UCB Math 228A, Fall 2009: Problem Set 2

Due September 25

1. Determine the region of absolute stability for the θ–method

$$u_{n+1} = u_n + hf(t_n + \theta h, (1-\theta)u_n + \theta u_{n+1})$$

for enough $\theta \in [0,1]$ to make the general picture clear. Note that this method includes explicit Euler, the trapezoidal method, and implicit Euler as special cases. For which $\theta \in [0,1]$ is the method A–stable? How about $A(\alpha)$–stable?

2. Show that the difference method

$$u_{n+1} = u_n + a_1 f(t_n, u_n) + a_2 f(t_n + \alpha_2, u_n + \delta_2 f(t_n, u_n)), \tag{1}$$

cannot have local truncation error $\mathcal{O}(h^4)$ for any choice of constants a_1, a_2, α_2, and δ_2.

3. Consider the ODE

$$u'(t) = \lambda(u - \cos t) - \sin t, \qquad u(0) = u_0, \tag{2}$$

with the exact solution

$$u(t) = e^{\lambda t}(u_0 - 1) + \cos t. \tag{3}$$

Write a function `p2_3(lambda,u0,h)` that solves (2) for $0 \le t \le 3$, using the two methods

$$\text{Backward Euler: } u_{n+1} = u_n + hf(t_{n+1}, u_{n+1}),$$

$$\text{Trapezoidal Method: } u_{n+1} = u_n + \frac{h}{2}(f(t_n, u_n) + f(t_{n+1}, u_{n+1})).$$

The code should produce a plot of the two computed solutions and the true solution. Run your code with $\lambda = -10^6$, $u_0 = 1.5$, and stepsize $h = 0.1$, and explain the results in terms of $R(z)$ for the two methods.

4. **a)** Write a function `p2_4a` that plots the stability regions of the 6-step BDF method:

$$147u_{n+6} - 360u_{n+5} + 450u_{n+4} - 400u_{n+3} + 225u_{n+2} - 72u_{n+1} + 10u_n = 60hf(u_{n+6}), \tag{4}$$

and the TR-BDF2 method:

$$k_1 = u_n + \frac{h}{4}(f(u_n) + f(k_1)), \tag{5}$$

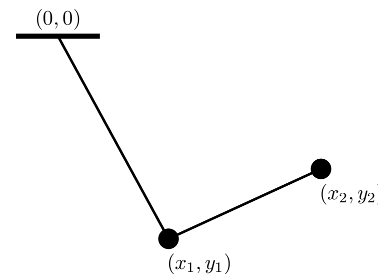
$$u_{n+1} = \frac{1}{3}(4k_1 - u_n + hf(u_{n+1})). \tag{6}$$

b) Write a function `p2_4b(n)` that computes the matrices A produced by the scripts `heat1d` and `conv1d` on the course web page for a given value of n, and plots their eigenvalues in the complex plane with `axis equal`. For $n = 100$, determine which of the two methods above is appropriate for which matrix (BDF6 is more accurate, so use it if possible).

c) Write two functions `heat1dimpl(n,dt)` and `conv1dimpl(n,dt)` that solve the original problems using the new methods and the given timestep. Use the final times $T = 0.2$ and $T = 1.0$, respectively, and plot the solution at the final time. Set all the six starting solutions in BDF6 equal to the initial condition, i.e., $u_1 = u_2 = \cdots = u_5 = u_0$. Test the methods using the timesteps $\Delta t = 10^{-3}$ for the heat equation and $\Delta t = 10^{-2}$ for the convection equation.

Turn page $\longrightarrow$

5. The double spring pendulum in the figure consists of two particles of mass m connected by linear springs. The force in each spring is given by $F = k_{\mathrm{spr}}(L - L_0)$, where k_{spr} is the spring constant, L the current length, and L_0 the equilibrium length. The direction of this force is along the orientation of the spring, and the force is repulsive when $L < L_0$ and attractive if $L > L_0$. The total force on each particle is equal to the vector sum of the forces from the attached springs, plus the gravitational force of mg in the negative y-direction. Use the constant values $m = L_0 = 1$ and $g = 10$.



a) Derive the equations of motion for the system in the form of a system of first order differential equations. Implement them in a MATLAB function of the form

```
function f = fpend(u,kspr)
```

where `u` is a vector with the 8 components x_1, y_1, x_2, y_2 and their first derivatives.

b) Write a function that solves the system using RK4, of the form

```
function rk4pend(u0,dt,kspr)
```

where `u0` is the initial condition, `dt` is the timestep, and `kspr` is the spring constant. Plot the pendulum after each timestep using the `pendplot.m` function on the course web page. You can test your function using the command

```
rk4pend([1;0;1;1;0;0;0;0],0.02,100);
```

and verify that the motion looks realistic.

c) Write a function

```
function trbdf2pend(u0,dt,kspr)
```

that solves the system using the implicit TR-BDF2 method. Use Newton iterations with a linearization of `fpend` for each stage. Output the norms of each Newton update, similar to the function `pendulum_impl` on the course web page. Like before, plot using `pendplot.m`. Test it using the command

```
trbdf2pend([1;0;1;1;0;0;0;0],0.02,1e6);
```

Code Submission: E-mail the MATLAB files `p2_3.m`, `p2_4a.m`, `p2_4b.m`, `heat1dimpl.m`, `conv1dimpl.m`, `fpend.m`, `rk4pend.m`, `trbdf2pend.m`, and any supporting files to Trevor at potter@math.berkeley.edu as a zip-file named `lastname_firstname_PS#.zip`, for example `potter_trevor_2.zip`.