Homework 4

1. Eigenvalues of Adams-Bashforth methods

(a) Compute or look up the coefficients (alphas and betas) of the 2-step Adams-Bashforth method.

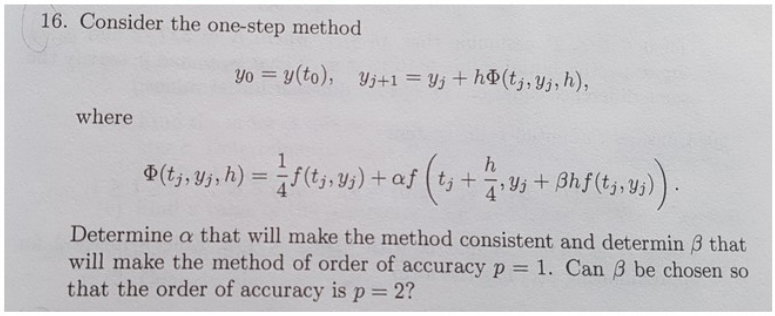
(b) Write down the characteristic polynomial of the method in the variable z when applied to the scalar equation $y' = \lambda y$ with step size h, and compute the eigenvalues.

(c) Show that one of the eigenvalues approximates the one-step change $e^{h\lambda}$ of the exact solution when $h\lambda$ is small, and that the other one is small leading to rapid decay of the corresponding mode.

(d) Find the value of $h\lambda$ at which stability is lost as h increases with λ real and negative.

2. Ackleh et al. Exercise 7.10.16¶

```
from IPython.display import Image
Image('Ackleh_ex_7.10.16.jpg',width=600)
```



16. Consider the one-step method

$$y_0 = y(t_0), \quad y_{j+1} = y_j + h\Phi(t_j, y_j, h),$$

where

$$\Phi(t_j, y_j, h) = \frac{1}{4} f(t_j, y_j) + \alpha f\left(t_j + \frac{h}{4}, y_j + \beta h f(t_j, y_j)\right).$$

Determine α that will make the method consistent and determin β that will make the method of order of accuracy $p = 1$. Can β be chosen so that the order of accuracy is $p = 2$?

3. Venn diagram

Give an example of a linear multistep formula of each of the following types:

- consistent and A-stable
- consistent and stable but not A-stable
- consistent but not stable
- stable but not consistent
- neither consistent nor stable

4. An implicit method for stability on a stiff system

(a) Implement the trapeziodal method (see p423) to solve the stiff system,

$$Y' = \begin{bmatrix} x' \\ y' \\ z' \end{bmatrix} = \begin{bmatrix} -y \\ x \\ -41(z - 2x^2 - y^2) \end{bmatrix}.$$

with the initial condition $Y(0) = [3, 0, 18]^T$, going to t=10 with h=.05. Use a fixed number of Newton iterations. Plot x,y,z vs. t in 3 separate subplots.

(b) Compare with using Euler's method with the same step size, and superimpose the results on the plot from part (a).

Compose your formal answer using GoodNotes, Notability, OneNote, Xournal, or a similar tool of your choice, with text, code, graphics all displayed in an integrated way so that your "story" can be read in one pass. Upload this to Gradescope.

Also upload your code to UBlearns.

Hints

You need to create functions like:

```
def f(t,y):
    ...

def Df(t,y):
    ...
```

To solve the matrix-vector equation Ax=b:

`np.linalg.solve(A, b)`

To get the identity matrix:

`np.eye(3)`

Calculate the Newton step $s = -Dg^{-1} g$ by solving $Dg\ s = -g$ for s.

Don't confuse the jacobian of the function g whose root you want to find with the jacobian of f (the right hand side of the differential equation), though the former is expressed simply in terms of the latter.