COMP5930M - Scientific Computation

Coursework 2

Daolin Sheng ml192ds

---

1. A one-dimensional PDE: Nonlinear parabolic equation

(a), Formulation of the discrete problem

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

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

- The central difference formulas in space:

$$u(x_i) \approx u_i$$

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

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

- The continuous problem is reduced to a semi-discrete system of ODEs for

$$\frac{\partial u}{\partial t} \equiv \dot{\boldsymbol{u}}(t) = \boldsymbol{f}(\boldsymbol{u})$$

- Implicit Euler

$$\frac{\boldsymbol{u}^{k+1} - \boldsymbol{u}^{k}}{\Delta t} = \boldsymbol{f}(\boldsymbol{u}^{k+1})$$

- Use implicit Euler. at node i of the gird:

$$\frac{u_i^{k+1} - u_i^k}{\Delta t} = \epsilon \frac{u_{i+1}^{k+1} - 2u_i^{k+1} + u_{i-1}^{k+1}}{h^2} + \alpha(\frac{u_{i+1}^{k+1} - u_{i-1}^{k+1}}{2h})^2$$

$$F_i(u_{i-1}^{k+1}, u_i^{k+1}, u_{i+1}^{k+1}) = F_i(U) = \frac{u_i^{k+1} - u_i^k}{\Delta t} - \epsilon \frac{u_{i+1}^{k+1} - 2u_i^{k+1} + u_{i-1}^{k+1}}{h^2} + \alpha(\frac{u_{i+1}^{k+1} - u_{i-1}^{k+1}}{2h})^2 = 0$$

- So, the Full discrete nonlinear system $F(U) = 0$ for the unknown ${u_i}^{k+1}$, i = 2, …, n-1, equations are formed at each internal node i = 2, …, n -1. Depends on previous u^k ,time step Δt and gird size h. Each equation F_i depends on three neighboring nodes U_{i-1}, U_i, U_{i+1} through the FDM approximation.
- Initial conditions are specified at $t = 0$ as u(x, 0) = x.

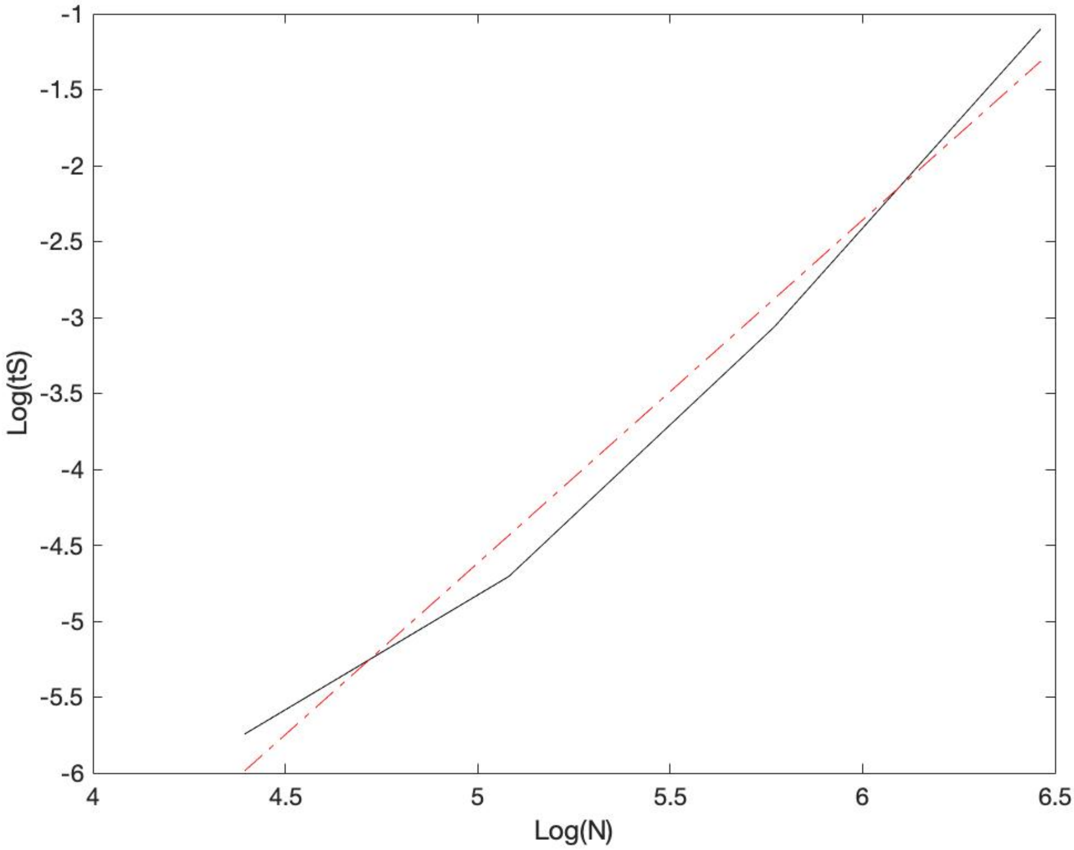
(b), Simulation and timings for different N_T

Different N_T	T (seconds)	Total Iterations (S)	Average time t_S = T/S (seconds)
20	0.189367	59	0.003209
40	0.417477	104	0.004014
80	0.743755	196	0.003794
160	1.293817	369	0.003506

(c), Timings and analysis of complexity, general case

Different N	T (seconds)	Total Iterations (S)	Average time t_S = T/S (seconds)
81	0.333213	104	0.003204
161	0.978038	108	0.009056
321	5.768211	123	0.046896
641	43.318342	130	0.333218

- $t_S = CN^P$ and $log(t_S) = logC + Plog(N)$
- By fitting the N values and t_S, can get $log(t_S) = -17.45 + 2.6log(N)$ approximately.
- So get the cost of one New iteration with the N, have the equation: $t_S = e^{-17.45}N^{2.6}$, and the cost of the algorithms is match computation cost of Jacobian's $O(n^3)$.



(d), Jacobian structure

- Finite differences + Implicit Euler. At node i of the gird:

$$F_i = \frac{U_i - u_i^k}{\Delta t} - \epsilon \frac{U_{i+1} - 2U_i + U_{i-1}}{h^2} + \alpha (\frac{U_{i+1} - U_{i-1}}{2h})^2 = 0$$

- For this problem we can analytically the Jacobian matrix:

$$\frac{\partial F_i}{\partial U_{i-1}} = -\frac{\epsilon}{h^2} + \frac{\alpha (U_{i-1} - U_{i+1})}{2h^2}$$

$$\frac{\partial F_i}{\partial U_i} = \frac{1}{\Delta t} + \frac{2\epsilon}{h^2}$$

$$\frac{\partial F_i}{\partial U_{i+1}} = -\frac{\epsilon}{h^2} + \frac{\alpha (U_{i+1} - U_{i-1})}{2h^2}$$

- **These are the only nonzero elements of the *i'th* row.**

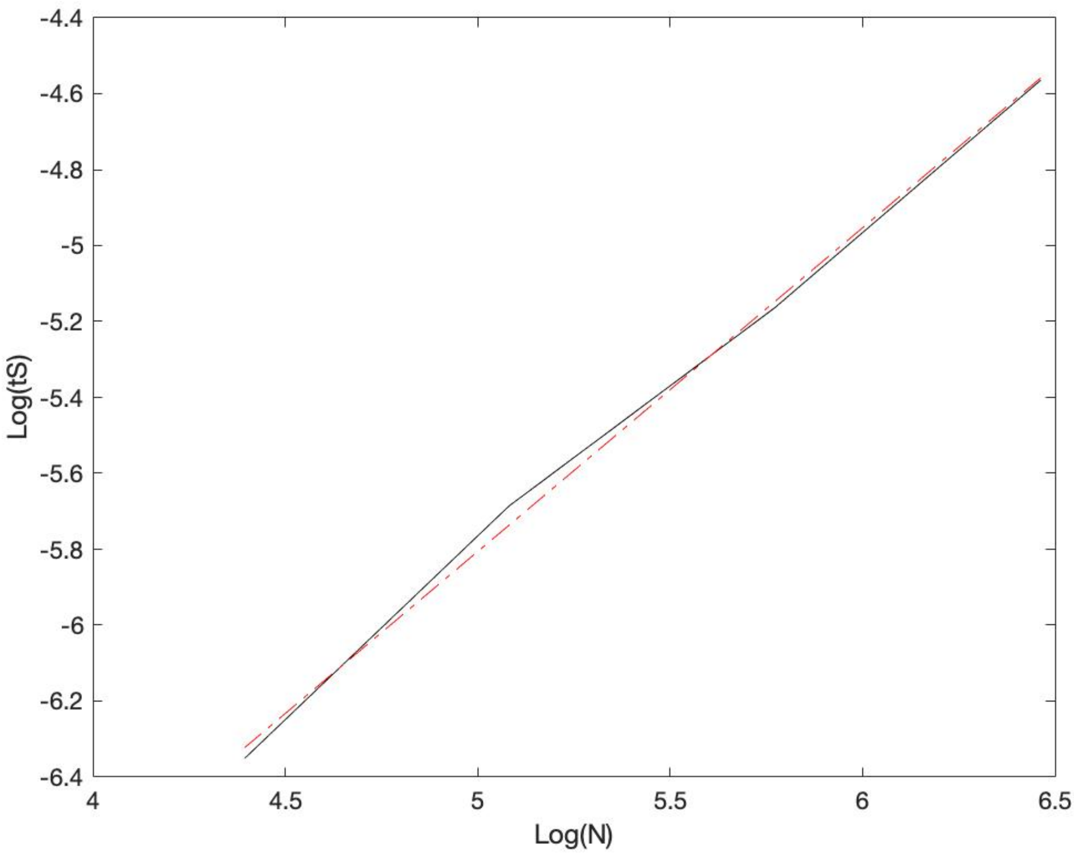
(e), Timings and analysis of complexity, tridiagonal case

- Using the Thomas algorithm and the tridiagonal implementation of numerical Jacobian computation.

Different N	T (seconds)	Total Iterations (S)	Average time t_S = T/S (seconds)
81	0.181544	104	0.001745

161	0.386499	108	0.003393
321	0.703420	123	0.005718
641	1.354466	130	0.010418

- $t_S = CN^P$ and $log(t_S) = logC + Plog(N)$
- By fitting the N values and t_S, can get $log(t_S) = -10.75 + log(N)$ approximately.
- So get the cost of one New iteration with the N, have the equation: $t_S = e^{-10.75}N^1$, and the cost of the algorithms is match the theoretical cost $O(n)$.



2. A three-dimensional PDE: Nonlinear diffusion

(a), Simulations with dense version of the solver

- PDE for $u(x, y, z)$, and $[x, y, z] \in [-10, 10]^3$ 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.

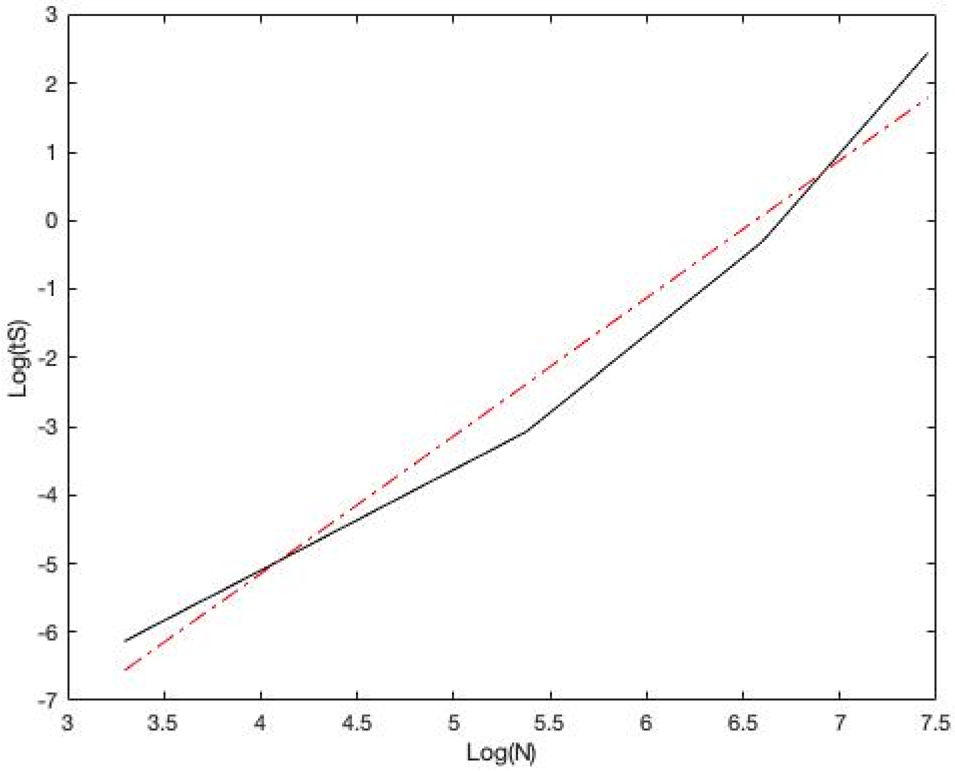
$$\frac{\partial}{\partial x}((1+u^2)\frac{\partial u}{\partial x}) + \frac{\partial}{\partial y}((1+u^2)\frac{\partial u}{\partial y}) + \frac{\partial}{\partial z}((1+u^2)\frac{\partial u}{\partial z}) = g(x, y, z)$$

- With gird dimension m = 5, 8 and 11 for $tol = 10^{-8}$

Different N	Time Taken (seconds)	Total Iterations (S)	Average time t_S = T/S (seconds)

27	0.008670	4	0.002167
216	0.322518	7	0.046074
729	5.044202	7	0.720600

- $t_S = CN^P$ and $log(t_S) = logC + Plog(N)$
- By fitting the N values and t_S, can get $log(t_S) = -13.2 + 2log(N)$ approximately.
- Get the cost of one New iteration with the N, have the equation: $t_S = e^{-13.2}N^2$, since using the spare storage schemas in 2d systems, and just using 2 x nnz vectors, So the problem is how to find out the sparsity pattern of the Jacobian. the cost turn into to $O(n^2)$, and the cost of the algorithms is match the theoretical cost $O(n^2)$.

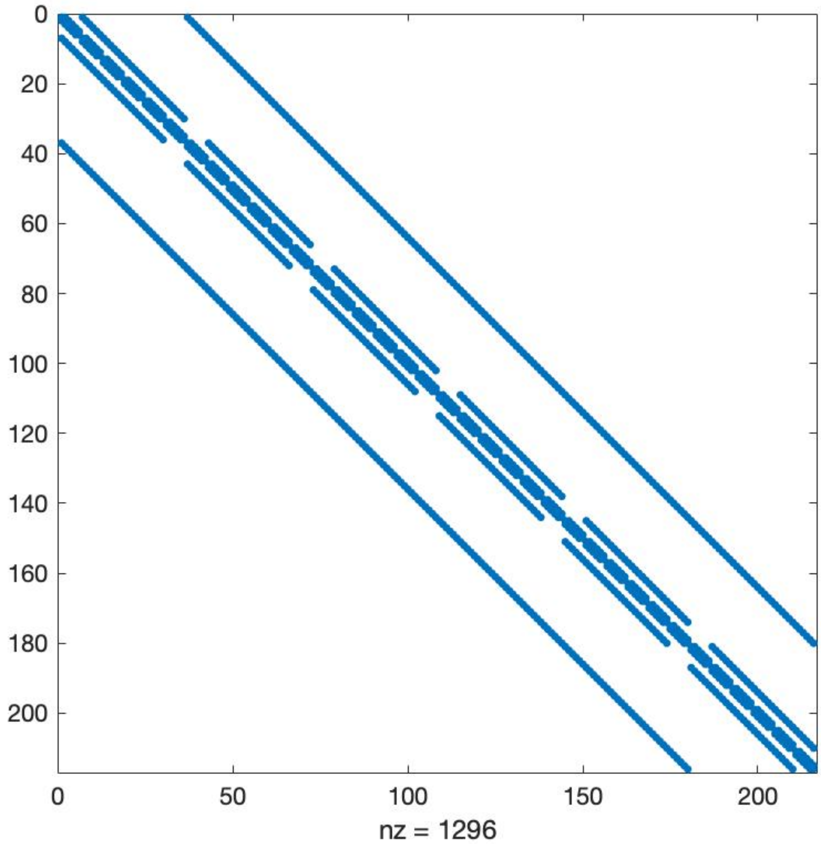


(b), Comparison of dense and sparse solver outputs

- When m = 5, using the dense linear algebra and sparse linear algebra.
- This two methods have the same numerical solutions, but the cost time of using dense linear algebra is longer than the cost time of using spares linear algebra.

(c), Analysis of the sparsity of the Jacobian matrix

(i), The graph of the sparsity pattern A blow:

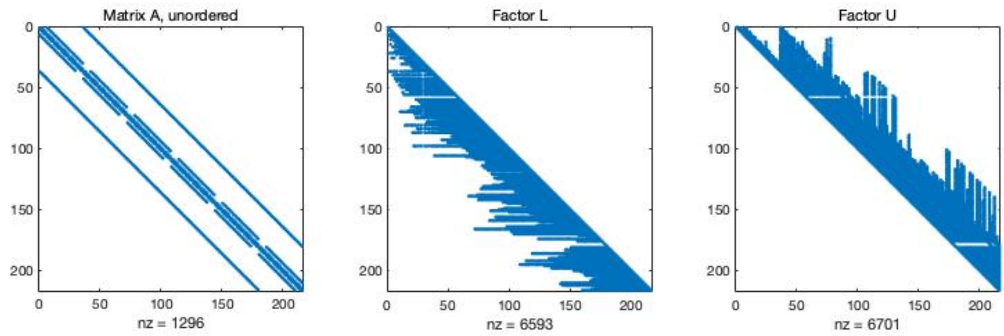


(ii),, base on the bandwidth of a matrix **A** is defined as the smallest number k s.t. $a_{i,j} = 0$ for all $|i - j| > k$.

- the smallest number $k = 37$, the bandwidth = 37.

(d), Analysis of the LU factors of the Jacobian

- The number of non-zero elements L is 6593.
- The number of noe-zero elements U is 7601.
- The graph of the sparsity pattern of the L-U factors below:
- Using LU(p) method can cause the fill-in problem.



(e), Simulations with sparse version of the solver

Different N	Time Taken (seconds)	Total Iterations (S)	Average time t_S = T/S (seconds)
2197	0.144467	4	0.036117
4913	0.219073	4	0.054768
9261	1.683312	14	0.120236

- $t_S = CN^P$ and $log(t_S) = logC + Plog(N)$
- By fitting the N values and t_S, can get $log(t_S) = -10.06 + 0.86log(N)$ approximately.
- Get the cost of one New iteration with the N, have the equation: $t_S = e^{-10.06}N^{0.86}$.
- Compare with the 2(a), which find the cost time of this new sparsity-based implementation using the timings of the numerical is less than the 2(a).