MATH 6737.A – Numerical Differential Equations

Spring 2025

HW # 0

Due: 01/17/25

Problem 1 (**0.5 point**)

Find the explicit form of the cubic term (i.e. the term with $(\Delta x)^n(\Delta y)^m$, $n+m=3$) in expansion (0.6).

Problem 2 (**0.5 point**)

Find the Lipschitz constant L for:

(a) $f(x,y) = x\,y^2$ on $R:\ 0 \le x \le 3,\ 1 \le y \le 5$;

(b) $f(x,y) = x + |\sin 2y|$ on $R:\ 0 \le x \le 3,\ -\pi \le y \le \pi$.

Hint: See Remarks after Theorem 0.1.

Problem 3 (**0.5 point**)

Solve the IVP

$$y' = 2y + e^{3x}, \qquad y(-1) = 4\,.$$

Problem 4 (**0.5 point**)

Find

$$\lim_{h\to 0} \left(\frac{1}{1-2h}\right)^{\pi/h}\,.$$

HW # 1

Due: 01/22/25

Problem 1 (**1.5 points**)

Consider an IVP

$$y' = ay, \qquad y(x_0) = y_0\,,$$

where a is a constant. Following the lines of the derivation of the global error (see Eq. (1.4)[1]), find how the *sign* of the global error will depend on the signs of a and y_0. See *Notes 1–3* below.

Verify your answer by solving the above IVP for $x \in [0,\ 1]$ [2] using the simple Euler method in four cases: (i) $a=1$, y_0=1; (ii) $a=1$, $y_0=-1$; (iii) $a=-1$, y_0=1; (iv) $a=-1$, $y_0=-1$. Use $h=0.1$. See *Note 4* below.

Note 1: The analytical solution of the above IVP can be found in Lecture 0.4 (i.e., Lecture 0, section 4). (Keep in mind that it, of course, depends on the initial condition, not just on the ODE.)

Note 2: If the signs of the local truncation error at each iteration are the same, then the sign of the global error will be the same as the sign of the local truncation errors.[3]

Note 3: For $f(x,y) = ay$, one can do more explicit calculations than in Sec. 1.3, and it *will* be possible to establish the sign of the error.

Note 4: An example of a code using the simple Euler method is posted as `hw1_EX_Euler_direct.m` on the course website under the link "Codes for examples and selected homework problems." Note, however, that it uses a somewhat different function than in this problem.

Problem 2

Find coefficient B for the Midpoint method (1.19). Follow the lines of the derivation of coefficient A for the Modified Euler method.

Problem 3

Derive the expression for the local truncation error of the Midpoint method. Follow the lines of the similar derivation for the Modified Euler method.

[1] You do *not* need to read past that equation to answer this question.

[2] Thus, the initial and final points of the computation are $x_0 = 0$ and $x_{\text{final}} = 1$.

[3] An additional condition that is required for this to hold is $(1 + ah) > 0$ (you will see that when you will have carried out the derivation). Assume that this condition holds for the case(s) in question.

Problem 4 (**1.5 points**)

The exact (and unique) solution of the IVP

$$y' = \sqrt{y}, \qquad y(0) = 1$$

is $y = (x+2)^2/4$. Solve the above IVP for $x \in [0, 1]$ using three methods: simple Euler, Modified Euler, and Midpoint. In each case, use two values for the step size: $h = 0.1$ and $h = 0.05$. Make a table that will show the error of your numerical solution at $x = 1$ (i.e., the global error) versus the method used and the step size. By what factor does the error decrease when the step size decreases from 0.1 to 0.05?

Are your results consistent with the expressions for the local truncation errors of these three methods, derived in Lecture 1 and in Problem 3 above? Namely, answer, with some explanation, each of the following questions:

(i) Do the computed errors indeed obey their expected orders $O(h^n)$ with respective n's? (The answer should follow from your Table.)

(ii) Does the sign of the error for the simple Euler method agree with that which follows from the derivation in Sec. 1.3 for this specific $y(x)$?

(iii) Do the signs and relative magnitudes of the errors of the Modified Euler and Midpoint methods agree with the result of Sec. 1.6 and your result in Problem 3? (This will require some calculation. Recall that in Eq. (1.32) and the formula you have derived in Problem 3, the derivatives of f are *partial*.)

Technical notes:

1. Two alternative examples of a code using the simple Euler method are posted under the link "Codes for examples and selected homework problems." You may mimic your work on either of these examples.

2.

When programming a new numerical scheme for the first time, it is a good idea to stay close to the notations in which this scheme is written in the lecture. In theory, you can use any notations. However, when you are stuck and ask for my help, I will provide useful feedback about your code most efficiently if it is written in my notations. I will not learn your notations just to help you.
Please keep this concept in mind when programming all assignments in this course.

Problem 5 (**1.5 points**)

A sky diver jumps from a plane. Assume that during the time before the parachute opens, the air resistance is proportional to the diver's velocity v. (In reality, it is proportional to v^a with $a > 1$, but we sacrifice the physics in exchange for being able to obtain a simple analytical solution to this problem.)

To begin, write down the ODE for the diver's velocity. Supply the numerical values for the coefficients in this ODE if it is known that the maximum diver's speed is 80 mph. (This speed could be achieved only asymptotically, assuming that the diver will be falling down forever.) See *Technical notes* at the end of this problem.

Solve the above ODE, assuming that the diver's initial velocity is zero. Namely:

First, find the analytical solution (you may consult Lecture 0.4).

Second, solve the corresponding IVP by the Modified Euler method for the first 2 seconds of the diver's fall. Use the step size of 0.2 sec. Plot both solutions in the same figure. Also, in a *separate* figure, plot the numerical error for all $t \in [0, 2]$.

Technical notes for setting up the ODE:

1. Choose as positive either the upward or downward direction. It does not matter which one you choose, but once you will have chosen it, you must stay consistent with your choice in all subsequent steps. Draw a picture, where you indicate your positive direction and the velocity of the diver.

2. Write the Second Law of Newton: $ma = F_{\text{gravity}} + F_{\text{air}}$, where m is the mass and a is the acceleration of the diver. How is a related to the velocity?

3. When writing down the expression for F_{gravity}, pay attention to its sign: see Note 1 above. Make sure to draw F_{gravity} in your picture.

4. Now, $|F_{\text{air}}| = \alpha|v|$, where $\alpha > 0$ is a proportionality coefficient, to be determined later. You must figure out whether $F_{\text{air}} = +\alpha v$ or $F_{\text{air}} = -\alpha v$. The way to do so is by drawing F_{air} in your picture and deciding whether it is positive or negative. (F_{air} is always directed opposite to the velocity.)
5. Finally, the value of a quantity related to α will be found from your analytical solution and the given terminal speed.

HW # 2

Due: 01/29/25

Problem 1

Derive Eqs. (2.4). Follow the lines of the derivation of coefficient A in Sec. 1.4. (In fact, Eq. (1.21) should be used without any changes.)

Problem 2

(a) Write a function file named `yourname_cRK.m` that can integrate any given IVP $y' = f(x, y)$, $y(x_0) = y_0$ using the classical Runge–Kutta method.

Note: Use as an example the posted sample code, mentioned in Problem 4 of HW# 1, which implements the simple Euler method to integrate the above IVP.

(b) Consider the IVP you derived in Problem 5 of HW# 1. Solve it using the function `yourname_cRK.m` with the same step size as in HW# 1 (i.e., 0.2 sec). Plot the error of your numerical solution.

Compare the final error (at $t = 2$) of the cRK method with the error of the Modified Euler method, obtained in HW# 1. Do you think that the relative magnitudes of these errors are consistent with the orders of the respective methods?
Hint for the last question: What are the orders of magnitude (in terms of h) of the errors of these two methods and what is their ratio?

Problem 3

Consider the following modification of Problem 5 in HW# 1. Assume that at $t = 2$ sec, the parachute opens. This results in the air resistance coefficient changing abruptly in such a way that the maximum possible velocity of the parachutist is now 4 mph (versus 80 mph without the parachute).

(a) Find the numerical value of the new air resistance coefficient and then solve analytically the modified ODE up to $t = 4$ sec (i.e., the first 2 seconds without a parachute, as in HW# 1, and the last 2 seconds with the parachute).[4]
Hint: To find the solution after $t_{\text{Open}} = 2$, use the formula from Lecture 0, where the initial condition will be your analytical solution at the end of the interval $0 \le t \le t_{\text{Open}}$.

(b) Solve the above IVP up to $t = 4$ sec using the function file `yourname_cRK.m` that you wrote in Problem 2 (see the *Technical notes* below on how to handle a discontinuous $f(t, v)$). Use the step sizes of 0.2 sec. Plot the exact and the numerical solutions in the same figure.

Compute its error for all $t \in [0,\, 4\,\text{sec}]$ and plot it in another figure.

Save the error in a file (type `help save` for the syntax). This error along with that for the `ode45` solution (see Problem 4) will be plotted in Problem 5.

Technical notes:

1. Define a discontinuous function for your ODE in a separate file `yourname_fun4_hw2_p3`. Type the following into the file:

```
function f = yourname_fun4_hw2_p3(t,y)
if t >= 0 &  t <= t_Open
   f= "r.h.s. of ODE before parachute opens";
else
   f= "r.h.s. of ODE after parachute opens";
end
```

[4] Even though now in equation $dv/dt = g - \alpha(t)\, v \equiv f(t, v)$ the function $f(t, v)$ is discontinuous in t (at $t = 2$ seconds), it is still continuous and Lipshitz in v for any one t. Thus, by the existence and uniqueness theorem for ODEs, we are still guaranteed to have a unique solution $v(t)$.

Of course, you supply the actual expressions on the r.h.s.
(Note that the name for your function must be the name of the file where you store it.)
After the line `function f=yourname_fun4_hw2_p3(t,y)`, declare as global variables those parameters which you need to communicate from the main code to the function; see the posted sample codes for HW# 1. Do *not* enter the numeric values of your terminal velocity (e.g., 80 mph) etc. into the function code. Instead, enter them in the main code and communicate them to the function through the `global` declaration.

2. To plot the analytical solution, follow a similar approach.[5] That is, define a discontinuous function, which you derived in part (a). Then for all values `t(i)` in your vector `t`, define the analytical solution using:

```
if t(i) <= t_Open
    yexact(i) = here use your discontinuous solution
else
    yexact(i) = ...
end
```

Problem 4

(a) Repeat Problem 3 using Matlab's built-in ODE-solver `ode45`. Plot the exact and numerical solutions in the same figure.

(b) Compute the error for all $t \in [0,\ 4\,\text{sec}]$ and plot it. (See *Technical notes* below.)

Save the error in a file (see Problem 3(b)). This error along with that for the cRK solution will be plotted in the next problem.

Technical notes:

1. To learn the syntax of `ode45`, type `help ode45` at Matlab's prompt. (I also strongly recommend that you see the example on the last page of my article on ODEs for Encyclopedia of Ecology, found on the course webpage just above the rubric on Matlab Resources.) As the function `f` for the `ode45`, supply the function file referred to in *Technical note 1* for Problem 3.

2. Make sure that you define the time span correctly: that is, you should not help `ode45` to decide where the discontinuity of your function is. Read the description of the command's syntax carefully. Using an incorrectly defined time span will result in score reduction.

3. To compute the error, you must first re-compute the analytical solution *on the time-vector output by* `ode45`. Follow the approach described in *Technical note 2* for Problem 3.

Problem 5 (worth **0.5 point**)

In a *single figure*, plot the global errors of the numerical solutions obtained in Problems 3 and 4. State how these methods perform for this ODE relative to one another.
Hint: Load the files with saved errors and the time-vectors from Problems 3 and 4 (one file at a time) using the command `load`. Plot both errors in the same figure.

Bonus for Problem 5 (worth **1 point**, given *only* for a detailed investigation)

In Problem 4, you found the solution (and the error) with default tolerances of `ode45`. To see the effect of setting different values for the relative and absolute tolerances, learn how to change them from the example mentioned in *Technical note 1* for Problem 4 (you may also learn how Matlab's `odeset` works).

Next, find how they are related to the error (which one: global or local? — please answer this question) either in Matlab's documentation or in a formula found at the top of p. 12 of the paper by Shampine and Reichelt posted after Lecture 2. In your solution to this problem, you must state what these tolerances mean.

Finally, for the solution at hand, devise a way to check this formula systematically. To that end, fix one of the tolerances and vary the other; then *explain* how the results you observe agree (or disagree)

[5] This is just one possible approach. It is fine if you use another, as long as it works.

with that formula. Then repeat by switching the roles of the two tolerances (i.e., vary the former while fixing the latter).

Bonus-1 [6] (worth **2 points**)
Solve the IVP in Problem 3 above by the Runge–Kutta–Fehlberg method (RKF), assuming the accuracy $\varepsilon_{\text{loc}} = 10^{-3}$ mph. Use $\kappa = 0.8$ (κ is defined in Sec. 2.2). Plot both the exact and numerical solutions in the same figure. Compute the global error, as well as the error controlled by the RKF, for all $t \in [0,\, 4\,\text{sec}]$. Plot each of these errors in its own figure.

Finally, compare the local errors of the RKF, cRK, and `ode45` by plotting them together; do not forget to label which curve represents which method.

Technical notes:
0.

Even though the RKF is a method, just like cRK or ME, for which you have previously written Matlab functions,

do not write the RKF code as a function!

The reason is that coding this method is not as straightforward as the previous methods, because it involves a logical decision loop. Therefore, you will most likely have to debug your code to make it work correctly. And, while debugging a "regular" Matlab script is straightforward:[a] you can simply enter the name of the variable that you want to examine at the prompt in the workspace, — it is **not straightforward to "interrogate" a function** in this way because functions keep their variables "to themselves," so to speak, i.e., they are not available in the workspace.

[a] This does not mean 'easy' or 'simple', though.

1. In order to compute the error for all t, you will have to compute the exact solution on the grid which you will create while computing the numerical solution. So, at each step, record the computed value of t at this step.
2. In the codes you have written so far, you knew the step size h and so knew exactly how many steps you had to make to obtain your numerical solution. This allowed you to use a `for`-loop in your ODE-solver (see, e.g., `hw1_EX_Euler_callsfun.m`). In this problem you do not know the number of steps and hence cannot use a `for`-loop. Use either a `while`- or `if`-loop instead. The calculations should exit that loop when the total computed time exceeds 4 seconds.
3. Since coefficients in the RKF method are cumbersome, you first need to make sure that you will have programmed them correctly. So, at first, just to verify your coefficients of the RKF method, do *not* include the h-controlling if-loop in your code; that is, just assume some constant step, say $h = 0.2$.
4. The algorithm of adjusting h, described in Sec. 2.2, works well if coefficients vary *smoothly* with x. However, *in this problem*, we focus on the situation where they change *abruptly*. Then, the following bad behavior of the algorithm can occur. Suppose that at iteration i, your h_i is so small that the computed local truncation error (LTE) $\epsilon_i \ll \varepsilon_{\text{loc}}$. Then at iteration $(i+1)$, step size h_{i+1} may be adjusted to increase so much that the computed LTE ϵ_{i+1} becomes *much greater* than ε_{loc} if an abrupt change of coefficients occurred between x_i and x_{i+1}. This will force h_{i+1} to be re-adjusted to be so small that $\epsilon_{i+1} \ll \varepsilon_{\text{loc}}$ again. Then the entire cycle "h_{i+1} too small" $\Rightarrow$ "h_{i+2} too large" $\Rightarrow$ "h_{i+2} re-adjusted to be too small" may repeat indefinitely, leading to a numerical overflow.

To prevent this from happening, pre-define the minimum and maximum step sizes, say $h_{\min} = 10^{-6}$ and $h_{\max} = 0.4$, and include provisions in your if-loop for h to always remain in the interval $[h_{\min},\, h_{\max}]$.

Bonus-2 (worth **0.5 point**)
Compare the speeds of Runge–Kutta–Fehlberg and `ode45` methods. Specifically, do the following. First, put your codes for each of these methods into a loop, so that the same calculation is repeated 200 times

[6] The following applies to any Bonus problem assigned in this course. (i) A Bonus problem is considered to be worth one regular problem unless stated otherwise. (ii) A Bonus problem must be done mostly correctly to receive credit; that is, no extra credit for it will be given if its solution has major errors or gaps.

(this is needed for statistical averaging, since the computational speed will fluctuate from run to run). Second, use any of the following 3 commands: `etime`, `cputime`, or `tic` and `toc`. Examples of their usage can be found by typing '`help tic`', etc. Which of the two methods appears to be faster? Remember that, as always, to receive full credit, you must submit your code(s) to me.

HW # 3
Due: 02/07/25

Problem 1
Using Taylor expansions of y'_{i-1} and y'_{i-2} about $x = x_i$, verify that

$$\frac{y'_i - 2y'_{i-1} + y'_{i-2}}{h^2} \;=\; y'''_i + O(h)\,.$$

Problem 2
Find the coefficients b_{-1}, b_0, b_1 in

$$Y_{i+1} = Y_i + h(b_{-1} f_{i+1} + b_0 f_i + b_1 f_{i-1})$$

that produce a 3rd-order method (called the 3rd-order Adams–Moulton method). Use a technique from either Secs. 3.1 or 3.2 (your choice).
Note: If you use software (e.g., Matlab, Mathematica, Wolfram Alpha) to solve the linear system for the coefficients b, please **attach a printout**.

Problem 3
Verify that method (3.22) is indeed of third order accuracy.
Hint 1: What should you show about its local truncation error then?
Hint 2: You should assume that all Y_{i-m} with $m \geq 0$ are known exactly, and then use the Taylor expansion

$$Y_{i-m} \equiv y(x_i - \Delta x) = y_i - \Delta x\; y'_i + \frac{(\Delta x)^2}{2} y''_i - \ldots, \qquad \Delta x = mh\,.$$

Here you will need to decide how many terms in this expansion you should keep.

Problem 4
This problem has two parts:
(i) The programming part is stated first. It is a Bonus assignment, worth **1.5 points**.
(ii) Some theoretical questions are stated in the *Technical notes*; answering them is the *mandatory part*, worth **0.5 points**.

Use the P–C method given by Eqs. (3.33), (3.39), and (3.40) to solve

$$y' = \sin y, \qquad y(0) = 1\,, \qquad x \in [0,\, \pi]\,.$$

Select the step size so that the local truncation error, given by (3.39), be at most $\varepsilon_{\rm loc} = 10^{-4}$. Provide an explanation for your choice of the step size (see below).

Plot your solution. Also, plot the estimate for the error (starting with $i = 2$), deducing it from Eq. (3.39), and verify that the above requirement on the error to be less than $\varepsilon_{\rm loc}$, is indeed satisfied.

Technical notes:

1. You need Y_1, in addition to Y_0, to start the method. Which of the known methods will you need to use to obtain Y_1? Please explain.
Hint: This issue is discussed at the end of Sec. 3.6.

2. Regarding the requirement about the step size:
As repeatedly stated in Lecture 3, it would be difficult to adjust the step size as you are running the calculation. Then, you need to use Eq. (3.37) to estimate the step size required for the error of the P–C method not to exceed a given value. To that end, you will need an *estimate* (in fact, the upper

bound) for $|y'''|$, which you can obtain using the very special form of the function $f(x, y)$ in the given ODE. Review Eq. (1.7) and then Appendix in Lecture 1.
Present a derivation of an upper bound for $|y'''|$ following the above guidelines.

Problem 5
Derive the $O(h^3)$-term in Eq. (3.35).
Hint: There are several ways to do so; for example, one can follow steps of the derivation of the 3rd-order Adams–Bashforth method in Sec. 3.1.
However, here you are required to follow a more systematic method. Namely, substitute the Taylor expansion of $f_{i-1} \equiv f(x_i - h)$ in Eq. (3.5) and compare the result with the Taylor series of the exact solution.

HW # 4
Due: 02/14/25

Notes to all problems in this homework assignment:
1. The stability is implied with respect to the model problem (4.15) of Lecture 4.
2. You must **attach printouts** of all codes/commands that you used to plot stability regions.
3. To plot a stability region, you should *not* define h and λ separately. Instead, define variables $z_R \equiv h\,\lambda_R$ and $z_I \equiv h\,\lambda_I$, each of which is within some range. You may start with $[-2, 2]$ for the range and then adjust it to make your plot look both presentable and informative.
4. You do not need to separate the real and imaginary parts of a complex number by hand. Instead, use commands `Re`, `Im`, `Abs` in Mathematica/Wolfram Alpha or `real`, `imag`, `abs` in Matlab.

Problem 1
For the Modified Euler method, obtain Eqs. (4.21) and (4.22). Then use Mathematica/Wolfram Alpha (using command `ContourPlot`) or Matlab (using command `contour`) to generate the graph of the stability region boundary given by Eq. (4.22).

Technical note:

If you use Matlab's `contour`, you will first need to create your matrix $\rho(z)$ (see below) on a matrix `X,Y`-grid, created out of vectors `x` and `y` with command `meshgrid`. Above, ρ is the expression for the amplification factor, defined in Eq. (4.21), and `z≡X+i*Y` is defined in Note 3 at the beginning of this HW.

Problem 2 (worth **0.5 point**)
Use inequality (4.23) to obtain the bound (4.24) for the stability of the cRK method when $\lambda < 0$ (i.e., is real). *Hint:* In this case, you do *not* need Mathematica. A simple plot in Matlab will suffice.

Problem 3 (worth **0.5 point**)
Show analytically that the region of stability for the Midpoint method coincides with that for the Modified Euler method.
FYI: If you are curious why this is so not technically, but conceptually, re-read the Remarks in Sec. 4.3.

Problem 4 (worth **0.5 point**)
Use Eqs. (4.27) and inequalities (4.30) to plot the boundary of the stability region of the 2nd-order Adams–Bashforth method (3.5).

Technical note:
If you need to show two graphs together in Mathematica, the procedure is the following. Name each plot as follows: `p1=ContourPlot[ ...here goes your command for graph 1 ... ]`, and similarly for `p2`. Then, on a new line, type `Show[p1,p2]`.

Problem 5
Obtain the analog of Eqs. (4.26) and (4.27) for the P–C method (3.33) of Lecture 3.
Plot its stability region following the suggestions of Problem 4. Your graph should have the shape of a heart pointing to the left.

Is this stability region larger or smaller than that of the 2nd-order Adams–Bashforth method?
Try to make an educated guess about how, in general, the stability region of a P–C method is related

to the stability regions of its predictor and corrector equations (see next sentence).
Note regarding the stability region of the corrector equation: You will obtain it in Problem 6. So, wait until then with answering the above question.

Problem 6

Find *analytically* (i.e., without Mathematica or Matlab) the stability region of the Modified Implicit Euler method (3.43) and make a sketch of this region. Based on the definition of an A-stable method, give a explain why your result implies that the Modified Implicit Euler method is A-stable.

Note: You will need to use this property of the modulus of a complex number: $|z_1/z_2| = |z_1|/|z_2|$.

Problem 7

Solve the I.V.P.

$$y' = -20y, \qquad y(0) = 1$$

with $h = 0.125$ up to $x = 1.5$ using the following three methods: (a) simple Euler, (b) cRK, and (c) implicit Euler. Plot your result in (a). In a separate figure, plot your results in (b) and (c) along with the exact solution. Which method, the 4th-order (b) or the 1st-order (c), gives the more accurate solution in this case?

Without doing additional simulations, what do you expect to change in these results if you use $h = 0.15$ instead of $h = 0.125$? Write a brief but coherent paragraph explaining your answer.

Bonus-1

As it is shown in the notes, the Leap-frog method, introduced in Lecture 3, is unstable for $\lambda < 0$. Now, construct a P–C method where the Leap-frog method is used as the predictor and the trapezoidal rule is used as the corrector (as in method (3.33)). Repeat Problem 5 for this new P–C method. (Your graph should look qualitatively similar to the stability region of the 3rd-order Adams–Bashforth method found in the notes.) How does this region compare with the stability regions of the predictor equation (Leap-frog) and of the corrector equation (Implicit Modified Euler; see Problem 6) alone? Does this agree with what you observed in Problem 5?

Bonus-2 (**2 points**)

Use formula (4.51) to confirm the expression for the amplification factor of the Modified Euler method stated in Remark 4. Follow these steps.

1. State the matrix $\mathbf{A}$ and vector $\mathbf{b}$ for this method (this is a QSA in Lecture 2).
2. Compute $\mathbf{A}^2$.
3. To invert the matrix in (4.51), use the (geometric series) expansion, which is valid not only for scalars but also for matrices. Namely, under certain conditions, which we will assume to be satisfied here, one has:

$$(I - \mathbf{M})^{-1} = I + \mathbf{M} + \mathbf{M}^2 + \dots$$

Given your result in Step 2, you will obtain a *finite sum* instead of an infinite series above.

4. Finally, finish the calculation as per (4.51).

Bonus-3

This problem builds on your work in Problems 5 and 6 to illustrate the idea stated in Appendix 4.
Plot the stability region of method (4.66) with $k = 2$.
State how it compares with the region you obtained in Problem 5. Discuss whether your answer agree, in the way described in Appendix 4, with what you have found in Problems 5 and 6?
For an additional **0.5 points**, explore how the region changes for $k = 3$ and then $k = 4$. You may want to zoom in on the vicinity of the origin and, in particular, along the imaginary axis. Again, discuss how this corresponds to your earlier results.