Course 19.155120.0 "Scientific Computing" Project description "Efficient iterative solution strategies in implicit time integration"

M.A. Botchev

Read the whole project description before starting to program

In this project we solve the following initial value problem for the system of ordinary differential equations (ODEs)

$$\boldsymbol{u}' = A\boldsymbol{u} + \boldsymbol{f}(t), \qquad \boldsymbol{u}(0) = \boldsymbol{u}^0 \in \mathbb{R}^M, \tag{1}$$

where $\boldsymbol{u}(t)$ is the unknown vector function, the $M \times M$ matrix A and the M-vector function $\boldsymbol{f}(t)$ are given. We solve (1) by the implicit midpoint rule scheme:

$$\frac{\boldsymbol{u}^{n+1} - \boldsymbol{u}^n}{\tau} = A\frac{\boldsymbol{u}^{n+1} + \boldsymbol{u}^n}{2} + \boldsymbol{f}(t_{n+1/2}), \tag{2}$$

where $\tau > 0$ is the time step size and $\boldsymbol{u}^n$ is an approximation to the solution $\boldsymbol{u}(t_n)$ at the time step n, $t_n = n\tau$. In the scheme the next time step solution $\boldsymbol{u}^{n+1}$ can be obtained by solving a linear system with the matrix $I - \frac{\tau}{2}A$. Since the solution $\boldsymbol{u}^{n+1}$ is not readily available, the scheme is implicit.

ODE system is obtained from a spatial discretization of an initial-boundary value problem (IBVP) for a three-dimensional partial differential equation. For example, you can take the problem and discretization described in Appendix A below. You can also take another matrix A but the matrix you choose should be large enough (5000 or larger).

The aim of the project is to compare different iterative solution strategies for the implicit scheme (2). Every time step the linear system in $\boldsymbol{u}^{n+1}$ has to be solved. One wants to compute an accurate approximate solution $\boldsymbol{u}^N \approx \boldsymbol{u}(T)$ at the final time $T = N\tau$ with as little as possible computational work.

- We will measure the computational work as the number of (eventually preconditioned) matrix-vector multiplications. The accuracy of $\boldsymbol{u}^N$ is assessed by the error defined as

$$\text{error} \equiv \left\| \frac{\boldsymbol{u}^N - \boldsymbol{u}^N_{\text{exact}}}{\boldsymbol{u}^N_{\text{exact}} + \epsilon} \right\|_\infty,$$

  where $\boldsymbol{u}^N_{\text{exact}}$ is the reference solution, ϵ is the machine precision (see "`help eps`" in MATLAB) and the division is done elementwise. The reference solution is to be computed with the MATLAB built-in time integration scheme `ode23` and this may take significant computing time. Do not change the parameters `AbsTol` and `RelTol`) and take care that the MATLAB solver does not return solution at all the intermediate moments of time (you only need the final time solution $\boldsymbol{u}^N_{\text{exact}} = \boldsymbol{u}(T)$).

- We are interested in the step sizes τ for which the error is within the range $[0.01, 0.1]$. The final time T should be chosen such that with the largest possible τ at least 100 time steps have to be done to reach T.
- Implement first the time integration scheme in such a way that the linear system in $\boldsymbol{u}^{n+1}$ is solved very accurately (use the MATLAB default stopping criterion $\|\boldsymbol{r}_k\|/\|\boldsymbol{r}_0\| \leqslant 10^{-6}$). The error delivered by the scheme should be of second order, i.e. it has to be approximately factor four smaller when τ is halved.
- Now try to experiment with the stopping criterion in the iterative solver to reduce the computational work. Can we relax the stopping criterion? How much? Perhaps it is a good idea to perform a fixed number of iterations in the linear solver at each time step? Make plots showing the error versus τ and the error versus the computational work for different iterative strategies.
- Incorporate simple preconditioning (IC/ILU, SSOR). Try to use different iterative solvers. What is the most efficient strategy?

Write a report of at most 10 pages on your investigations. Include your codes into the report as appendices and email your report and all your Matlab files to me.

A An example of IBVP and its discretization

You can take the following IBVP:

$$
\begin{aligned}
&u_t = u_{xx} + u_{yy} + u_{zz} + f(x,y,z,t),\\
&(x,y,z) \in \Omega \equiv [0,1]\times[0,1]\times[0,1], \quad u(x,y,z,0) = \text{given function in } x,y,z,\\
&\left.\frac{\partial u}{\partial n}\right|_{\partial\Omega} = 0,
\end{aligned}
$$

and reduce it to (1) by the standard second order finite difference approximation

$$
u_{xx}(x_i,y_j,z_k,t) \approx \frac{u_{i-1,j,k} - 2u_{i,j,k} + u_{i+1,j,k}}{h_x^2},
$$

and similarly for u_{yy} and u_{zz}. Here $u_{i,j,k} \equiv u(x_i,y_j,z_k,t)$ are the (approximate) values of the unknown function u at the nodes (x_i,y_j,z_k) of the spatial mesh which can be taken as

$$
x_i = (i-0.5)h_x, \qquad y_j = (j-0.5)h_y, \qquad z_k = (k-0.5)h_z,
$$

with h_x, h_y and h_z being the mesh sizes. Since the mesh is shifted by the half step size, we can approximate the derivative in the boundary conditions by the second order central differences:

$$
\left.\frac{\partial u}{\partial n}\right|_{\partial\Omega,x=0} \approx \frac{u_{1,j,k} - u_{0,j,k}}{h_x},
$$

and similarly for $x = 1$ and the other parts of $\partial\Omega$.

For this project, a spatial mesh with, say, $20 \times 20 \times 20$ nodes can be taken.