COMP5930M - Scientific Computing

Coursework 2

November 16, 2020

Deadline

10:00, Monday 7th December

Task

The numbered sections of this document describe problems which are modelled by partial differential equations. A numerical model is specified which leads to a nonlinear system of equations. You will use one or more of the algorithms we have covered in the module to produce a numerical solution.

Matlab scripts referred to in this document can be downloaded from Minerva under Learning Resources / Coursework. Matlab implementations of some algorithms have been provided as part of the module but you can implement your solutions in any other language if you prefer.

You should submit your answers as a single PDF document via Minerva before the stated deadline. MATLAB code submitted as part of your answers should be included in the document. MATLAB functions should include appropriate `help` information that describe the purpose and use of that function.

Standard late penalties apply for work submitted after the deadline.

Disclaimer

This is intended as an individual piece of work and, while discussion of the work is encouraged, plagiarism of writing or code in any form is strictly prohibited.

1. A one-dimensional PDE: Nonlinear parabolic equation

[22 marks total]

A nonlinear parabolic equation is: find $u(x,t)$ such that

$$\frac{\partial u}{\partial t} = \epsilon \frac{\partial^2 u}{\partial x^2} + \alpha \left(\frac{\partial u}{\partial x} \right)^2 \qquad (1)$$

in the spatial interval $x \in (0,1)$ and time domain $t > 0$.
Here ϵ and α are known, positive, constants.

Boundary conditions are specified as $u(0) = 0$ and $u(1) = 1$.

Initial conditions are specified at $t = 0$ as $u(x,0) = x$.

We numerically approximate (1) using the method of lines on a uniform spatial grid with m nodes on the interval $[0,1]$ with grid spacing h, and a fixed time step of Δt.

Code for this problem can be downloaded from Minerva and is in the **`Q1/`** folder.

(a) Find the fully discrete formulation for (1) using the *central finite difference formulas*

$$u(x_i) \approx u_i, \quad \frac{\partial u}{\partial x}(x_i) \approx \frac{u_{i+1} - u_{i-1}}{2h}, \quad \frac{\partial^2 u}{\partial x^2}(x_i) \approx \frac{u_{i+1} - 2u_i + u_{i-1}}{h^2}$$

in space and the *implicit Euler method* in time.
Define the nonlinear system $\mathbf{F}(\mathbf{U}) = \mathbf{0}$ that needs to be solved at each time step to obtain a numerical solution of the PDE (1).

(b) Solve the numerical problem for $N = 81$ equations on the time interval $t \in [0,2]$ with a different number of time steps: $N_T = 20, 40, 80, 160$. The main program is executed using the `runModel.m` function. The nonlinear system is implemented in the Matlab function `residual.m` such that Newton's method can be applied to find a numerical solution. The Jacobian can be computed using `fdJacobian.m` and `linearSolve.m` should be used to solve the linear system at each iteration. Note: `linearSolve.m` uses dense linear algebra.

Create a table that shows for each different N_T:

i. the total computational time taken, T
ii. the total number of Newton iterations, S
iii. the average time spent per Newton iteration, $t_S = T/S$

(c) Solve the same problem for different problem sizes: $N = 81, 161, 321, 641$ (choose m appropriately for each case). Use $N_T = 40$ time steps. Create a table that shows for each different N:

i. the total computational time taken, T
ii. the total number of Newton iterations, S
iii. the average time spent per Newton iteration, $t_S = T/S$

Estimate the algorithmic cost of one Newton iteration as a function of the number of equations N from this simulation data. Does your observation match the theoretical cost of the algorithms you have used?

Hint: Take the values N and the measured $t_S(N)$ and fit a curve of the form $t_S = CN^P$ by taking logarithms in both sides of the equation to arrive at

$$\log t_S = \log C + P \log N,$$

then use `polyfit( log(N), log(tS), 1 )` to find the P that best fits your data.

(d) Derive the exact expression for row i of the Jacobian for this problem.

How many nonzero elements are on each row of the Jacobian matrix?

(e) Repeat the timing experiment (c), but using now the Thomas algorithm `sparseThomas.m` to solve the linear system and the tridiagonal implementation of the numerical Jacobian computation `tridiagonalJacobian.m`. You will need to modify the code to call `newtonAlgorithm.m` appropriately.

Create a table that shows for each different N:

i. the total computational time taken, T
ii. the total number of Newton iterations, S
iii. the average time spent per Newton iteration, $t_S = T/S$

Estimate the algorithmic cost of one Newton iteration as a function of the number of equations N from this set of simulations. Does your observation match the theoretical cost of the algorithms you have used?

Hint: Take the values N and the measured $t_S(N)$ and fit a curve of the form $t_S = CN^P$ by taking logarithms in both sides of the equation to arrive at

$$\log t_S = \log C + P \log N,$$

then use `polyfit( log(N), log(tS), 1 )` to find the P that best fits your data.

2. A three-dimensional PDE: Nonlinear diffusion

[18 marks total]

Consider the following PDE for $u(x,y,z)$:

$$\frac{\partial}{\partial x}\left((1+u^2)\frac{\partial u}{\partial x}\right)+\frac{\partial}{\partial y}\left((1+u^2)\frac{\partial u}{\partial y}\right)+\frac{\partial}{\partial z}\left((1+u^2)\frac{\partial u}{\partial z}\right) = g(x,y,z), \tag{2}$$

defined on $[x,y,z]\in[-10,10]^3$ with $u(x,y,z)=0$ on the boundary of the domain, and some known function $g(x,y,z)$ that does not depend on u.

Using a finite difference approximation to the PDE (2) on a uniform grid, with grid size h and n nodes in each coordinate direction, the nonlinear equation $F_{ijk}=0$ to be solved at an internal node may be written as

$$\begin{aligned}
F_{i,j,k} &= \frac{1}{h^2}\left[\left(1+\left(\frac{u_{i+1,j,k}+u_{i,j,k}}{2}\right)^2\right)(u_{i+1,j,k}-u_{i,j,k})-\left(1+\left(\frac{u_{i,j,k}+u_{i-1,j,k}}{2}\right)^2\right)(u_{i,j,k}-u_{i-1,j,k})\right.\\
&+\left(1+\left(\frac{u_{i,j+1,k}+u_{i,j,k}}{2}\right)^2\right)(u_{i,j+1,k}-u_{i,j,k})-\left(1+\left(\frac{u_{i,j,k}+u_{i,j-1,k}}{2}\right)^2\right)(u_{i,j,k}-u_{i,j-1,k})\\
&+\left.\left(1+\left(\frac{u_{i,j,k+1}+u_{i,j,k}}{2}\right)^2\right)(u_{i,j,k+1}-u_{i,j,k})-\left(1+\left(\frac{u_{i,j,k}+u_{i,j,k-1}}{2}\right)^2\right)(u_{i,j,k}-u_{i,j,k-1})\right]\\
&- g(x_{i,j,k},y_{i,j,k},z_{i,j,k}) = 0
\end{aligned}\tag{3}$$

Code for this problem can be downloaded from Minerva and is in the **Q2/** folder. The Matlab function `runFDM.m` controls the numerical simulation of the discrete nonlinear system (3).

(a) Run the model using `runFDM.m` with grid dimension $m=5$, 8 and 11 for $tol=10^{-8}$ and initial guess $u(x,0)=0$ for all x. Create a table that shows for each different N:

i. the total number of unknowns, N
ii. the total computational time taken, T
iii. the total number of Newton iterations, S
iv. the average time spent per Newton iteration, $t_S=T/S$

Estimate the algorithmic cost of a single Newton iteration in terms of the size of the nonlinear system N, i.e. $T=\mathcal{O}(N^p)$. How does it compare to the theoretical worst-case complexity?

Hint: Take the values N and the measured $t_S(N)$ and fit a curve of the form $t_S=CN^P$ by taking logarithms in both sides of the equation to arrive at

$$\log t_S=\log C+P\log N,$$

then use `polyfit( log(N), log(tS), 1 )` to find the P that best fits your data.

(b) Replace in `runFDM.m` the call to `linearSolve.m` function that uses dense linear algebra with the alternative `matlabLinearSolve.m` that uses sparse linear algebra.

Solve the problem for $m=5$ with both dense linear algebra and sparse linear algebra, and show that the numerical solution does not change when we change the solver by measuring the norm of the difference between the two solutions (Hint: Pay attention to the outputs of `runFDM.m` when doing your comparison).

(c) Replace in `runFDM.m` the call to a dense Jacobian implementation `fnJacobian.m` with a call to a sparse implementation, `buildJacobian.m`. You will need to uncomment the following function calls before calling the Newton algorithm:

```
[ A, nz ] = sparsePattern3DFD( m ); % 3d FDM sparsity pattern
[ loc, nf ] = sparseJac( A,N );     % function calls required to evaluate J
```

i. Visualise the sparsity pattern of `A` using the command `spy` in the case $m = 8$ and include the plot in your report. Note: The matrix `A` has the same sparsity pattern as the Jacobian for this problem.

ii. What is the *bandwidth* of the matrix `A` in the case $m = 8$? The bandwidth of a matrix `A` is defined as the smallest number k s.t.

$$a_{i,j} = 0 \text{ for all } |i - j| > k,$$

i.e. the smallest distance from the diagonal after which all elements will be 0.

(d) Use the command `lu` to compute the LU-factorisation of `A` in the case $m = 8$. What are the number of non-zero elements L and U, respectively?

Visualise the sparsity pattern of the LU-factors using the command `spy` and include the plot in your report. Can you observe something unexpected about the matrix L? How can you explain this perceived discrepancy?

Use these matrices as an example to explain what we mean by *fill-in* of the LU-factors.

(e) Solve the problem using the sparse implementation for $m = 15$, 19 and 23. Create a table that shows for each different N:

i. the total number of unknowns, N
ii. the total computational time taken, T
iii. the total number of Newton iterations, S
iv. the average time spent per Newton iteration, $t_S = T/S$

Estimate the algorithmic cost of this new sparsity-based implementation using the timings of the numerical experiments. Compare the results with those of 2(a).

Hint: Take the values N and the measured $t_S(N)$ and fit a curve of the form $t_S = CN^P$ by taking logarithms in both sides of the equation to arrive at

$$\log t_S = \log C + P \log N,$$

then use `polyfit( log(N), log(tS), 1 )` to find the P that best fits your data.

Learning objectives

- Formulation of sparse nonlinear systems of equations from discretised PDE models.
- Measuring efficiency of algorithms for large nonlinear systems.
- Efficient implementation for sparse nonlinear systems.

Mark scheme

This piece of work is worth 20% of the final module grade.

There are 40 marks in total.

1. 1d PDE [22 marks total]
   (a) Formulation of the discrete problem [5 marks]
   (b) Simulation and timings for different N_T [3 marks]
   (c) Timings and analysis of complexity, general case [5 marks]
   (d) Jacobian structure [4 marks]
   (e) Timings and analysis of complexity, tridiagonal case [5 marks]
2. 3d PDE [18 marks total]
   (a) Simulations with dense version of the solver [5 marks]
   (b) Comparison of dense and sparse solver outputs [2 marks]
   (c) Analysis of the sparsity of the Jacobian matrix [2 marks]
   (d) Analysis of the LU factors of the Jacobian [4 marks]
   (e) Simulations with sparse version of the solver [5 marks]