AMath 586 / ATM 581
Homework #2
Due Thursday, April 23, 2019

Homework is due to Canvas by 11:00pm PDT on the due date.

To submit, see `https://canvas.uw.edu/courses/1271892/assignments/4790261`

Problem 1

Which of the following Linear Multistep Methods are convergent? For the ones that are not, are they inconsistent, or not zero-stable, or both?

(a) $U^{n+3} = U^{n+1} + 2kf(U^n)$,

(b) $U^{n+2} = \frac{1}{2}U^{n+1} + \frac{1}{2}U^n + 2kf(U^{n+1})$,

(c) $U^{n+1} = U^n$,

(d) $U^{n+4} = U^n + \frac{4}{3}k(f(U^{n+3}) + f(U^{n+2}) + f(U^{n+1}))$,

(e) $U^{n+3} = -U^{n+2} + U^{n+1} + U^n + 2k(f(U^{n+2}) + f(U^{n+1}))$.

Problem 3

(a) Determine the general solution to the linear difference equation $U^{n+2} = U^{n+1} + U^n$.

(b) Determine the solution to this difference equation with the starting values $U^0 = 1$, $U^1 = 1$. Use this to determine U^{30}? (Note, these are the *Fibonacci numbers*, which of course should all be integers.)

(c) Show that for large n the ratio of successive Fibonacci numbers U^n/U^{n-1} approaches the "golden ratio" $\phi \approx 1.618034$.

Problem 2 Any r-stage Runge-Kutta method applied to $u' = \lambda u$ will give an expression of the form

$$U^{n+1} = R(z)U^n$$

where $z = \lambda k$ and $R(z)$ is a rational function, a ratio of polynomials in z each having degree at most r. For an explicit method $R(z)$ will simply be a polynomial of degree r and for an implicit method it will be a more general rational function.

Since $u(t_{n+1}) = e^z u(t_n)$ for this problem, we expect that a pth order accurate method will give a function $R(z)$ satisfying

$$R(z) = e^z + \mathcal{O}(z^{p+1}) \quad \text{as } z \to 0.$$

This indicates that the one-step error is $\mathcal{O}(z^{p+1})$ on this problem, as expected for a pth order accurate method.

The explicit Runge-Kutta method of Example 5.13 is fourth order accurate in general, so in particular it should exhibit this accuracy when applied to $u'(t) = \lambda u(t)$. Show that in fact when applied to this problem the method becomes $U^{n+1} = R(z)U^n$ where $R(z)$ is a polynomial of degree 4, and that this polynomial agrees with the Taylor expansion of e^z through $O(z^4)$ terms.

We will see that this function $R(z)$ is also important in the study of absolute stability of a one-step method.

Problem 2

Determine the function $R(z)$ described in the previous exercise for the TR-BDF2 method given in (5.37). Note that this can be simplified to the form (8.6), which is given only for the autonomous case but that suffices for $u'(t) = \lambda u(t)$. (You might want to convince yourself these are the same method).

Confirm that $R(z)$ agrees with e^z to the expected order.

Note that for this implicit method $R(z)$ will be a rational function, so you will have to expand the denominator in a Taylor series, or use the Neumann series

$$1/(1-\epsilon) = 1 + \epsilon + \epsilon^2 + \epsilon^3 + \cdots .$$

Problem 4

The Jupyter notebook `Pendulum_ForwardEuler.ipynb` gives an implementation of Forward Euler on the nonlinear pendulum problem

$$\theta'(t) = v(t) \tag{1}$$
$$v'(t) = -\sin(\theta(t)) \tag{2}$$

Modify this code to implement the Backward Euler method. Since this is an implicit method, you need to solve a nonlinear equation in each step. Although it is a system of two equations, you can reduce it to a scalar nonlinear equation. In each step we have to solve

$$\Theta^{n+1} = \Theta^n + kV^{n+1}, \tag{3}$$
$$V^{n+1} = V^n - \sin(\Theta^{n+1}) \tag{4}$$

and substituting the second equation in the first gives a single equation to solve for Θ^{n+1}, of the form $\phi(theta) = 0$ with $\phi(\theta) = \theta + k^2\sin(\theta) - (\Theta^n - kV^n)$. This can be done in each time step by defining the function ϕ and then calling the root-finder scipy.optimize.fsolve. The value Θ^n from the previous time step is generally a good starting guess for the root. So this gives a loop like:

```
for n in range(0,nsteps):
    phi = lambda theta: theta + dt**2*sin(theta) - (U[0,n]  + dt*U[1,n])
    new_theta = fsolve(phi, U[0,n])
    U[0,n+1] = new_theta
    U[1,n+1] = U[1,n] - dt * sin(new_theta)
```

Comment on how solutions computed with this method compare to those obtained with Forward Euler. Test with different size time steps to confirm that you see the expected first order accuracy. For this you may want to take a much shorter time interval than in the notebook in order to get the expected asymptotics with a reasonable number of time steps. Also note that you cannot necessarily assume the reference solution computed with `solve_ivp` is as accurate as the tolerance specified, especially over long time intervals!

Problem 5

Repeat the previous problem for the Trapezoidal method. Again comment on the behavior of the solution in this case, and check the order of accuracy.