Numerical Methods for Differential Equations
Homework 3

Due by 2pm Tues 1 Nov. ***Please submit your hardcopy at the start of lecture.*** *The answers you submit should be attractive, brief, complete, and should include program listings and plots where appropriate. Possible approaches include "publish" in Matlab/Octave and* `pims.jupyter.ca`.

Problem 1 (RK Stability). What is the amplification factor for "*the* 4th-order Runge–Kutta method"? (That is, the one covered in the notes). Plot its region of absolute stability in the complex plane. Note that the stability region includes part of the imaginary axis; write a code to determine (to 10 decimal places) the largest value of $y \in \mathbb{R}$ such that $z = iy$ is included in the linear stability region.

Problem 2 (BDF-2 Stability Analysis).

(a) Consider the BDF-2 linear multistep method. Complete a zero-stability analysis of the method.

(b) Now perform an absolute stability analysis of the scheme. Hint: while we didn't look at this in lecture, the idea is quite similar to the Runge–Kutta case: try to solve the Dahlquist test problem, but now *two* quantities will need to each be bounded by 1.

(c) Plot the BDF-2 region of absolute stability in the complex domain $[-4, 8] \times [-6, 6]$.

(d) Show that the BDF-2 scheme is L-stable. Hint: consider the limit as $z = k\lambda \to -\infty$.

(e) Now consider a fixed λ and consider the $k \to 0$: comment on any similarity or difference to your results in (a).

Problem 3 (Forward Euler and IVP versus BVP). This problem looks at relating convergence in the initial value problem back to the example we did for the BVP $u_{xx} = f$.

Consider the Forward Euler method applied to the Dahlquist test problem. Assume we start at $t = 0$ and compute until $t = T_f = Nk$. Instead of "telescoping" the recursion (as we did in lecture when looking at absolute stability), we will write *all* the steps of the method (from $n = 1$ up to $n = N$) as one $N \times N$ matrix. It should have a diagonal and a subdiagonal only. Call this matrix A.

(a) Write down a system $AU = F$ where $U = \begin{bmatrix} U^1 \approx u(t_1) \\ U^2 \approx u(t_2) \\ \vdots \\ U^N \approx u(T_f) \end{bmatrix}$ is the vector of numerical solutions at each time-step t_n and F is mostly zeros (except the first entry). You can leave some "$\ddots$" but you should otherwise give the matrix A explicitly. Hint: if you get stuck, you can find something very similar in text of LeVeque 2007, Chapter 6.

(b) Now let $\hat{U}$ be the exact solution, sampled at the time-steps t_n. Following our approach in lecture for the BVP, find an expression for the global error E (a vector of values for each time-step t_n), in terms of the local truncation error τ (also a vector of values for each time-step t_n).

(c) Compute (in Octave/Matlab or otherwise) the inverse of this matrix, and measure its size in the matrix max norm $\|A^{-1}\|_\infty$ and the matrix Frobenius norm $\|A^{-1}\|_F$. Report these numbers for the pairs $(N, k) = (10, 0.1)$, $(50, 0.02)$, $(100, 0.01)$ (that is give 6 numbers). For a fixed final time T_f, does it seem like the inverse is bounded independent of k?

(d) Explain how this shows convergence of the Forward Euler method for fixed T_f (at least for this problem, or more generally, at least for linear problems).

Problem 4 (Symmetry and Simpson's Rule). Explain, by application of one of the error theorems, why the Simpson's Rule quadrature scheme is exact for polynomials of degree 3. (Recall this result is counter-intuitive because the method can be viewed as fitting a merely quadratic polynomial proxy and integrating exactly)

Next give a geometric/algebraic argument for why this happens. You should assume (without loss of generality) that $x_0 = -h, x_1 = 0, x_2 = h$. Hint: consider the polynomial in monomial form.

Problem 5: (Trapezoidal Rule). For certain classes of problems, composite Trapezoidal Rule works much better than our analysis in lecture. Consider the problem of evaluating $\int_0^{2\pi} e^{\cos x} dx$. The exact answer is $2\pi I_0(1)$, in terms of the modified Bessel function of the first kind (say what? just put "`2*pi*besseli(0, 1)`" into Octave). Make a table of errors for $n = 1, 2, 3, \ldots, 15$. Plot on a log-log scale; do you see the usual straight-line? Make another plot of log(err) versus n ("`semilogy`").

See the article *The Exponentially Convergent Trapezoidal Rule*, Trefethen and Weideman, Sirev 2014. This example comes from Section 4 of their paper; but be careful there is a small typo[1].

Compute this integral using composite Simpson's Rule: what is the convergence rate?

Problem 6: Git (for bonus points). Hopefully you have a working "git clone" of your fork (`gitlab.math.ubc.ca/yourname/math405_2016`). Hopefully you can also "git pull upstream/master" (or similar) to get the most changes from my original repo.

Option 1: Make a new *branch* in your clone. On the command line, I use "git checkout -b myfix". Make some very small change (suggestions below). Now "git push" the branch to your fork (on the gitlab server).

Log into the gitlab server and submit a "merge request" from "yourname/math405_2016" to my "cbm/math405_2016". Take a screenshot of your merge request: it should have a ticket number and URL (see for example `https://gitlab.math.ubc.ca/cbm/math405_2016/merge_requests/1`).

Suggestions for possible changes:

- Fix a typo in my notes or demos.
- Add a comment to one of the demos, e.g., at the top of the file summarizing what it did.
- There are some missing parts in the written notes (sometimes I covered more things in lecture). You could insert a new section in the notes. But make sure its a small change, like a Section heading and two or three sentences, possibly ending with a "TODO".

Key thing here is to make a small change (the "diff" should just be a few lines.)

Option 2: Alternatively, you can *review* someone else's merge request: login to `gitlab.math.ubc.ca/cbm/math405_2016`, find a merge request. Add a comment saying whether or not you agree with the proposed change. You can ask them to revise if necessary. Take a screenshot showing your comment. *Please note*, we probably don't want 1 person submitting a minor typo fix and 38 people giving "+1"s...

Note: If you don't get this exercise now, leave it for the next homework. But it might be easier to do it now (because of the potential for dreaded merge conflicts and the hopefully monotonic decrease of typos).

[1] Finding a typo in Nick Trefethen's publication: bucket list item ticked ;-)