Division of Scientific Computing
Department of Information Technology
Uppsala University

NGSSC: Numerical Methods in Scientific Computing
Spring 2008

Project - Solving the Heat equation in 2D

Aim of the project

The major aim of the project is to apply some iterative solution methods and preconditioners when solving linear systems of equations as arising from discretizations of partial differential equations.

Introduction: The problem

Consider the time-dependent heat equation in two dimensions

$$u_t = \Delta u + f(u,t),\ u = u(\mathbf{x},t), \mathbf{x} \in \Omega = [0,1]^2,$$

where $\Delta u = \sum_{i=1}^{2} \frac{\partial^2 u}{\partial x_i^2}$.
For the test example we let $f = 1$, $u(\mathbf{x},t)\ |_{\mathbf{x}\in\partial\Omega} = 0$ and as an initial condition we choose the discontinuous function

$$u(\mathbf{x},0) = \begin{cases} 1 & \text{if } \sqrt{(x_1-0.5)^2+(x_2-0.5)^2} < 0.2, \\ 0 & \text{elsewhere.} \end{cases}$$

We discretize the problem in two steps.

(i) **Space discretization**
Issues related to space discretization fall out of the scope of this project and all related matrices are readily provided. A short description follows.

The space discretization is done using the Finite Element method. The spatial domain is discretized with triangular elements and an example of the spatial mesh is provided in Figure 1.

As a result we obtain the so-called *semi-discrete* problem which constitutes a system of ODEs and has the following form:

$$M\frac{\partial \mathbf{U}(t)}{\partial t} = K\mathbf{U}(t) + \mathbf{F}(t), \tag{1}$$

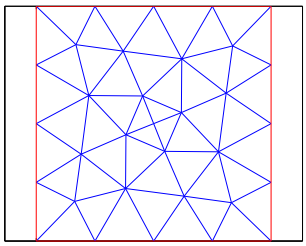

Figure 1: A coarse space discretization

where $\mathbf{U}(t)$ is the unknown solution, discretizes in space and being now a function only of time t. Similarly, $\mathbf{F}(t)$ is the discrete counterpart of $f(u,t)$.

The matrices M and K are the corresponding standard mass and stiffness finite element matrices, which are symmetric positive definite and symmetric negative definite correspondingly.

(ii) **Time discretization**
The time discretization will be performed using the so-called θ method. The fully discretized system reads as follows:

$$(M - \theta\delta_t K)\mathbf{U}^{k+1} = (M + (1-\theta)\delta_t K)\mathbf{U}^k + \delta_t(\theta\mathbf{F}^{k+1} + (1-\theta)\mathbf{F}^k), \quad 0 \leq \theta \leq 1, \tag{2}$$

where $\mathbf{U}^{k+1}$ is the solution on time level $k+1$ to be computed and it is assumed that we already have obtained $\mathbf{U}^k$.

As we can see, $\theta = 0$ corresponds to explicit Euler method, $\theta = 1$ corresponds to implicit Euler method and $\theta = 0.5$ corresponds to Crank-Nicolson's scheme (also referred to as the trapezoidal method).

Regarding stability of the above discretization scheme, theory says that for $\theta \in [0, 0.5)$ we have a conditionally stable scheme, where the time step δ_t has to be related to the space discretisazion parameter (h), and for $\theta \in [0.5, 1]$ the time discretization scheme is unconditionally stable.

It is also known that in order to balance in a best way the contributions of the space and time discretization errors into the global discretization error, θ should be chosen as $\theta = 0.5 + \xi$ for some small ξ. Observe that $\theta = 0.5$ might lead to unphysical oscillations in the computed solution, in particular in the beginning of the simulation process, since the initial solution is discontinuous.

Tasks

The task is to test a preconditioned iterative method to solve the so-arising linear systems of equations. Observe that in the case when the Finite Element method is used, even if we use explicit time discretization schemes, we always have to solve a linear system of equations.

Below we provide a description of the preconditioner for the matrix $A = M - \theta\delta_t K$ which has to be implemented. We assume that the matrix A has a two-by-two block structure

$$A = \begin{bmatrix} A_{11} & A_{12} \\ A_{21} & A_{22} \end{bmatrix}$$

It is well-known (and can be checked with straightforward computation) that A admits the following exact factorization:

$$A = \begin{bmatrix} A_{11} & 0 \\ A_{21} & S_A \end{bmatrix} \begin{bmatrix} I_1 & A_{11}^{-1}A_{12} \\ 0 & I_2 \end{bmatrix} \tag{3}$$

where I_1 and I_2 are identity matrices of a proper order and S_A is the corresponding Schur complement matrix $S_A = A_{22} - A_{21}A_{11}^{-1}A_{12}$.
In practice, even if A is sparse, its Schur complement can be a dense matrix and we do not want to compute it explicitly.
For now we assume that we are able to construct an approximation S of S_A, such that S is sparse and it is also spectrally equivalent to S_A, i.e.,

$$\kappa(S^{-1}S_A) = O(1).$$

In other words, we assume that we have a very good sparse approximation to a in general dense matrix. Then we consider the following preconditioner to A, based on the exact factorization (3):

$$C = \begin{bmatrix} A_{11} & 0 \\ A_{21} & S \end{bmatrix} \begin{bmatrix} I_1 & Z_{12} \\ 0 & I_2 \end{bmatrix} \tag{4}$$

where we also assume that Z_{12} is a reasonably good approximation of the matrix product $A_{11}^{-1}A_{12}$.

1. Write a MATLAB code which implements the following algorithm:
   For a given $\theta, \delta_t, M, K, A_{11}, A_{21}, S, Z_{12}, \mathbf{F}$ perform ten timesteps, based on the discrete problem (2).

   During each timestep solve the corresponding matrix A using the PCG method. To this end, use the `pcg` function from MATLAB without and with preconditioning. For the latter case write your own function to implement the preconditioner C from (4).

   The way to do it in MATLAB is as follows

```
[u_next,flag,relres,iter,resvec] = pcg(A,rhs,tol,maxit,@blkprec,...
                                   [],[],A11,A21,S,Z12)
```

   The MATLAB function

```
function w = blkprec(v,A11,A21,S,Z12)
```

   should implement the solution of the system $C\mathbf{w} = \mathbf{v}$. Use MATLAB backslash operator to solve systems with A_{11} and S.

2. Observe the pcg iterations per timestep, check the convergence history `resvec`.

3. Compare the performance of the unpreconditioned and the preconditioned CG methods. Estimate the computational complexity and give your reasonings on whether it is relevant to use an involved preconditioner in this case.

4. Check the spectrum of A and how it is transformed after preconditioning:

```
EA = sort(eig(full(A)));
EC = sort(eig(full(A),full(C)));
```

   for some small matrix sizes.

5. Chech how well S approximates S_A and include a comment on that.

Remark 1: In order to run the unpreconditioned CG, it suffices to use only the matrices M and K, which do not depend on the choice of δ_t and θ. Therefore, such experiments can be done with arbitrarily chosen δ_t (and, of course θ).
For the preconditioned CG, the blocks $A11, A21, S, Z12$ are constructed for a particular values of δ_t and θ, which are included in the corresponding data file. For the cases where the blocks of the preconditioner are precomputed, $\delta_t = h_{max}$, where h_max measures the largest triangle in the discretization mesh.
Remark 2: Use 10^{-4} as a stopping criterium for the CG method.

Writing a report on the results

The report has to have the following issues covered:

1. A brief problem description

2. Numerical experiments

   Describe the experiments as consistently as possible. Include some relevant information on your choice - iteration counts, plots of residual histories, condition numbers (as a function of the size) etc. Is there a noticeable difference in the quality of the solution with respect to the parameter θ? How does the number of iterations grow with the problem size? Add a discussion on the suggested preconditioner - is it good (robust, computationally feasible etc.) or not and why. Comment on the relation between numerical stability (robust discretization scheme) and robust linear system solver.

   A listing of the `MATLAB` program code has to be attached to the report.

3. Conclusions

Provided data for the experiments

All the input matrices are to be loaded from MATLAB `.mat` files with the following name convention:

`Heat_in_N_theta_dt`

where `N` is the size of the matrix A, `THETA` is the corresponding value of θ and `dt` is the timestep. The latter is needed since the corresponding matrices A_{11}, A_{21}, S and Z_{12} do depend on both θ and δ_t.
Each data file contains the following variables:

```
M, K, A11, A21, S, Z12, FF, theta, dt, u0
lvl_total, Node, Face_Node
```

Some of the arrays are provided in order to enable you to plot the solution, which for example could be done as follows:

```
clf
eval(['u0_nr(O' int2str(lvl_total) ',1) = u0;'])  % Re-reorder
trisurf(Face_Node(:,:,2)',Node(1,:),Node(2,:),u0_nr,'facecolor','interp')
disp('Initial condition')
```

and

```
eval(['u_nr(O' int2str(lvl_total) ',1) = u_next;'])  % Re-reorder
clf
trisurf(Face_Node(:,:,2)',Node(1,:),Node(2,:),u_nr,'facecolor','interp')
```

In this particular case the function $f(u,t)$ is constant and is contained in the vector FF.

File name	File size	Remark
Heat_in_417_0.98438_0.125.mat	161736	$\theta = 1 - h_{max}^2$, $\delta_t = h_{max}$
Heat_in_1601_0.99609_0.0625.mat	665448	
Heat_in_6273_0.99902_0.03125.mat	2704752	
Heat_in_417_0.75_0.125.mat	161736	$\theta = 0.5 + 2h_{max}$, $\delta_t = h_{max}$
Heat_in_1601_0.625_0.0625.mat	665448	
Heat_in_6273_0.5625_0.03125.mat	2704752	
Heat_in_417_0_0.00024414.mat	161728	$\theta = 0$, $\delta_t = h_{max}^4$

Good luck!