ELEC 418/518 – Homework #4

Problem 1. Consider using the following multi-step scheme in order to solve for $x(t)$ which satisfies the differential equation $\frac{d}{dt}x(t) = \lambda x(t)$,

$$\frac{x[m+1] - x[m-1]}{2h} = \lambda x[m],$$

where $x[0] = 1$. Here $x[m]$ approximates $x(t)$ at time point $t = mh$.

a. Determine the local truncation error of this "leap frog" method.

b. Is the method stable? Is the method convergent?

c. Plot and compare the computed and the exact solution for the case when $\lambda = 1$, and on the interval $t \in [0, 10]$. Use $h = 2$, $h = 0.5$, and $h = 0.1$.

d. Examine your plots carefully, and explain your results in part (c).

Problem 2. In this problem, you will generalize your circuit simulator to include capacitors, and then use the simulator to solve a dynamic version of the heat equation.

a. Modify your nodal analysis based resistor and current-source circuit simulator so that circuits that also include capacitors can be analyzed. Your implementation should read a file with the list of resistors, current sources, capacitors and initial node voltages. The format for the first two was given in Homework #1, and the format for capacitors is:

clabel node1 node2 val

where *label* is an arbitrary label, *node1* and *node2* are integer circuit node numbers, and *val* is the capacitance (a floating point number).
The format for specifying initial node voltages is

zlabel node val

where *label* is an arbitrary label, *node* is a circuit node number, and *val* is the circuit node's voltage at time $t = 0$.
In order to compute the circuit behavior as a function of time, implement fixed time-step versions of forward and backward Euler and BDF2 (described further below) schemes in your circuit simulator.

b. Write code that can generate an N-node resistor and capacitor circuit model (netlist) which represents the spatial discretization of the heat equation

$$\frac{\partial^2 T(x,t)}{\partial x^2} = \frac{\partial T(x,t)}{\partial t}, \quad t \in [0,10], \quad x \in [0,1]$$

where $T(x,t)$ is the temperature at location x at time t. To be completely specified, the heat equation requires initial conditions, $T(x,0)$ for all x, and boundary conditions, $T(0,t)$ and $T(1,t)$ for all t. For this example, please use

$$T(1,t) = T(0,t) = 0$$

for the boundary conditions and

$$T(x,0) = 0 \text{ for } x > 0.7 \text{ and } x < 0.3$$

$$T(x,0) = 1 \text{ for } 0.3 \leq x \leq 0.7$$

for the initial conditions.
Be sure to clearly specify how the initial conditions translate to initial node voltages in your circuit model.

c. Use your N-node circuit model generator and your modified circuit simulator to solve the dynamic heat equation above using forward and backward Euler and the BDF2 schemes with fixed time-steps.
The BDF2 is an implicit 2-step scheme, with $\alpha_0 = 1$, $\alpha_1 = -4/3$, $\alpha_2 = 1/3$, $\beta_0 = 2/3$, $\beta_1 = \beta_2 = 0$ for fixed time steps.
Note that backward Euler and BDF2 will run faster if you use sparse matrix techniques.

What is the largest time-step for which the forward Euler solution does not oscillate and grow for $N = 10$ and $N = 100$?
What is the largest time-step for which the plots of the backward Euler and BDF2 solutions (for $N = 10$ and $N = 100$) still look reasonable?
Explain your results.

Problem 3. The equation

$$\frac{d^2}{dt^2}x(t) = -x(t) + \mu(1 - x(t)^2)\frac{d}{dt}x(t)$$

known as the Van der Pol equation, represents a simple oscillator.

a. Reduce the Van der Pol equation to a system of first-order ODEs.

b. Implement forward Euler, backward Euler and the trapezoidal schemes for the solution of the Van der Pol oscillator.

c. Use your implementations from part (b) to generate solutions for the Van der Pol oscillator for $\mu = 1$ and $\mu = 10$. Use $x(0) = 2$, $\dot{x}(0) = 0$ and experiment with various step sizes to obtain reasonable accuracy for each value of μ with all of the schemes you implemented in part (b). Compute the solution in the time interval $t \in [0, \max(20, 10\mu)]$.
Comment on your results.

Problem 4.

a. Compute the coefficients for the most accurate (in the sense of satisfying as many exactness constraints as possible) *explicit* linear multi-step formulas for which

$$\alpha_0 = 1,\ \alpha_1 = -1, \text{ and } \alpha_3 = \cdots = \alpha_k = 0.$$

Compute the coefficients for $k = 1, 2, 3, 4$. What power of Δt is LTE proportional to for $k = 1, 2, 3, 4$?

b. Compute the coefficients for the most accurate (in the sense of satisfying as many exactness constraints as possible) *implicit* linear multi-step formulas for which

$$\alpha_0 = 1,\ \alpha_1 = -1, \text{ and } \alpha_3 = \cdots = \alpha_k = 0.$$

Compute the coefficients for $k = 1, 2, 3, 4$. What power of Δt is LTE proportional to for $k = 1, 2, 3, 4$?

c. Compute the coefficients for the most accurate (in the sense of satisfying as many exactness constraints as possible) *implicit* linear multi-step formulas for which

$$\alpha_0 = 1, \text{ and } \beta_1 = \cdots = \beta_k = 0.$$

Compute the coefficients for $k = 1, 2, 3, 4$. What power of Δt is LTE proportional to for $k = 1, 2, 3, 4$?

Problem 5.

a. Write a Matlab function that can produce the plot for the region of absolute stability on the complex plane for a general k-step linear multi-step scheme.

b. Use your Matlab function from part (a) to produce three plots for the regions of absolute stability for all of the linear multi-step formulas from Problem 4. Produce three plots for the formulas in Problem 4(a), 4(b) and 4(c) for k = 1, 2, 3, 4.

Problem 6. Consider the following initial value problem (IVP):

$$\frac{d}{dt}x(t) = -100(x(t) - \cos(t)) - \sin(t), \quad x(0) = 1$$

a. Determine the exact solution of the above problem.

b. Implement the solution of the IVP using the linear multi-step formula from Problem 4.a with $k = 2$.

c. Implement the solution of the IVP using the linear multi-step formula from Problem 4.c with $k = 2$.

d. Use your implementations both in part (b) and part (c) to compute $x(1)$. Repeat for step sizes of $0.2, 0.1, 0.05, 0.02, 0.01, 0.005, 0.001$. Produce a log-log plot of the computed errors (when compared with the exact solution) for $x(1)$ as a function of step size for both of the methods you have implemented.

e. Comment on your results in part (d). Are both methods accurate for all values of step size? If not, at what step size do the method(s) fail? Why? Are you able to theoretically determine the largest step size for which the method(s) will be accurate in computing $x(1)$, even if you did not have the experimental data from part (d)?