Homework #8 • Numerical Analysis II (math 416) • Crank-Nicolson for Diffusion

- due Wednesday 22 November.
- quality of presentation will count heavily.

A) (2 pages) For the diffusion equation, $u_t = D\, u_{xx}$, that is periodic on the domain $0 \le x \le 2L$, show that the local truncation error for the Crank-Nicolson scheme is second-order

$$\bar{L}_{\Delta x,\Delta t}[u(x_j, t_k)] = O(\Delta x^2) + O(\Delta x \Delta t) + O(\Delta t^2) \ .$$

B) (2 pages) Present the Von Neumann analysis that shows the above Crank-Nicolson scheme is unconditionally stable. Explain your logic well.

C) (4 pages) Read carefully, this problem is not as difficult as it may seem. The goal is to implement the Crank-Nicolson algorithm for the diffusion problem

$$u_t = u_{xx} \quad \text{on} \quad 0 \le x \le \pi \ , \ 0 \le t$$

with zero initial conditions, and time-dependent Neumann boundary conditions

$$u_x(0,t) = \cos(t) \quad ; \quad u_x(\pi, t) = 0 \ .$$

Impose the boundary conditions using the ghost point approach, so that

$$u_1^k - u_{-1}^k = 2\,\Delta x\, \cos(t_k) \quad ; \quad u_{N+1}^k - u_{N-1}^k = 0$$

where these relations should be used to eliminate the ghost values from the discretized equations ($j = 0 \to N$).

The implicit vector iteration for $\vec{U}^k = (u_0^k \ldots u_N^k)^T$ can be written as the matrix equation

$$[\mathbf{M}(-\lambda)]\, \vec{U}^{k+1} = [\mathbf{M}(\lambda)]\, \vec{U}^k + \frac{1}{2}\,(\vec{f}^{k+1} + \vec{f}^k) + (\vec{b}^{k+1} + \vec{b}^k)$$

where $[\mathbf{M}(\lambda)]$ is a tridiagonal matrix that depends on $\lambda = D\Delta t/\Delta x^2$. Be sure to optimize these matrices using the Matlab *sparse* command. For this problem, the forcing ($\vec{f}$) is zero, but there is a boundary effect ($\vec{b}$).

The pseudocode for this script should roughly follow:

1. initialize solution vector
2. define sparse **M**-matrices
3. execute timestepping loop:
   increment time variable
   calculate $\vec{b}$-vector
   sparse inversion for solution update
4. graphical output

Remember, your code should be second-order convergent. After a fairly long time ($T \approx 60\pi$ = 30 periods of the BC), the solution should settle into a time-periodic solution. After this settling has occurred, compare the behaviour of the end values $u(0,t)$ and $u(\pi/2, t)$ over a t interval of 2π – what might this say about judging the temperature inside a room by touching the outside walls?

(For hint: see the class webpage. Also, deciding on good parameter values ($\Delta x, \Delta t$) may take some collaboration.)