FYS3150 - Project 1

Daniel Heinesen, Halvard Sutterud, Gunnar Lange

September 16, 2016

Abstract

In this project we will study speed and numerical precision of numerical algorithms. Given a tridiagonal matrix, we used both a general solving algorithm for tridiagonal matrices and a special tailored 'ferrari' method for our specific problem. This is also compared to a cumbersome LU-decomposition of the matrix.

Using an analytic expression for the diagonal elements in our special case, we were able to reduce the total number of FLOPS down to a total of (INSERT N) computations, where N is the number of row and column elements in our matrix.

Contents

Introduction

We will solve the one-dimensional Poisson equation with Dirichlet boundary conditions by reducing it to a set of differential equations on the form of a tridiagonal matrix.

Theoretical model

Discretizing the Poisson equation

Assume that $u(x)$ is a continuous, twice differentiable function, $u(x) \in \mathbb{C}^2$. Using Taylor polynomial to the second degree, a general approximation formula to the second derivative can be derived as:

$$\frac{d^2u}{dx^2} \approx \frac{u(x+h) + u(x-h) - 2u(x)}{h^2} + O(h^2)$$

Where h is a small number. Discretizing $u(x)$ at points $x_1, x_2, ..., x_n$, and introducing the convenient notation $u(x_i) = u_i$, this formula can be rewritten in discrete form as:

$$\frac{d^2u}{dx^2} \approx \frac{u_{i+1} + u_{i-1} - 2u_i}{h^2} + O(h^2)$$

Where h is the distance between the grid points, given by $h = 1/(n+1)$. If one additionally imposes Dirichlet boundary conditions, i.e. that $u_0 = u_{n+1} = 0$, this formula is valid for all x_i, $i \in [1, n]$. Inserting this formula into the Poisson equation with right-hand side $f(x)$, and letting $f(x_i) = f_i$, gives:

$$-\frac{u_{i+1} + u_{i-1} - 2u_i}{h^2} = f_i$$

This can be rephrased as a linear algebra problem, of the form:

$$\mathbf{Au} = h^2\mathbf{f}$$

Where $\mathbf{A}$ is a $n \times n$ given by:

$$\mathbf{A} = \begin{pmatrix} 2 & -1 & 0 & \dots & \dots & 0 \\ -1 & 2 & -1 & 0 & \dots & 0 \\ 0 & -1 & 2 & -1 & 0 & \dots \\ 0 & 0 & -1 & 2 & -1 & \dots \\ \vdots & & & & \ddots & \vdots \\ 0 & & \dots & & -1 & 2 \end{pmatrix}$$

Notice that $\mathbf{A}$ is a tridiagonal matrix. This makes the general solution algorithm much simpler.

Solving a tridiagonal matrix problem

A general tridiagonal matrix problem can be written as:

$$\mathbf{A} = \begin{pmatrix} a_1 & b_1 & 0 & \dots & \dots & 0 \\ c_1 & a_2 & b_2 & 0 & \dots & 0 \\ 0 & c_2 & a_3 & b_3 & 0 & \dots \\ 0 & 0 & c_3 & a_4 & b_4 & \dots \\ \vdots & & & & \ddots & \vdots \\ 0 & & \dots & & c_{n-1} & a_n \end{pmatrix} \begin{pmatrix} u_1 \\ u_2 \\ u_3 \\ u_4 \\ \vdots \\ u_n \end{pmatrix} = \begin{pmatrix} v_1 \\ v_2 \\ v_3 \\ v_4 \\ \vdots \\ v_n \end{pmatrix}$$

Where, in the specific case above, $v_n = f_n/h^2$. This problem may be solved in two steps: a decomposition and forward substitution, and a backward substitution. The goal is to make each column a pivot column. For the forward substitution, it is easiest to first transform the matrix into an upper-diagonal matrix. Thus, a_1 needs to be zero. This can be achieved by subtracting a_1/b_1 times the first row from the second row. This gives:

$$\mathbf{A} = \begin{pmatrix} a_1 & b_1 & 0 & \dots & \dots & 0 \\ 0 & \tilde{a_2} & b_2 & 0 & \dots & 0 \\ 0 & c_2 & a_3 & b_3 & 0 & \dots \\ 0 & 0 & c_3 & a_4 & b_4 & \dots \\ \vdots & & & & \ddots & \vdots \\ 0 & & \dots & & c_{n-1} & a_n \end{pmatrix} \begin{pmatrix} u_1 \\ u_2 \\ u_3 \\ u_4 \\ \vdots \\ u_n \end{pmatrix} = \begin{pmatrix} v_1 \\ \tilde{v_2} \\ v_3 \\ v_4 \\ \vdots \\ v_n \end{pmatrix}$$

Where:

$$\tilde{b_2} = b_2 - \frac{a_1}{b_1}c_1, \quad \tilde{v_2} = v_2 - \frac{a_1}{b_1}v_1$$

Notice how only b change. It is therefore easy to generalize the above formulae to:

$$\tilde{b_{i+1}} = b_{i+1}$$

Counting the number of timesteps

There are three different steps to compute:

$$\tilde{a}_i = a_i - \frac{b_{i-1}c_{i-1}}{\tilde{a}_{i-1}} \tag{1}$$

$$\tilde{f}_i = f_i - \tilde{f}_{i-1}\frac{c_{i-1}}{\tilde{a}_{i-1}} \tag{2}$$

$$u_i = \frac{\tilde{f}_i - b_i u_{i+1}}{\tilde{a}_i} \tag{3}$$