For these devices it is essential to determine population densities of acceptors N_A and donors N_D as well as electrons n and holes p. In this work, a simplified model of the more general equation is solved using a spherically symmetric system in one dimension:

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

and by letting $\phi \to u$ and $r \to x$ we get

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

For the purpose of this work we utilize Dirichlet boundary conditions for simplicity and a source function $f(x) = 100e^{-10x}$ with a known analytical solution $u(x) = 1 - (1 - e^{-10})x - e^{-10x}$ to compare to. This allows for the system to be rewritten by a set of linear equations. The solution variables are written as v_i with grid points $x_i = ih$ and a grid spacing of $h = \frac{1}{n+1}$ from $x_0 = 0$ to $x_{n+1} = 1$. The dirichlet boundary conditions are expressed as $v_0 = v_n + 1 = 0$ which leads to the approximation:

$$-\frac{v_{i+1} - 2v_i + v_{i-1}}{h^2} = f_i \tag{4}$$

by then rearranging this equation and defining $\tilde{b}_i = h^2 f_i$ we get

$$-v_{i+1} - 2v_i + v_{i-1} = \tilde{b}_i \tag{5}$$

which can be written in matrix form

$$\mathbf{A}\mathbf{v} = \tilde{\mathbf{b}},$$

where $\mathbf{A}$ is an $n \times n$ tridiagonal matrix which we rewrite as

$$\mathbf{A} = \begin{bmatrix} 2 & -1 & 0 & \dots & \dots & 0 \\ -1 & 2 & -1 & 0 & \dots & \dots \\ 0 & -1 & 2 & -1 & 0 & \dots \\ & \dots & \dots & \dots & \dots & \dots \\ 0 & \dots & & -1 & 2 & -1 \\ 0 & \dots & & 0 & -1 & 2 \end{bmatrix},$$

I

2. General Thomas Algorithm

For a general matrix that does not necessarily have the tridiagonal property, the linear system could be solved by Gaussian elimination. Although effective, the method scales as $\frac{1}{3}(n^3 + 3n^2 - n)$ which is extremely inefficient when considering the desirable matrix properties our system contains. In addition to repeated values down the diagonals, we have a high degree of sparsity and tridiagonal structure. As with many sparse matrices, it would be ideal to construct an iterative approach that does not require explicit matrix construction and avoids any gaussian elimination steps that are trivial. To remedy this, we utilize the Thomas algorithm [1]. To begin in a more general framework, we do not assume the diagonals of our matrix contain only one repeated value. The steps are summarized:

$$m = \frac{a_k}{b_{k-1}} \tag{6}$$

$$b_k = b_k - mc_{k-1} \tag{7}$$

$$\tilde{b}_k = \tilde{b}_k - m\tilde{b}_{k-1} \tag{8}$$

and equivalently in FORTRAN code,

```
subroutine tridiag_general(a, b, c, v_sol, rhs, N)
!{=============================================
!  Solves the general tridiagonal setup where
!  a_i and c_i do not need to retain the same
!  values down their diagonals
! ============================================
    implicit none
    integer(kind=4) :: i
    integer(kind=4), intent(in) :: N
    real(kind=8), dimension(N), intent(inout) :: a, b, c, rhs
    real(kind=8), dimension(N), intent(inout) :: v_sol
    real(kind=8), dimension(N) :: rhs_new
    real(kind=8) :: m

    ! FORWARD SUBSTITUTION
    rhs_new(1) = rhs(1)
    do i=2, N
        m = a(i)/b(i-1)
        b(i) = b(i)-m*c(i-1)
        rhs_new(i) = rhs(i) - m*rhs_new(i-1)
    end do
```

repeating these operators will triangularize the matrix such that the solution can be computed through a simple backwards substitution:

$$v_n = \frac{\tilde{b}_n}{b_n} \tag{9}$$

iterating in reverse from $k = n - 1$ to $k = 1$,

$$v_k = \frac{\tilde{b}_k - c_k v_{k+1}}{b_k} \tag{10}$$

The backward substitution is implemented in FORTRAN like so:

```
!BACKWARD SUB STEP
v_sol(N) = rhs_new(N)/b(N)
do i = N-1, 1, -1
    v_sol(i) = (rhs_new(i)-c(i)*v_sol(i+1))/b(i)
end do
```

This method will be much more efficient as it does not require matrix computations over the sparse majority of the matrix. In terms of floating point operations, there will be 6 FLOPS per iteration in the forward substitution and additional 2 from the backward substitution. This gives a total of $8(N-1)$ flops which is approximately $O(N)$.

3. Specialized Thomas Algorithm

For the specific matrix used in this case, we can use the property that the diagonals contain the same value for all i to precompute values when possible and further speed up the algorithm as shown in the following code:

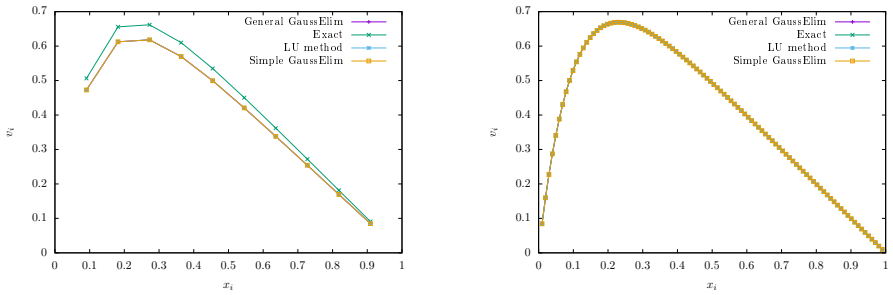
```
! FORWARD SUBSTITUTION
rhs_new(1) = rhs(1)
do i=2, N
     b(i) = b(i)-1.d0/b(i-1)
     rhs_new(i) = rhs(i) + rhs_new(i-1)/b(i-1)
end do

!BACKWARD SUB STEP
v_sol(N) = rhs_new(N)/b(N)
do i = N-1, 1, -1
    v_sol(i) = (rhs_new(i)+v_sol(i+1))/b(i)
end do
```

4. LU Decomposition

In the general case of a system $A\vec{v} = \vec{f}$, we can solve the system for any non-singular, invertible matrix A using the LU decomposition method. This consists of rewriting the system as a matrix product between the lower and upper triangular constituents (L and U respectively) i.e $A = LU$. This decomposition consists of determining

$$L = \begin{bmatrix} 1 & 0 & 0 & \dots & 0 & 0 \\ l_{21} & 1 & 0 & \dots & 0 & 0 \\ l_{31} & l_{32} & 1 & \dots & 0 & 0 \\ & \vdots & & \ddots & \vdots & \\ l_{n-11} & l_{n-12} & l_{n-13} & \dots & 1 & 0 \\ l_{n1} & l_{n2} & l_{n3} & \dots & l_{nn-1} & 1 \end{bmatrix} \tag{11}$$



(A) N=10 (B) N=100

FIGURE 1. Results using the presented methods for $N = 10$ (left) and $N = 100$ (right)

as well as

$$U = \begin{bmatrix} u_{11} & u_{12} & u_{13} & \dots & u_{1n-1} & u_{1n} \\ 0 & u_{22} & u_{23} & \dots & u_{2n-1} & u_{2n} \\ 0 & 0 & u_{33} & \dots & u_{3n-1} & u_{3n} \\ & \vdots & & \ddots & \vdots & \\ 0 & 0 & 0 & \dots & u_{n-1n-1} & u_{n-1n} \\ 0 & 0 & 0 & \dots & 0 & u_{nn} \end{bmatrix} \tag{12}$$

From here the problem is solved by first computing $L\vec{z} = \vec{f}$ using standard methods and then substituting those results into $U\vec{v} = \vec{z}$. Because this method ultimately reduces the problem to a form where Gaussian elimination becomes usable, the scaling should by on the same order as gaussian elimination N^3.

5. Results

5.1. **Convergence.** Comparing to an exact solution for our problem, all algorithms used were found to be sufficiently accurate with a rapidly converging solution with respect to grid size as is shown in 1 with a grid of $N = 100$ being almost exact. The max error with respect to grid size is shown in 2 where we can see the spacing size where machine precision becomes a computational bottleneck.

Another bottleneck found was in the case of the LU factorization method. By construction, the method requires a full matrix to be stored in memory unless the algorithm is modified in some fashion to accomadate sparse matrix storage, and thus the memory requirements become prohibitively large for grid sizes above $N = 1000$.

5.2. **Efficiency.** In the previous section it was shown that all methods are sufficiently accurate for modest grid sizes, thus we now investigate the corresponding computational times in 1. The specialized TDMA method does not improve significantly on the computational time of the general algorithm until the grid size surpasses $N = 100$ while the LU-decomposition method remains significantly slower for all grids.

6. Conclusion

The comparisons shown in this work outline the trade-offs when it comes to choosing numerical algorithms for scientific computing. Gaussian elimination while general, may not be the best choice if a speed alone is of importance in solving the 1D poisson equation. While the LU decomposition method is both slow and

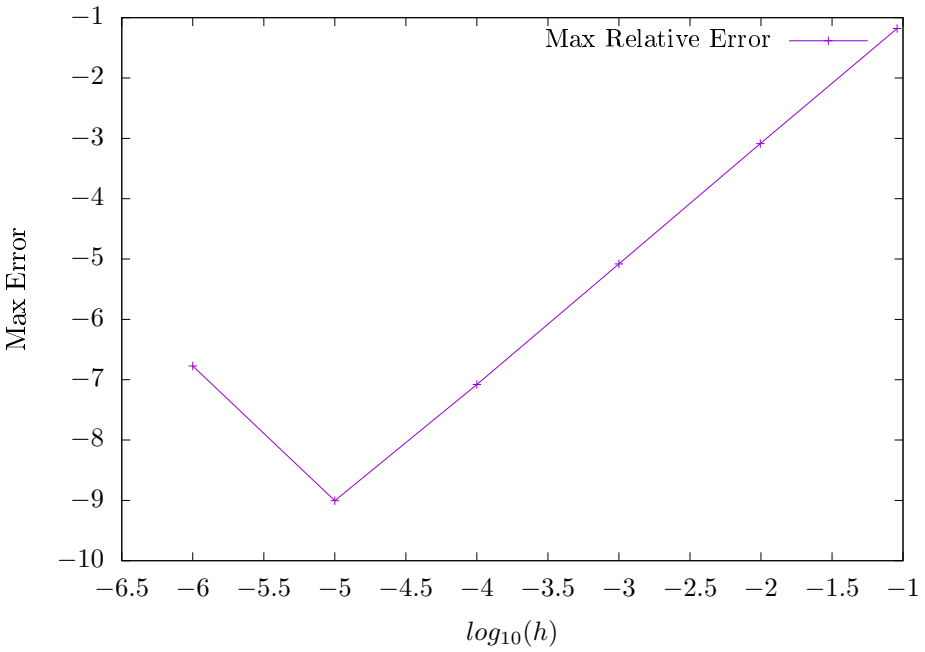


FIGURE 2. Maximum relative error scaling with relative grid spacing

N	TDMA	General	LU
10	3.0×10^{-6}	3.0×10^{-6}	2.0×10^{-3}
100	5.1×10^{-6}	6.5×10^{-6}	0.5×10^{-3}
1000	3.5×10^{-5}	6.1×10^{-5}	5.7×10^{-2}
10000	2.3×10^{-4}	1.2×10^{-5}	21.8

TABLE 1. Computational times for varying grid size and algorithm

memory intensive, it hold the distinct advantage of being completely general and portable. The advantage of an algorithm that can be ported over to another problem trivially is potentially very useful, especially if this leads to faster development and deployment of the code. The addition of human hours spent developing and testing is an important factor when choosing an algorithm to implement given a specific outcome desired. Additionally we have seen the vast improvements possible when refining an algorithm to a problem specific form as in the TDMA method. It is also interesting to note how applicable this trade-off of algorithm generality vs efficiency is to the debate over programming languages for implementation. Generally, an object-oriented programming language can significantly streamline the process of adapting an algorithm to a new problem, at the expense of computational time with respect to a non object oriented language such as FORTRAN.

REFERENCES

[1] William Press. *Numerical recipes in FORTRAN : the art of scientific computing.* Cambridge University Press, Cambridge England New York, NY, USA, 1992.