Computational Physics

Project 1

Academic year 2015-2016

Team group:
Alessio Pizzini
Giulio Isacchini
Giovanni Pederiva
Mattia Alberto Ubertini

Sto scrivendo un po di cose molto generali per ora, leggete e in caso aggiungete/corregete In this project we've solved a one-dimensional Poisson equation with Dirichlet boundary conditions. The equation reads: $-u^{''}(x) = f(x)$ with $x[0;1]$ and $u(0) = u(1) = 0$. These kinds of equations can be solved analytically. In our case, $f(x) = 100e^{-10x}$, so that the previous equation has a solution $u_e(x) = 1 - (1 - e^{-10})x - e^{-10x}$. In this work we've solved it numerically and then we've compared our solution with the analytic one. The first step we did has been the discretization of our domain and consequently of the functions in the equation. Then through the 3-point formula for the second derivative we've rewritten the equation in this way: $-\frac{u_{i-1}+ui+1-2u_i}{h^2} = f_i$. At this point the problem becomes a linear equation system with unknown variables the u_i, where u_i means the function u evaluated at the point x_i and the same with f_i. The system can be written as $Au = \tilde{f}$ where $\tilde{f} = h^2 f$. The matrix is tridiagonal, and the tridiagonal system related to it can be written as:

$$a_i v_{i-1} + b_i v_i + c_i v_{i-1} = \tilde{f}_i$$

in our case the coefficients are costant, precisely $b = 2$, $a = c = -1$. We've coded an alghoritm for solving this systems. The alghoritm written is built upon the Gaussian elimination method. The alghoritm consists in a procedure divided in two parts, the first step basically let us to reduce the matrix and consecutively to compute directly the last component u_i, in the second part we use the last component to compute recursively the previous ones.

1 A

In order to solve the differential equation:

$$-u''(x) = f(x), \qquad x \in (0,1), \qquad u(0) = u(1) = 0.$$

you discretize the problem into the numerical equation

$$-\frac{v_{i+1} + v_{i-1} - 2v_i}{h^2} = f_i \qquad \text{for } i = 1, \ldots, n,$$

where $f_i = f(x_i)$. If you define the three quantities:

$$\mathbf{A} = \begin{pmatrix} 2 & -1 & 0 & \dots & \dots & 0 \\ -1 & 2 & -1 & 0 & \dots & \dots \\ 0 & -1 & 2 & -1 & 0 & \dots \\ 0 & \dots & \dots & \dots & \dots & \dots \\ 0 & \dots & & -1 & 2 & -1 \\ 0 & \dots & & 0 & -1 & 2 \end{pmatrix} \quad \mathbf{v} = \begin{pmatrix} v_1 \\ v_2 \\ \dots \\ v_i \\ \dots \\ v_n \end{pmatrix} \quad \tilde{\mathbf{g}} = \begin{pmatrix} \tilde{g}_1 \\ \tilde{g}_2 \\ \dots \\ \tilde{g}_i \\ \dots \\ \tilde{g}_n \end{pmatrix}$$

with $\tilde{g}_i = h^2 f_i$ it is trivial to show that the numerical equation can be rewritten as:

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

In order to solve it with compare three different algorithms:

- **Tridiagonal Matrix Algorithm**, also known as the Thomas algorithm
- Standard Gaussian Elimination
- LU decomposition

Tridiagonal Matriix Algorithm Given a general tridiagonal matrix system:

$$\mathbf{M} = \begin{pmatrix} b_1 & c_1 & 0 & \dots & \dots & 0 \\ a_2 & b_2 & c_3 & 0 & \dots & \dots \\ 0 & a_3 & 2 & b_3 & 0 & \dots \\ 0 & \dots & \dots & \dots & \dots & 0 \\ 0 & \dots & \dots & \dots & \dots & c_{n-1} \\ 0 & \dots & \dots & 0 & a_n & b_n \end{pmatrix} \begin{pmatrix} v_1 \\ v_2 \\ \dots \\ v_i \\ \dots \\ v_n \end{pmatrix} = \begin{pmatrix} \tilde{g}_1 \\ \tilde{g}_2 \\ \dots \\ \tilde{g}_i \\ \dots \\ \tilde{g}_n \end{pmatrix}$$

The algorithm can be summed up in:

- Forward sweep:

$$c'_i = \begin{cases} \dfrac{c_i}{b_i} & i = 1 \\ \dfrac{c_i}{b_i - a_i c'_{i-1}} & i = 2, \dots, n. \end{cases} \qquad \tilde{g}'_i = \begin{cases} \dfrac{\tilde{g}_i}{b_i} & i = 1 \\ \dfrac{\tilde{g}_i - a_i \tilde{g}'_{i-1}}{b_i - a_i c'_{i-1}} & i = 2, \dots, n. \end{cases}$$