MATH 337.A – Numerical Differential Equations

Spring 2022

HW # 0

Due: 01/24/22

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: 01/28/22

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.

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.[2]

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.

Verify your answer by solving the above IVP for $x \in [0,\ 1]$ [3] 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 $h = 0.1$.

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] 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.

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

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 α will be found from your analytical solution and the given terminal speed.

HW # 2

Due: 02/02/22

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 erros 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 **0.5 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).

Make sure to discuss/conclude what you learned about the effect of these tolerances on the solution/error. Is this effect the same for continuous (Problem 2) and discontinuous (Problem 3) function `f`?

Bonus-1 [6] (worth **2 points**)

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

[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.

Solve the IVP in Problem 3 above by the Runge–Kutta–Fehlberg method, 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. Also, save the global error in a file (see Problem Bonus-3 below, where you will be asked to plot that error).

Technical notes:

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, predefine 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 (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? *Note*: To receive full credit for this problem, you must submit its code(s) to me.

Bonus-3 (worth **0.25 point**)

Compare the error in your RK-Fehlberg method with the errors from Problem 5 above.

HW # 3

Due: 02/09/22

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 $\varepsilon_{\text{loc}} = 10^{-4}$. 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 ε_{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 in a Remark 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: 02/18/22

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.

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 2. 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).

HW # 5

Due: 02/28/22

Problem 1

In the three examples given below, write the given higher-order ODE(s) as a system of first-order ODEs. Try to use variables with subscript notations, as illustrated in (5.9).

(a) The equation for the charge on the capacitor in an electric circuit:

$$L\frac{d^2Q}{dt^2} + R\frac{dQ}{dt} + CQ = \mathcal{E}\,;$$

here Q is the charge, and L, R abd C are the inductance, resistance, and capacitance of the circuit.

(b) The equation for the deflection of a loaded beam from the horizontal axis:

$$E\frac{y''}{(1+(y')^2)^{3/2}} = M(x)\,;$$

here E is the beam's elasticity modulus, Y is the deflection, and M is the bending moment.

(c) The equations of the Kepler two-body problem (5.30):

$$q'' = -\frac{q}{(q^2+r^2)^{3/2}}, \qquad r'' = -\frac{r}{(q^2+r^2)^{3/2}},$$

where q and r are the Cartesian coordinates of a certain radius vector relative to the center of mass of the bodies.

Problem 2

Write down the explicit (i.e., component-by-component) equations for (a) the Midpoint method and (b) the cRK method for a system of two ODEs $\vec{\mathbf{y}}' = \vec{\mathbf{f}}(x, \vec{\mathbf{y}})$.

Note 1: This is *not* a coding problem. You just need to write these equations on paper. You will be asked to code them in a later problem.

Note 2: For part (a), notice from Eq. (5.5) that *both* $\overline{Y^{(1)}}$ and $\overline{Y^{(2)}}$ needed to be computed before computing $Y_{n+1}^{(1)}$ and $Y_{n+1}^{(2)}$. Proceed similarly for the Midpoint method.

Note 3: In part (b), should you compute $k_1^{(2)}$ before $k_2^{(1)}$ or vice versa? See Note 2.

Problem 3

(a) mandatory assignment (**0.5 point**)

Show that the local truncation error of the simple central-difference method (5.12) equals

$$\frac{h^4}{12}\frac{d^4y(x_n)}{dx^4} + O(h^5)\,.$$

See the Note and Hint below.

(b) Bonus assignment (**0.5 point**)

Show that the local truncation error of Numerov's method (5.18) equals

$$-\frac{h^6}{240}\frac{d^6y(x_n)}{dx^6} + O(h^7)\,.$$

Note: Recall that in n the calculation of the local truncation error at the $(n+1)$st node, the error at all previous nodes must be set to equal zero.

Hint: Similarly to Problem 5 in HW 3, note that $f_{n\pm1} = f(x_n \pm h, y(x_n \pm h)) \equiv f[x_n \pm h]$, where $f[x] \equiv f(x, y(x))$. Now use the Taylor expansion for $f_{n\pm1}$ written in the above form. Now recall that $f = y''$. What can you then say about df/dx, etc?

Problem 4

Show that the Verlet method (5.32) is a second-order method. Follow the steps of a similar derivation for the Modified Euler method, presented in Sec. 5.1.

Note: You may be tempted to let $f(Y_{n+1}) = f(y_{n+1}) \equiv f_{n+1}$ and then proceed as in Problem 3. However, this step requires a more careful justification than presented in the previous sentence because $Y_{n+1} = y_{n+1} + O(h^3)$. You may either combine the last equation with the treatment of f_{n+1} in Problem 3, or follow the derivation of the error of the Modified Euler method, as originally suggested. In either case, your derivation will involve f_y and not just $f'[x]$.

Problem 5

In the notes, we derived the Verlet method for Eqs. (5.21), (5.22) by first going over the half-step $[x_n, x_{n+\frac{1}{2}}]$ with method (5.4) and then going over the remaining half $[x_{n+\frac{1}{2}}, x_{n+1}]$ with method (5.3). Let us refer to the resulting Verlet method (5.32) as Verlet-1. Use the same ideas to derive a method by reversing the order of (5.3) and (5.4) (i.e., apply (5.3) over $[x_n, x_{n+\frac{1}{2}}]$ and then (5.4) over $[x_{n+\frac{1}{2}}, x_{n+1}]$). We will refer to this method as Verlet-2.

Problem 6 (worth **2.5 points**)

Consider the equations of a simple harmonic oscillator (5.25), i.e.

$$y'' = -y, \quad y(0) = 0, \quad y'(0) = 1\,.$$

(a) Write out its analytical solution (consult any ODE or Physics-I textbook).

Solve this equation numerically using the following methods:

(b) Verlet-1 (for Verlet-2, the results will be similar); (c) Modified Euler (5.5); (d) cRK; (e) simple central-difference (5.12) with (5.17); (f) Matlab's `ode45`.

In all cases, run the simulations until $t = 1000$. To have a fair comparison among the methods, use $h = 0.2$ for the second-order methods and $h = 0.5$ for the cRK method. For the `ode45`, obtain *two* solutions with different values of the absolute tolerance: 0.002 and 0.003. This is done by defining `options=odeset('AbsTol',0.002)` and then calling `ode45` with the last argument `options`: `[t_ode45, y_ode45] = ode45( ... , options)`.

Your output for this problem should contain:

- phase plots of the numerical solutions (one per method);
- plots of the error in the Hamiltonian versus time;
- a table (handwritten is okay) summarizing which of the methods nearly conserve the Hamiltonian

and which methods do not;[7]

• for those methods that do not conserve the Hamiltonian, include in your table two comments stating: (i) Whether the Hamiltonian increases or decreases and (ii) Whether the trajectory in the phase space winds inside or outside the unit circle. (To answer this for some of the methods, you will need to zoom in on some of the phase plots.)

As a benchmark, you should find that the modified Euler method performs the worst for this problem.

Technical notes:

1. For the central-difference method, you should first obtain the array of y-values and then compute v-values from the former array. It is straightforward to do so using the 2nd-order accurate central-difference formula (3.19) for the *first* derivative. It can be used to compute all v-values except the last. The last v-value can be computed using the following 2nd-order accurate approximation:

$$v_n = \frac{y_n - y_{n-1}}{h} + \frac{h}{2}\left(-y_n\right),$$

which was obtained similarly to Eqs. (3.3) and (3.4) in Lecture 3.

2. One way to define a function with a vector output for passing it to `ode45` is shown in the example at the end of my Encyclopedia of Ecology article on ODEs, posted on the course website (towards the end). Another way is:

```
function  z = vectorF(t,y)
z(1) = expression in terms of t, y(1), and y(2)
z(2) = expression in terms of t, y(1), and y(2)
z = transpose(z);   % This is needed to comply with the  ode45's  convention
                    %  that its output must be a column vector.
```

3. Do not compute the Hamiltonians in the same loops as the solutions. It is not wrong, but does not fully utilize Matlab's capabilities of handling vectors. Instead, compute the Hamiltonian using the already computed *arrays* of the y- and v-values.

4. I recommend that you use an excerpt from my code that uses subplots to combine plots showing similar behavior for easier comparison. It is posted under "Codes for examples and selected homework problems" as the first link for HW 5. Please: (i) Note (for future use) how the axis limits are set and (ii) Do not change them.

Plots for the Hamiltonian errors can be organized similarly. Do not forget to label axes.

Food for thought 1: From the comments in your table, notice a correlation between the behavior of the phase space trajectory and the behavior of the Hamiltonian error. This observation will be useful for Problem 7.

Food for thought 2: If you wonder why the solutions of `ode45` with rather similar tolerances, as above, behave *qualitatively* differently, attempt the Bonus-1 problem.

Problem 7

Find an *estimate* for the growth of the Hamiltonian, $H = (1/2)(v^2 + y^2)$, of the numerical solution of the harmonic oscillator model (with $\omega = 1$) obtained with the regular Euler method. Follow the steps described below.

1. For concreteness, consider the IVP of Problem 6, for which you have obtained the analytical solution. (The analysis for other initial conditions is analogous.) Use the result of Problem 6(a) to find the exact value of the Hamiltonian.

2. Write down the formulae for the corresponding numerical solutions obtained with the regular Euler method. The corresponding calculation for Y_n is described in Sec. 5.4.1; it is similar for V_n. Then, with these solutions, calculate an estimate for the numerically computed Hamiltonian.

[7] The following clarifies the word 'nearly'. None of the methods conserve the Hamiltonian exactly. However, you may ask: If the simulations continued for a much longer time, say $t = 100,000$, would the Hamiltonian remain close to the initial value or would it move away from it? In the former case you declare that the method nearly conserves the Hamiltonian; in the latter, it does not.

3. Finally, verify that your answer for $H_{\text{computed}} - H_{\text{exact}}$ agrees with that found from a figure in Sec. 5.3 (for the largest value of x).

Problem 8

In the notes, we showed that the symplectic Euler methods (5.3) and (5.4) approximate very well the energy of a system without friction-like forces. Now, investigate the question of whether these methods still produce accurate approximations to the energy when a small friction *is* present.

Consider the equation of a slightly damped harmonic oscillator

$$y'' = -2\gamma y' - \omega_0^2 y, \quad y(0) = 0, \quad y'(0) = 1\,,$$

where $\gamma \ll 1$ represents a small friction coefficient. Its exact solution is $y = \frac{1}{\omega} e^{-\gamma t} \sin(\omega t)$, where $\omega = \sqrt{\omega_0^2 - \gamma^2}$. It decays exponentially, but slowly, since $\gamma \ll 1$.

Use the regular Euler and one of the two symplectic Euler methods (it is up to you which one – the results will be similar) to simulate the damped oscillator with $\omega_0 = 1$ and $\gamma = 0.015$ up to $t = 50$. For each of the methods, use three different values of step size: $h = 0.01$, $h = 0.03$, and $h = 0.05$. For each of the above values of h, you need to present 3 plots. Plot 1 should contain the phase plane plots of the exact solution and the solution obtained by the regular Euler method. Plot 2 should contain the phase plane plots of the exact solution and the solution obtained by the symplectic Euler method. Plot 3 should contain the error in the total energy, $E = (v^2 + \omega^2 y^2)/2$, for both (regular and symplectic) Euler methods.

Using an analysis essentially repeating that in Sec. 5.4.1 in the Notes, explain *quantitatively*[8] why the *regular* Euler solution $\{Y_n, V_n\}$ behaves, relative to the exact solution, in the observed way for each given h. In particular, you must explain why the regular Euler solution changes from decaying to growing as h is varied.

Conclude whether it is reasonable to use symplectic methods for problems with *small* friction (or any other source of damping). Is it a good idea to use general-purpose methods (like the regular Euler) for the same task?

Technical notes:

1. The following approach may save you some typing when preparing this code. Instead of typing in each new value of h (and also γ and $t_{\max}$ in part (b)) inside the code, you may specify these variables using the command `input`:

```
h=input('enter the step size:  h = ');
```

(The text you write inside the `input` can be arbitrary, but what you enter in the command window when prompted should, of course, be your desired value of h.

2. To save paper, please use the command `subplot` with a 1×3 or 2×2 tile of panels. Use `axis equal` for all phase plots and do not forget to label your axes and indicate what h you use in each series of plots.

3. For the analysis of the regular Euler solution, you will need to first find eigenvalues of some matrix. You can do so either by hand of with Mathematica (in the latter case, attach your printout). Keep in mind the obvious fact that $\sqrt{-\omega^2} = i\omega$; also, neglect terms $O(\gamma^2)$.

Problem 9 (worth **1.5 points**)

Follow the lines of the stability analyses for the regular and symplectic Euler methods (see Sec. 5.4) and obtain the stability regions for the Modified Euler method (5.5) and Verlet method (5.32). Use (5.26) as the model problem for both methods. (You may denote $\omega y = \tilde{y}$ to simplify notations.) In your analysis, assume that ω is complex.

Note: Strictly speaking, you should be able to give the answers for both methods without doing any calculations, because these answers can be found in Lectures 4 and 5. However, I want you to do those calculations *explicitly*[9] in order to get some practice with stability analysis of a system of ODEs.

[8] This explanation is worth *half of the point* for this problem.

[9] as opposed to referring to the 'important conclusions' stated in Sec. 5.4

Technical notes:

1. For both ME and Verlet, you need to obtain a matrix equation relating $(\omega Y, V)^T_{n+1}$ to $(\omega Y, V)^T_n$ (here superscript 'T' stands for transpose).

- For Verlet, this work should be similar to that leading to Eq. (5.58), although your matrix will be different.
- For ME, you can do a similar calculation or, alternatively, you can follow the steps of Eqs. (5.48)–(5.50). In the latter case, you do *not* need to repeat (5.46), but you *do* need to present your calculation of D.

2. Once you have the matrix form, you need to get ρ's from it. You can do it either by hand or by Mathematica (see *Help* for `Eigenvalues`). You must include either your handwritten work or a printout from Mathematica.

3. Once you have the ρ's, you do *not* need to make a contour plot as in HW 4. Instead, for Verlet, you can simply explain how the result is the same as (or similar to) the result for ρ's of symplectic Euler. For ME, you need to explain how your result for ρ's is the same as that in Lecture 4. To do so, recall how ω in Lecture 5 is related to λ in Lecture 4. (If you forgot, find the answer stated explicitly in Sec. 5.4.1.)

Problem 10

Solving the quadratic equation (5.62) by hand, verify the statements found in each line of (5.61), assuming that ω is *real*.

Note: This quadratic equation arises in many different contexts, including those which will later appear in this course. You may also need to analyze roots of this equation if you take a Qualifying Exam on Numerical Methods.

Hint 1: Denote $2 - (h\omega)^2 \equiv 2k$ and express the solution in terms of k. You will first need to find for what k the method is stable (see below), and then translate those bounds for k into bounds for $h\omega$.

Hint 2: To obtain bounds for k, note that ρ is not always real. Then you need to consider separately two cases (corresponding to different intervals for k), one where ρ is real and one where it is complex. For each of these cases, you need to find when $|\rho| \le 1$.

- Recall that the modulus of a complex and of a real number are handled differently. E.g., you would not use the same method to find $|2 + i\sqrt{3}|$ and $|2 + \sqrt{3}|$. *Make sure to clearly explain* how you compute the modulus in each case in your solution.

Hint 3: Recall that since quadratic equations have two roots, you will need to investigate the bounds for *both* roots, i.e., find when $|\rho_1| \le 1$ *and* $|\rho_2| \le 1$. For the case when ρ is real, each of these conditions yields a double inequality, and so you may need to investigate *four* conditions (i.e., two inequalities for each of the two roots).

However, note that once *any one* of these four conditions is violated, the method is not stable: see Eq. (4.30) in Lecture 4. Then you no longer need to consider the remaining conditions.

Problem 11

Consider a system of two linear ODEs:

$$\frac{d}{dt}\begin{pmatrix} x(t) \\ y(t) \end{pmatrix} = A \begin{pmatrix} x(t) \\ y(t) \end{pmatrix}, \qquad A = \begin{pmatrix} 100 & 99 \\ 99 & 100 \end{pmatrix}.$$

(a) Would you call this system numerically stiff?

(b) Will your answer change if both entries of A with '100' are replaced with '-100'?

Please explain.

Bonus-1 **(3 points)**

Explain why the solutions of `ode45` with rather similar tolerances used in Problem 6 behave *qualitatively* differently.

Hint: The method used in `ode45` is not the RK–Fehlberg method, studied in Lecture 2, but its relative

called the Dormand–Prince method. Its Butcher's tableau is available, e.g., on Wikipedia. Using it and formula (4.51) from Lecture 4, plot the boundary of a stability region for this method. (Of course, have Matlab invert the matrix in the formula.) Then, examine the behavior of this boundary in the vicinity of the imaginary z-axis.

Your explanation must be sufficiently detailed and coherent to receive credit. If you are not sure what "sufficient" is in this case, ask the instructor.

Bonus-2

(a) **(0.25 point)** Program the Verlet-2 method for the harmonic oscillator of Problem 6. Verify that its error is opposite of the error of the Verlet-1 method.

(b) **(1.25 points)** Combine analytically Verlet-1 and Verlet-2 methods into a Super-Verlet, using the same approach as was used in the Notes to combine the two symplectic Euler methods into a Verlet method. Given the observation stated at the end of part (a), the Super-Verlet should have the order higher than two. Verify this prediction for the harmonic oscillator equation. For that, first show that the error of the Super-Verlet method is much smaller than that of Verlet-1 and -2, and then numerically investigate how it scales with h, using at least three values of h.

Bonus-3

(a) **(1 point)** Apply either of the Verlet methods to the Kepler two-body problem (5.35) (see also Problem 1(c) above). Use $h = 0.1$ and $t_{\max} = 500$. As the initial condition, use

$$q(0) = 1 - \mathsf{ecc}, \quad r(0) = 0, \quad Q(0) = 0, \quad R(0) = \sqrt{\frac{1+\mathsf{ecc}}{1-\mathsf{ecc}}} \qquad \text{for } \mathsf{ecc}{=}0.6\,,$$

which corresponds to the exact solution being an ellipse with eccentricity 0.6. Compute the conserved quantities: the Hamiltonian (5.36), the angular momentum (5.37), and the components of the Runge–Lenz vector (5.38), and then plot them versus time.

Now plot the trajectory defined by Eqs. (5.35) (recall the meaning of q and r). What effect does the non-conservation of the Runge–Lenz vector appear to have on the numerical solution of (5.35)? If you are not sure, reduce the step size (but keep $t_{\max}$ the same) and see what this does to your numerical solution.

Technical notes: Think how you can organize your plots to make changes in these quantities well visible. Also, and irrespective of the request in the previous sentence, please use the command `subplot` to save paper.

(b) **(1 point)** Apply the Super-Verlet method to the Kepler two-body problem — Problem 1(c) — and compare its performance with that of a Verlet method. In particular, how are the amplitudes and periods of the oscillations of the Runge–Lenz vector found by these two methods seem to be related? Answer this question for $h = 0.1$ and $h = 0.05$.

HW # 6

Due: 03/01/22

Problem 1

Verify that the *homogeneous* versions of Problems I and III in the Notes have nontrivial solutions, while the homogeneous version of Problem II has only the trivial solution. (The homogeneous version of a BVP has *both* the r.h.s. of the equation *and* the boundary conditions all equal to zero.)

Hint: Solve for the constants A and B in the counterpart of the solution (6.6).

Problem 2

Find the solution of Problem I with the π^2 replaced by $(\pi + \epsilon)^2$, where $\epsilon \ll 1$. *Based on your answer*, explain why such a problem is ill-posed (or, equivalently, ill-conditioned).

Suggestion: If you are not sure, re-read what it means for a BVP to be ill-conditioned, at the end of Lecture 6. If you still have difficulty answering the question, pick a couple of small values for ϵ and plot the solution you obtained.

Note: Use trigonometric identities to simplify $\sin(\pi+\epsilon)$ and $\cos(\pi+\epsilon)$. Then, use the Maclaurin series for each of them and keep only the first term in each expression.

HW # 7
Due: 03/14/22

Compare your plots for all Problems with those found under "Codes for examples and selected homework problems" on the course webpage.

Problem 1
Solve the BVP

$$y'' + xy' - 3y = 3x, \qquad y(0) = 1, \quad y(2) = 5$$

by the shooting method. Use the Modified Euler method with $h = 0.1$ as the IVP solver. Plot your solution.

Problem 2
Solve the BVP

$$x^3 y''' + xy' - y = -3 + \ln x, \qquad y(1) = 1, \quad y'(2) = \frac{1}{2}, \quad y''(2) = \frac{1}{4}$$

by "shooting" from the left end point and using the example given in the notes. Use the Modified Euler method with $h = 0.02$ to solve the IVPs. Plot your solution so as to show that all the boundary conditions are satisfied.

Problem 3
Repeat the previous problem while "shooting" from the right end point of the interval. (Use the same IVP-solving method and the same h as in Problem 2.)
Hint: Set up the corresponding auxiliary IVPs of the form (7.27). Note that you will only need *two* such IVPs. Correspondingly, you will have only one parameter, θ, to determine.

Problem 4
Solve the nonlinear BVP

$$y'' = \frac{y^2}{2+x}, \qquad y(0) = 1, \quad y(2) = 1$$

following the outline given in Sec. 7.5. Find (and, of course, plot) the solutions of this BVP corresponding to both $\bar{\bar{\theta}}$ and $\bar{\theta}$. Use 10^{-3} as the tolerance for the value of $y(2)$.
Note: When you are finding the solution corresponding to $\bar{\theta}$, the slope of the solution at the left end point is quite large. Therefore, you may want to use a smaller h than you did when finding the solution corresponding to $\bar{\bar{\theta}}$.

How many iterations do you need to obtain each of these solutions?

Problem 5
Solve the eigenvalue problem[10]

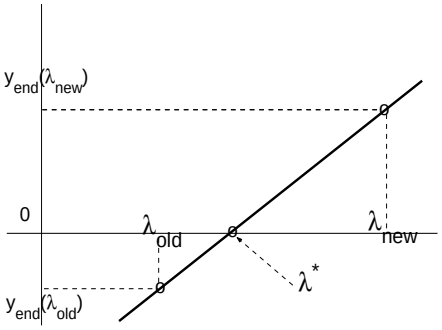
$$y'' + (2\mathrm{sech}^2 x - \lambda^2) y = 0, \qquad x \in (-\infty, \infty), \quad y(|x| \to \infty) \to 0\,.$$

As is explained in the notes, solve the IVP on the interval $[-R, R]$ for some reasonably large R (say, $R = 10$). Choose $y(-R) = \exp[-c\,R]$ and $y'(-R) = \lambda \cdot y(-R)$, where the constant c is of order one (so you may just take $c = 1$.) Then follow the algorithm outlined in the notes. Namely, take the step in λ to be $\Delta\lambda = 0.01$. Start with $\lambda_0 = 0.1$ and then increase it as $\lambda_i = \lambda_0 + i \cdot \Delta\lambda$ until two consecutive values $y(R, \lambda_i)$ and $y(R, \lambda_{i+1})$ become of opposite sign. Thus, you do *not* need to keep your solution $y(x)$ for all values of λ but only for the last two, λ_{old} and λ_{new}.

[10]Recall that when solving an eigenvalue problem $M\vec{\mathbf{v}} = \lambda\vec{\mathbf{v}}$ in Linear Algebra, you needed to find *both* the eigenvalue λ and the eigenvector $\vec{\mathbf{v}}$ that satisfy this problem. Similarly, here, you need to find both λ and $y(x)$.
This eigenvalue problem, with $2\mathrm{sech}^2 x$ possibly replaced with some other function $U(x)$, is known in Quantum Mechanics as the Schrödinger equation.

Technical notes:
1. Although you may use many of the IVP methods, I suggest that you use Matlab's `ode45` so as to easily control its accuracy (see Problem 6 in HW 5). In fact, if you set the accuracy too low (you may experiment how low), you will get a conspicuous "burst" at the right end of your eigenfunction.
2. In general, the true value of the eigenvalue, $\lambda = \lambda^*$ where $y(R, \lambda^*) = 0$, will lie between two trial values $\lambda_i = \lambda_{\rm old}$ and $\lambda_{i+1} = \lambda_{\rm new}$. Since the solution $y(x)$ at a "wrong" (even if it just slightly wrong) λ will have a nonzero component proportional to $(\lambda - \lambda^*)\exp[\lambda R] \gg 1$ (for $R \gg 1$), then it is important to determine λ^* as closely as possible. To this end, you may approximate the curve $y(R, \lambda)$ on the small interval $[\lambda_{\rm old}, \lambda_{\rm new}]$ by a straight line (see the figure below). Then the value λ^* where this straight line crosses zero is given by:

$$\begin{aligned} \lambda^* &= \lambda_{\rm old} + \frac{\lambda_{\rm new} - \lambda_{\rm old}}{y(R,\lambda_{\rm new}) - y(R,\lambda_{\rm old})}\,\big(0 - y(R,\lambda_{\rm old})\big) \\ &= \frac{\lambda_{\rm old}\, y(R,\lambda_{\rm new}) - \lambda_{\rm new}\, y(R,\lambda_{\rm old})}{y(R,\lambda_{\rm new}) - y(R,\lambda_{\rm old})}\,. \end{aligned}$$



Having found λ^*, do one more shooting at $\lambda = \lambda^*$ to determine an approximation of the eigenfunction.

Bonus-1
Solve the BVP

$$y'' = 30^2\,(y - 1 + 2x), \qquad y(0) = 1, \quad y(1.62) = -2.24$$

by the usual (i.e., not multiple) shooting method. Use any appropriate IVP-solving method and any reasonable value for h. Plot your numerical solution along with the exact solution, found in the notes.
Note: This problem is not difficult at all, but is made an extra credit so as not to overwhelm you. Its purpose is not to make you learn a new technique but rather to illustrate a phenomenon described in Section 7.4.

Bonus-2 (**1.5 points**)
Repeat Bonus-1 by using the simplest version of the multiple shooting method, whereby you divide the interval [0, 1.62] into two sub-intervals of equal lengths. Follow the lines of Sec. 7.4. Describe how your result compares with that of Bonus-1.

HW # 8
Due: 03/28/22
Problem 1 (worth **0.5 point**):
Use the Gerschgorin Circles Theorem and the fact that eigenvalues of real symmetric matrices are real to obtain the best estimate for the location of the eigenvalues of the following tri-diagonal matrix:

$$A = \begin{pmatrix} -a & 1 & 0 & . & . & . & 0 \\ 1 & -a & 1 & 0 & . & . & 0 \\ 0 & 1 & -a & 1 & 0 & . & 0 \\ & & & . & . & . & \\ 0 & . & . & 0 & 1 & -a & 1 \\ 0 & . & . & . & 0 & 1 & -a \end{pmatrix},$$

where $a > 0$. In particular, what is the *minimum distance* between an eigenvalue of this matrix from zero?
The result of this problem will be used in Sec. 8.6 and then later in Lecture 13.

Problem 2

Consider a linear BVP

$$y'' + 2(2-x)y' = 2(2-x), \qquad y(0) = -1, \quad y(6) = 5.$$

Discretize it using scheme (8.4) with $h = 1$.

(i) Verify that you obtain a linear system

$$\begin{pmatrix} -2 & 2 & 0 & 0 & 0 \\ 1 & -2 & 1 & 0 & 0 \\ 0 & 2 & -2 & 0 & 0 \\ 0 & 0 & 3 & -2 & -1 \\ 0 & 0 & 0 & 4 & -2 \end{pmatrix} \begin{pmatrix} Y_1 \\ Y_2 \\ Y_3 \\ Y_4 \\ Y_5 \end{pmatrix} = \begin{pmatrix} 2 \\ 0 \\ -2 \\ -4 \\ 4 \end{pmatrix}.$$

(ii) Solve it using Matlab. (To solve a linear matrix system $A\,x = b$, type `x=A\b`.) What do you obtain?

(iii) The result you have obtained in part (ii) occurs because one of the conditions of Theorem 8.3 is violated. What is that condition?

Problem 3 (worth **0.5 point**)

(a) Give an operation count for finding L and U for a tridiagonal matrix, as per Eq. (8.21). (Each arithmetic operation must be counted as one operation.)

(b) Give operation counts for solving the systems in (8.17), as per (8.22) and (8.23).

(c) Give the total operations count for the Thomas algorithm.

Problem 4

Preamble: This Problem itself counts as *extra credit*. **However**, you will need to study the *Technical notes* for it and follow the requirement typeset in boldface in the rule box at the end of this Problem, in all subsequent homework problems that require solution of a tridiagonal linear system.

Assignment: Use the `thomas.m` function, posted under "Codes for examples and selected homework problems", to solve a tridiagonal system $A\vec{\mathbf{y}} = \vec{\mathbf{r}}$ where A has '2' on the main diagonal and '-1' on the two subdiagonals. Take $\vec{\mathbf{r}} = [1,\ -1,\ 1,\ -1,\ \ldots\,]^T$ and $M = 1000$ and 5000.

Now solve the same system using the Matlab's solver (as in Problem 2(ii)). Here, you need to investigate *two* cases: One, when A is constructed as a regular (i.e., full) matrix, and two, when it is constructed as a sparse matrix; see Technical notes below.

Compare the computational times required to solve this system for your code and for the Matlab's solver in those two cases. (Consult the Bonus problem of HW# 2 on how to compare computational times.) In particular, comment *how the computational times scale with* M in each of the three cases considered.

Technical notes:

1. An efficient way to set up a tridiagonal matrix A with the lower subdiagonal, main diagonal, and upper subdiagonal given by respective vectors `a`, `b`, and `c` is this:

```
A = diag(a,-1) + diag(b) + diag(c,1);
```

type `help diag` for more details. Note that `diag` is a *dual-purpose* command: it can both set up matrices with given diagonals and extract the specified diagonals from a given matrix.

Also note that vectors `a`, `b`, and `c` are not all of the same length.

Even though the matrix you have just constructed may appear sparse to you (it has only three diagonals!), it is treated as a full matrix by Matlab; see the next note.

2. A way to define the same matrix A as *sparse* is by using the command `spdiags`. Note that the syntax of this command is different from that of `diag`. Namely, if `a`, `b`, and `c` are *column* vectors of the *same* length M, then

```
A = spdiags(a,-1,M,M) + spdiags(b,0,M,M) + spdiags(c,1,M,M)
```

defines a matrix with vectors: `a(1:M-1)` on the sub-diagonal, `b(1:M)` on the main diagonal, and `c(2:M)` on the super-diagonal.

This command *declares* to Matlab that the matrix A constructed with it is sparse (to check this, type `issparse(A)`). The significance of the sparseness is that Matlab "knows" how to handle sparse matrices more efficiently than full ones.

From now on, i.e. in the remainder of this HW and in all subsequent HWs, whenever you are to solve an equation $A\vec{y} = \vec{r}$ with a *tridiagonal* matrix A, you must do so with either the `thomas.m` code or `spdiags`. If you use just `diags` to solve that equation, your grade will be reduced.

Problem 5

Modify Method 1 of Sec. 8.4 in such a way that it can handle the mixed-type boundary condition at the *right* end point of the interval. Follow the derivation of (8.34) and obtain a counterpart of that equation at the right end point. Include this derivation in your submitted work.

Use your result to solve the BVP

$$(1+x)^2\, y'' = 2y - 4, \qquad y(0) = 0, \quad y(1) + 2y'(1) = 2$$

using the second-order accurate discretization (8.4) (for $n = 1, \ldots, N-1$) of this BVP.

Confirm that your numerical solution has the second order of accuracy by comparing it at different h with the exact solution $y_{\text{exact}} = 2x/(1+x)$. For this, do the following:

(i) Run your code for $h = 0.05$ and $h = 0.025$;

(ii) Plot the error as a function of x;

(iii) Confirm that the maximum error scales as $O(h^2)$.

Technical notes:

1. In this and many other problems where you need to account for boundaries, begin your work by sketching the interval $[a, b]$ (where $x = a$ and $x = b$ are its end point) and labeling points $a = x_0$, x_1, ..., x_{N-1}, $x_N = b$. (You can also include any virtual nodes as needed.) Circle those of these points whose corresponding y_n's you are *not* solving for. This will tell you how many unknowns you will have in your linear system.

2. Do *not* define `x = a:h:b`, as you must have done in HW 7. The reason is that, as you have seen by following Note 1, not all points of the `x`-vector defined as above correspond to your unknowns. Instead, define `xplot = a :h: b` [11] and then define `x` as a subset of `xplot` corresponding only to those points which you will solve for. For example, for Dirichlet boundary conditions, you would define `x = xplot(2 : end-1)`.

3. After defining `x`, define `N` in relation to `length(x)`, as it is defined in Lecture 8. (Please re-read the boxed Technical note for Problem 4 of HW 1.)

4. Do *not* define your `Y`-vector to include the boundary values. By the same token as in Note 2 above, your `Y`-vector should include only those points that you are solving for. Only at the very end will you need to define `Yplot`, which will incorporate your `Y` as a subset.

Problem 6 (**0.5 point**)

Show that if condition (8.42) and the two conditions stated one line below it hold, then the coefficient matrix in Method 2 based on Eq. (8.37) is SDD.

Hint 1: Write the condition that you want to prove in mathematical notations and *then* assume that (8.42) and the other two conditions hold.

Hint 2: Start with (8.37) and multiply it by h/A_2.

Problem 7

Solve the BVP stated in Problem 4 of HW7 using Picard's iterations (Method 1 of Sec. 8.6). Do the calculations for $h = 0.2$ and $h = 0.1$.

[11] the name comes from the fact that the *only* place where you will need this vector is when plotting your solution at the very end

As you remember, the BVP in question has two solutions: one with $y'(0) \approx -0.3$, and the other with $y'(0) \approx -16$. Below you are directed to attempt to obtain both of those solution. Which solution you will obtain will depend on the initial guess that you will use.

To find the first solution, use the intial guess $\{Y^{(0)}\}$ that equals the straight-line segment connecting the given boundary values of y (i.e. the segment $y = 1,\ 0 \le x \le 2$ in this case). Given the shape of that solution, we will refer to it as "shallow".

To find the second solution, take the initial guess being a triangular profile that has slopes of -16 and $+16$ at the left and right end points, respectively:

$$Y^{(0)} = \begin{cases} 1 - 16x, & 0 \le x \le 1\,; \\ -31 + 16x, & 1 \le x \le 2\,. \end{cases}$$

We will refer to this solution as "deep".

For each value of h that you used and for each solution, record how many iterations (if they converge) are required to achieve the tolerance $||Y^{(k+1)} - Y^{(k)}|| \le 10^{-6}$.

For each h, plot *the logarithm to base 10* of the maximum discrepancy between two consecutive iterations, $||Y^{(k+1)} - Y^{(k)}||$, as a function of the iteration number. Your graph should asymptotically (i.e., for small discrepancies) look like a straight line; for this reason, the Picard method is said to converge (when it converges) *linearly*.

Does the number of iterations (or convergence itself) depend on h?

Technical notes:

1.

Many codes from this point on will get increasingly more complex, which may increase the chances that you will make coding mistakes and then will ask for my help with finding those mistakes. Therefore, it is now a good time to remind you of the strong suggestion that I stated in boldface font in HW 1. Namely, **when programming a new numerical scheme for the first time, stay close to the notations in which this scheme is written in the lecture.** If you ask for my help in finding a mistake in your code, I may only be able to provide useful feedback if your code is written in **my** notations. **I will not learn your notations just to help you.**

2. Follow the ideas described in the Technical notes for Problem 5.

Problem 8 (**3 points**)

General guidelines:

Repeat Problem 7 using modified Picard's iterations.

- Use the same two values of h and the tolerance 10^{-6} as in Problem 7.
- Present your results as a table with two columns (one for each h), where you will record the number of iterations required for convergence for each of the cases described in the *Specific guidelines* found below. These cases will ask you to investigate the behavior of the iterations for various values of the parameter c and for different initial guesses.
- From that table, conclude *when* (i.e., for what behavior of the iterations) the dependence of your results on h is most pronounced.
- For one value of h and for one value of c where you observe convergence for each of the two solutions, plot the logarithm to base 10 of the error versus the iteration number. You should observe *linear* convergence of the modified Picard method (when it converges), as you did for the original Picard method.
- Note that for either (i.e., "shallow" or "deep") solution, the optimal values of c can be estimated from (8.75) and the information deduced from the plots obtained in HW7. To that end, one first needs to estimate values of $L^{\pm}$ using (8.74) and the aforementioned plots, a procedure for which we now outline. Denote $\partial f/\partial y = 2y/(2+x) \equiv \mathcal{L}(x, y)$. Estimate $L^{\pm}$ by comparing three values: $\mathcal{L}(0, y(0))$, $\mathcal{L}(1, y(1))$, and $\mathcal{L}(2, y(2))$.

  - For the "shallow" solution, you should find that $L^- \approx \mathcal{L}(2, y(2))$ and $L^+ \approx \mathcal{L}(0, y(0))$, whence $c_{\text{opt, shallow}} = L^{\text{av}} \approx 0.75$ (please verify this on paper).
  - For the "deep" solution, you should find that $L^- \approx \mathcal{L}(1, y(1))$ and $L^+ \approx \mathcal{L}(0, y(0))$, whence $c_{\text{opt, deep}} = L^{\text{av}} \approx -2.9$ (please verify this on paper).

- *Technical note*: Limit the number of iterations in your loop by some large number. For example, you may insert the following statement in your loop:

```
if k > 1000
    disp('iterations do NOT converge !!!!!!!!!!!!!')
    break
end
```

Specific guidelines for the two types of solution:

1. Use the **"shallow" initial guess**, let $c = 100,\ 20,\ 10,\ 1.5,\ 0.75,\ 0,\ -0.5,\ -0.9,\ -1,\ -1.5,$ and record the number of iterations required for convergence.

- Does the value $c_{\text{opt, shallow}}$, estimated above, appear to be indeed optimal? Does this agree with some assumption about the solution found somewhere below (8.75)? (You should answer not simply 'yes' or 'no', but also name that assumption.)
- For one or more of the values of c, be prepared for a surprise. *It is an important part of this assignment — and hence, combined with the similar part for the "deep" solution, is worth 1.5 points — that you figure out what is going on with consecutive iterations of the solution!* If you are puzzled, **"ask" the code what it is doing**: plot the results of individual iterations. For these values of c, write a short descriptive paragraph about the behavior of the iterations. You may attach a plot if that helps to illustrate your description.

Very important note:

> You may be tempted to save time and program you code to run it in a loop for both values of h and for all values of c. **Do not do that!**
>
> In general, **NEVER EVER, EVER** attempt to automate a process whose behavior you do not fully understand!
>
> (Note that in this problem, I have told you that you would encounter an unexpected behavior.) The only way that I know, save divine revelation, for understanding a puzzling behavior is to look at it one case at a time.

2. For the **"deep" initial guess**, let $c = -7.45,\ -6.95,\ -6.45,\ -5.95,$ and then skip to $-4,\ -3.9,\ -3.75,\ -3.62,\ -3.6,\ -3.55,\ -3.5,\ -2.9,\ -2.5$.

- Does the value $c_{\text{opt, deep}}$, estimated above, appear to be indeed optimal? Does this agree with some assumption about the solution found somewhere below (8.75)? (You should answer not simply 'yes' or 'no', but also name that assumption.)
- For some values of c, you will observe the behavior that you have already observed when obtaining the "shallow" solution; you will also observe its more complicated forms here. You should comment on *each type* of those different unexpected behaviors that you have found. You may attach plots if that helps to illustrate your description. Also, if you have taken course MATH 266, you may say how what you have observed in this problem jives with what you had learned in that course.

3. Repeat the assignment of item 2 above with a slightly **deeper initial guess**,

$$Y^{(0)} = \begin{cases} 1 - 20x, & 0 \le x \le 1 \\ -39 + 20x, & 1 \le x \le 2 \end{cases}$$

To save time, use only those values of c for which you obtained convergence for the "deep" initial guess. *In addition* to the questions in item 2, answer one more question:

- For what kind of c values (with respect to the behavior of the iterations) is the convergence most sensitive to how close the initial guess is to the true solution?

The moral of this exercise is simple: When solving a nonlinear BVP by iterations, be prepared that the behavior of your code may dramatically depend on the initial guess as well as on auxiliary parameters of your iteration scheme, including the discretization size h. *This observation extends beyond the particular iteration scheme considered in this Problem.*

Problem 9

Repeat Problem 7 using the Newton–Raphson method.
Answer all the questions posed in Problem 7. As for the logarithm of the error versus the iteration number, you should obtain a parabola pointing downward. Thus, the Newton–Raphson method converges (when it does so) *quadratically*. Is this type of convergence asymptotically[12] faster or slower than the linear convergence of Picard's methods?

Problem 10

Compare your results in Problems 7–9 and draw conclusions. Namely, based on your experience with the previous three problems, list strengths and weaknesses for each of the three methods. Then explain under which circumstances you would and would not use, each of these methods.
Note: For example, ease of coding is a strength, while inability to obtain some of the solutions is certainly a weakness.

Bonus-1

Use the Gerschgorin Circles Theorem to obtain the best estimate for the location of the eigenvalues of the following matrix:

$$A = \begin{pmatrix} 2 & \frac{3}{2} & 0 \\ \frac{i}{2} & 1 & 0 \\ 0 & -\frac{i}{2} & -2 \end{pmatrix} .$$

Bonus-2

(a) (**1 point**) Redo Bonus-1 problem of HW7 using the discretized BVP (8.4) with $h = 0.09$ (step size $h = 0.1$ will not "fit" into the interval [0 1.62]). Compare the result with that found in HW7. Which method, shooting or finite-difference discretization, is preferable for solving BVPs like this one?

(b) (**0.5 point**) Plot the error of your numerical solution. Explain the result.

(c) (**0.25 point**) Repeat part (a) with $h = 0.01$ and plot the error. Explain why it is greater than that for $h = 0.09$. *Hint*: Review Sec. 8.3.

Bonus-3

Equations (8.34) and (8.37) each lead to a second-order accurate method. Therefore, solutions obtained by those methods must differ by $O(h^3)$. Show *analytically* that this is indeed the case. Obviously, it is sufficient to show that (8.34) and "(8.37)$\cdot h$" [13] differ by $O(h^3)$, since the remainders of the methods use the same set of equations.
Hint 1: It is easier to begin by (8.37)$\cdot h$ and then, after dividing by a certain factor, show that it equals (8.34)$+O(h^3)$.
Hint 2: A useful formula to recall is $1/(1+\alpha) = 1 - \alpha + O(\alpha^2)$ for $\alpha \ll 1$.

Bonus-4

Find some vectors $\vec{\mathbf{w}}$ and $\vec{\mathbf{z}}$ and matrix A_{tridiag} as explained after (8.52).

[12] i.e., for many iterations
[13] We need to multiply by h to make the coefficient of Y_1 to be $O(1)$, as it is in (8.34).

HW # 9

Due: 04/04/22

Compare your plots in Problems 1 and 3 with those found under "Codes for examples and selected homework problems" on the course webpage.

Problem 1

Solve the BVP

$$y'' - \frac{2y}{(1+x)^2} = -\frac{4}{(1+x)^2}, \qquad y(0) = 0, \quad y(1) = 1$$

using the collocation method with $\phi_j = \sin(j\pi x)$ for $j = 1, \ldots, M$ and $M = 10$. Use the equidistant collocation points $x_k = x_0 + k \cdot h$. (See the *Technical note* below.) Compare your finite-element solution with the exact solution $y_{\text{exact}} = 2x/(1+x)$ by plotting them together, and also by plotting, in a separate figure, the error for $x \in [0, 1]$.

Hint: Note that the above BVP does not have zero boundary conditions, for which the theory in Lecture 9 is developed. One can transform this BVP into a one with zero boundary conditions by the following trick. Consider $z = y - x$. Clearly, if $y(0) = 0$, then $z(0) = 0$, and if $y(1) = 1$, then $z(1) = 0$. Thus, the new variable z has zero boundary conditions on $[0, 1]$, as required. Now, substituting $y = z + x$ into the original ODE, one obtains the ODE for z:

$$z'' - \frac{2z}{(1+x)^2} = \frac{2x - 4}{(1+x)^2}\,.$$

Thus, we have transformed the original BVP for y with nonzero boundary conditions into a BVP for z with zero boundary conditions. The methods of Lecture 9 can now be applied to this latter BVP to find $z(x)$; then $y = z + x$.

In Lecture 9 we did not mention the accuracy of the FEMs. Investigate this issue for the collocation method here. Namely, repeat the above calculations for $M = 20$ and $M = 40$ and discuss how the maximum error scales with $(1/M)$.

Note: Do ***not*** write your code as a loop over several values of M. Make it work for one value of M and then run it for different M. See the *Very important note* for Problem 8 of HW 8 as to why. *The same applies to Problem 3 below.*

A word of caution: As far as I know, the accuracy of the collocation method depends on the smoothness properties of the basis functions. Thus, your conclusion for $\phi_j(x) = \sin(j\pi x)$ may not necessarily apply to the case when ϕ_j's are the B-splines.

Technical note:

Note that M in Lecture 9 is related to the length of the vector `xplot = a:h:b` *differently* than N was related to that length in Lecture 8. Then follow the direction of *Technical notes* for Problem 5 of HW 8.

Problem 2

Obtain Eqs. (9.20) and (9.22) of the notes.

Hint: Integration should be done from 0 to h rather than from x_j to x_{j+1}. This will produce the same result, but with much less effort.

Problem 3

Solve the BVP in Problem 1 by the Galerkin method with the hat functions and $M = 10$. Compare your result with the exact solution.

As in Problem 1, investigate how the error of the Galerkin method scales with $(1/M)$.

Technical notes:

1. The matrix here, unlike in Problem 1, is tridiagonal; so you should set it up accordingly and then solve the linear system by an appropriate technique.

2. I recommend that you compute terms in (9.17) by a method similar to that of (9.21).

HW # 10

Due: / /05

Problem 1:

Let $\underline{\mathbf{w}}$ be any n-dimensional vector with $||\underline{\mathbf{w}}|| = 1$. Show that $Q = I - 2\underline{\mathbf{w}}\,\underline{\mathbf{w}}^T$ is: (a) symmetric and (b) orthogonal. [*Hint:* Use the corresponding definitions from Linear Algebra.]

Problem 2:

Let $\underline{\mathbf{v}} = [1, 12, 3, 4]^T$. Find the Householder matrices Q_1 and Q_2 that satisfy:

$$\text{(a)} \quad Q_1\underline{\mathbf{v}} = \begin{pmatrix} 1 \\ s_1 \\ 0 \\ 0 \end{pmatrix}, \qquad \text{(b)} \quad Q_2\underline{\mathbf{v}} = \begin{pmatrix} 1 \\ 12 \\ s_2 \\ 0 \end{pmatrix}.$$

Problem 3:

Use the Householder transformations to transform the matrix

$$\begin{pmatrix} 1 & 2 & 4 & 2 \\ 12 & 3 & -4 & 2 \\ 3 & 3 & 9 & -1 \\ 4 & -4 & -2 & 8 \end{pmatrix}$$

to the upper Hessenberg form. Also, exhibit the matrix Q that performs such a transformation, and verify by direct calculation that it is orthogonal (as proven in general on p. 9-8 of the notes).

Problem 4:

Use the (direct) power method and the inverse power method to find the eigenvalues of

$$\begin{pmatrix} 19 & -20 \\ -20 & 19 \end{pmatrix}$$

with the relative error less than 10^{-3}. (That is, iterate until $|\lambda^{(k+1)} - \lambda^{(k)}|/|\lambda^{(k)}| < 10^{-3}$.) How many iterations is required for each of the eigenvalues?

Problem 5:

Find the smaller eigenvalue of the matrix in Problem 4 using the direct power method and the orthogonalization technique described on pp. 9-12, 9-13. How many iterations are required in this case to achieve the same accuracy?

Problem 6:

Consider a 20×20 matrix

$$A = \begin{pmatrix} 2 & -1 & 0 & . & . & . & 0 \\ -1 & 2 & -1 & 0 & . & . & 0 \\ 0 & -1 & 2 & -1 & 0 & . & 0 \\ & & & . & . & . & \\ 0 & . & . & 0 & -1 & 2 & -1 \\ 0 & . & . & . & 0 & -1 & 2 \end{pmatrix}.$$

(You should set up the entries of this matrix in a loop.)

(a) What does the Gerschgorin Circles Theorem say about the location of the eigenvalues of A? (b) Use the direct and inverse power methods to find the largest and the smallest eigenvalues of A. How many iterations are required to achieve the relative accuracy of 10^{-3}?

Problem 7:

For the matrix in Problem 6, find the second largest eigenvalue (λ_2) using the orthogonalization method. How many iterations are required in this case to achieve the same accuracy?

HW # 11
Due: 04/06/22

Problem 1 (worth **0.5 point**)
Determine if the following system of two 1st-order PDEs leads to a hyperbolic, elliptic, or parabolic 2nd-order PDE:

$$\begin{cases} u_x - v_y = 0 \\ v_x + u_y = 0 \end{cases}$$

Hint: Differentiate one of the equations with respect to either x or y. Then use the other equation to eliminate a derivative of one of the functions, i.e., u or v, and obtain a second-order PDE for the other function.

Problem 2 (worth **0.5 point**)
(a) Determine the type of the PDE

$$xu_{xx} - u_{yy} + \frac{1}{2}u_x = 0\,.$$

(b) Find and then solve the equations for the real-valued characteristics of this PDE wherever they exist.

HW # 12
Due: 04/15/22

Problem 1 (worth **1.5 points**)
Use scheme (12.12) to solve IBVP (12.1)–(12.3) with $g_0(t) = g_1(t) \equiv 0$ and $u_0(x) = \sin \pi x$ up to $t = 0.5$. Use $\kappa = 0.004$ with (a) $h = 0.1$ and (b) $h = 0.05$.

- Answer this question *before you begin doing this problem*:[14]
  Do you think that the answer for part (b) will be more accurate than that for part (a)? Please briefly explain.
- Now, compare your results with the exact solution: $u_{\text{exact}} = \sin(\pi x)\, \exp(-\pi^2 t)$ (that is, plot the error at $t = 0.5$ as a function of x.) Which result, for part (a) or for part (b), is more accurate?
- As you inspect the evolution of your solution with time (see *Technical note 4* below), describe briefly what the solution does for $h = 0.1$ and for $h = 0.05$.

Technical notes:
1. The following reiterates the *Very important note* in Problem 8 of HW 8:
The assignment above asks you to obtain the solution for two values of h. However, when programming your code, ***do not attempt to make it work for more than one value of*** h ***at a time***! Make sure your code works for one h and one κ, and then change their values inside the code (or inline via the `input` command) as needed.
This is a particular instance of an important programming principle: *Never make several significant changes to your code at once.*[15] Otherwise, if you code happens not to work correctly, you will not know what caused it.

2. Define solution `U` over the *entire* interval `x = a :h: b`. At each time step, first set the boundary conditions for `U(1)` and `U(end)`. This will allow you to apply formula (12.12) to points near the boundaries in the same way as to points away from the boundaries.

3. You *may* define the solution at all time levels, `U(m,n)`, but you don't really need to do so. It will suffice if you keep only the solution at the current time level and that at the new level (which you may

[14] You will be graded *not* for the correctness of your answer but on its coherency.
[15] Or, in this case, do not program a new scheme trying to include all bells and whistles from the start.

call, say, `U_new`). The need for keeping both solutions should become clear when you examine the left- and right-hand sides of (12.12).

4.

> In your `for`-loop for time evolution, insert a plotting command after every time step followed by the `pause` command. This will allow you to inspect your solution and its error after every step and use this information ***in case that you need to debug your code***.
>
> When you submit your code, please comment out these plotting and pausing commands.
>
> **In all subsequent programming problems, you must have this *debugging tool* in your time-evolution loop.**

Problem 2

Consider the coupled system of ODEs (12.16), which approximates the IBVP (12.1)–(12.3). Use a technique of Sec. 5.4.1 of Lecture 5 to obtain the stability condition of the simple Euler method when it is applied to (12.16). Compare the stability criterion you will obtain with Eq. (12.29) in the notes. *Hint*: You will need the result of the Lemma in Sec. 12.3.

Is system (12.16) stiff or non-stiff? Please explain in detail.

Problem 3 (worth **1.5 points**)

A straightforward way to improve the order of approximation of the simple scheme (12.12) is to replace the forward difference in Eq. (12.9) with the central difference:

$$\frac{\partial u}{\partial t} \;\to\; \frac{U_m^{n+1} - U_m^{n-1}}{2\kappa} + O(\kappa^2)\,.$$

(The method resulting from this is often called the Richardson method.)

(a) Write the difference scheme corresponding to this method.

(b) Draw the stencil (see notes) for this scheme.

(c) Apply the von Neumann stability analysis to this scheme and show that it is unstable for any $r \neq 0$. Which of the unstable Fourier harmonics has the greatest growth rate?

Technical note:

> **In this and in all other problems involving the von Neumann analysis**, whenever you see a combination like $(U_{m+1} - 2U_m + U_{m-1})$, do **not** break it apart! It always gives rise to a factor $2r(\cos\beta h - 1) = -4r\,\sin^2(\beta h/2)$; see the identity before (12.38). Denote that factor as z. Thus, by keeping those terms together, you have combined two parameters, r and βh, into one, z. This will allow you to make conclusions of your von Neumann analyses valid not just for some selected, but for *all*, values of βh by identifying the "most dangerous" value, as we did after Eq. (12.39).[a]
>
> [a] With some effort, one can show (you do *not* need to do it) that the z is the counterpart of λh in Lecture 4.

Hint specific to part (c): Show that for any z, there is a ρ with $|\rho| > 1$. Refer to the directions and Hints in Problem 10 of HW 5, where you did a similar work. Also, since $z < 0$, it may be more convenient to work with $(-z)$, which you may call, say, w.

(d) What method for ODEs is the Richardson method similar to? *Hint*: Look at its stencil, focusing on how it goes along the axis of time (since evolution occurs along that axis, not along x). You may need to look closely through the Lecture where we introduced this type of methods.

(e) Explain how your answer to part (c) could have been obtained without any calculations, but instead based on the following information:

- The stability region of the method which you identified in part (d) (that region is displayed in Lecture 4);
- The values of the eigenvalues of the matrix in Eq. (12.16), which you used in Problem 2.

Let us recap on Problems 2 and 3. They show that using different discretizations of u_t, one can obtain different finite-difference schemes for the Heat equation, which are then

the counterparts of various methods that we studied in Part I (IVPs for ODEs) of this course. It also transpires that the stability condition of a given finite-difference scheme for the Heat equation is closely related to the stability condition of the corresponding method for ODEs. The Bonus problem at the end of this HW reiterates the same point.

Problem 4 (worth **1.5 points**)

Preamble:

The purpose of this problem is to explore the effect of smoothness of the initial condition and of the value of $r = \kappa/h^2$ on the order of approximation of the simple explicit method (12.12). The IBVP you will solve here is almost the same as in Problem 1 of this Homework, except the the initial condition will be different.

Download a code `hw12_p4.m` from the website (it is posted under the link "Codes for examples and selected homework problems"). This code calculates the solutions to the Heat equation for $h_1 = 0.05$, $h_2 = h_1/2$, and $h_3 = h_2/2$. The parameter $r = \kappa/h^2$ is specified by the user to be any value of: 0.5, 0.4, or 0.3; then the time step $\kappa = rh^2$ is calculated within the code.

The initial conditions used in the code are:

$$u_0(x) = \begin{cases} 0, & 0 \le x \le 1/4; \\ \left(\cos\left[2\pi(x-\frac{1}{2})\right]\right)^p, & 1/4 < x < 3/4; \\ 0, & 3/4 \le x \le 1; \end{cases} \qquad p = 0, 1, 2, \ldots . \tag{1}$$

For $p \ge 1$, this function has $p-1$ continuous derivatives while its pth derivative is discontinuous; for $p = 0$, the function is itself discontinuous at $x = 1/4$ and $x = 3/4$. The value of p is specified inside the code.

Your goal will be to compute the exponent $\gamma(x)$ that relates the error $\epsilon(x)$ of the numerical solution to the mesh size h:

$$\epsilon(x) = C\, h^{\gamma(x)}, \tag{2}$$

where C is some constant. Note that the exponent may depend on x. You will then be asked to use your results to draw certain conclusions.

Assignment:

(a) Assuming that the error satisfies Eq. (2), verify that

$$\gamma = \log_2 \frac{\epsilon[h] - \epsilon[h/2]}{\epsilon[h/2] - \epsilon[h/4]}\,. \tag{3}$$

Here we have denoted by $\epsilon[h]$, $\epsilon[h/2]$, $\epsilon[h/4]$ the errors computed using mesh sizes h, $h/2$, and $h/4$, respectively. (Since ϵ depends on x, the values of γ are computed only for the coarsest grid, associated with $h_1 \equiv h$.)

Now, this formula is not yet a practical tool for computing the exponent γ because we do not know what the error is (unless we know the exact solution, u). However, the exponent can still be found using (3) even if we do not know the exact solution. To see that, notice that the numerical solution $U[h]$ obtained with a step size h satisfies:

$$U[h] = u + \epsilon[h].$$

Then, Eq. (3) easily becomes:

$$\gamma = \log_2 \frac{U[h] - U[h/2]}{U[h/2] - U[h/4]}\,, \tag{4}$$

which *can* be used to compute γ based only on the available numerical solutions.

(b) For each $p = 0$, 1, 2, run the code for $r = 0.5$, 0.4, and 0.3.

Using the output of the code, make a table that shows γ as a function of p and r. In a case where γ has a nonzero imaginary part, it means that its real part does not carry any actual information, and

so you may simply say "N/A" (for "not available"). If your γ is real, but oscillates between two values along x, write both these values into your table. *Round your answers to two decimal places.*

From your table, conclude about the following:
(i) How the accuracy (i.e., γ) of your solution varies with p (i.e., the smoothness of the initial datum) for each of the three values of r; and
(ii) How the accuracy of your solution varies with r for each of $p = 0, 1, 2$.

(c) For what values of r and p does the final solution appear jagged to the naked eye (i.e., without your zooming into the curve)? Why does this jaggedness occur for these values of r and p, but not for the others? (See the end of Sec. 12.5.)

(d) Finally, answer the following question, which is the *main intended outcome* of this problem. Suppose that you need to solve the Heat (or a more complicated) equation with a non-smooth initial condition. What should be your choice of the time step κ (that is, $r = \kappa/h^2$) so as to obtain a reasonably accurate solution and yet not to make your code run unreasonably long?

Bonus (worth **2 points**)
An alternative method to improve the accuracy (in time) of scheme (12.12) is to follow the lines along which the modified Euler method was constructed. Namely, it first computes

$$\bar{U}_m = rU^n_{m+1} + (1-2r)U^n_m + rU^n_{m-1}, \quad 1 \le m \le M-1, \tag{1a}$$

and then computes

$$U^{n+1}_m = U^n_m + \frac{r}{2}\left[\left(U^n_{m+1} - 2U^n_m + U^n_{m-1}\right) + \left(\bar{U}_{m+1} - 2\bar{U}_m + \bar{U}_{m-1}\right)\right]. \tag{1b}$$

Note that for the boundary values $\bar{U}_{0,M}$, one should use the *known* boundary values $U_{0,M}$. The code that solves the IBVP stated in Problem 1 by the modified Euler scheme is posted under the link to "Codes for examples and selected homework problems."

(a) (**0.75 point**) Apply the von Neumann stability analysis and verify that the stability condition for this method is also $r \le 1/2$.
Hint: Carry out the calculations using the same notation z as in Problem 3(c) above. For example, you should verify that the error made in $\bar{U}$ satisfies:

$$\bar{\epsilon}_{m+1} = \rho^n e^{i\beta h(m+1)}\left(1 + r\left\{e^{i\beta h} - 2 + e^{-i\beta h}\right\}\right) \equiv \rho^n e^{i\beta h(m+1)}(1+z), \qquad \text{etc.}$$

FYI: Your expression for ρ in terms of z must jive with the corresponding factor obtained for the modified Euler method in Lecture 4.

Technical notes:

1. To satisfy the condition $|\rho| < 1$, you will obtain a double inequality for z. Remember that you need to consider each side of this inequality separately.

2. Each of your inequalities will be a *quadratic* inequality for z of the form: $(z-a)(z-b) > 0$ or < 0. If you do not remember how to solve those, you will need to review that (high school) material.

3. When considering one of the sides of the double inequality, you will need to decide for what value of βh this side is violated for the smallest possible r. Follow the analysis found after (12.39).

(b) (**0.25 point**) Now suppose that $r > 1/2$ (say, $r = 0.6$). Then, in what *qualitative* way will the time evolution of the most unstable Fourier harmonic for scheme (1) above be different from that of the most unstable harmonic for scheme (12.12)?
Hint: You can get inspiration by running the code and using the plotting-and-pausing "technique" described in Problem 1.

(c) (**0.25 point**) Note that, in complete analogy with the modified Euler method, the discretization error of the above method is $O(\kappa^2 + h^2)$. Explain whether you would want to use this method if you want your error to have the order $O(\kappa^2)$.

(d) (**0.25 point**) Confirm (or prove wrong) your answer to part (c) by using the code provided and comparing your results with those obtained in Problem 1(a) for $h = 0.1$ and $\kappa = 0.004$ (the stable case).

(e) (**0.5 point**) Now run the code for $h = 0.1$ and $\kappa = 0.004$ (the unstable case) and compare your results with those obtained in Problem 1(a). Explain why the errors in this part and in Problem 1(a) are so vastly different.
Hint: Look at ρ as a function of the parameter z defined in Problem 3 (for the "most dangerous" harmonic).

HW # 13
Due: 04/20/22

Problem 1 (worth **0.5 point**)
Write down the analog of Eq. (13.9) for the *inhomogeneous* Heat equation:

$$u_t = a\, u_{xx} + f(x,t),$$

where a =const and $f(x,t)$ is a given function.

Problem 2
Use the von Neumann stability analysis to obtain the stability criterion (i.e., Eq. (13.27) of the notes) for the θ-family of methods (13.17). Make sure to carry out the analysis for all values of βh, not just for $\beta h = 0$ and $\beta h = \pi$. For this, recall the *Technical note* for Problem 3(c) of HW 12. Also, you need to analyze condition(s) that follow from *both* parts of a certain double inequality.

Problem 3 (worth **0.5 point**)

(a) Show that the statements about eigenvectors and eigenvalues of certain matrices found immediately after Eq. (13.20) are true. (Review an undergraduate Linear Algebra textbook if needed.)
Note: Your proof must explicitly rely on the relation $A\vec{v} = \lambda\vec{v}$.

(b) Use these results to show that if λ_j is an eigenvalue of matrix A in Eq. (13.20), then the eigenvalue of the entire matrix on the r.h.s. of (13.20) is given by (13.21).
Note: Again, you must "act" with certain matrices on the eigenvector(s) of A.

Problem 4 (worth **1.5 points**)
The purpose of this exercise is two-fold. First, you will need to program the Crank–Nicolson scheme into a code. Second, you will explore the caveats associated with the notation $O(\kappa^2 + h^2)$ for the truncation error.

Task 1 (**1 point**):
Solve Problem 1 of HW 12 using the Crank–Nicolson scheme. Use $h = 0.1$ and (a) $\kappa = 0.1$, (b) $\kappa = 0.05$, (c) $\kappa = 0.025$, and (d) $\kappa = 0.01$.[16] Plot these four solutions, using different line styles and a legend, in the same figure with the exact solution. In a separate figure, plot all four errors.[17]

Technical notes:

1. Review the *Technical notes* for Problem 5 of HW 8, especially Notes 2 and 4, and follow a similar approach here. Note that this is *different* from the approach that was recommended for Problem 1 of HW 12, the reason being the same as that explained in the aforementioned Notes.

2. *As* in Problems 1 and 4 of HW 12, do **not** keep your solution at all time levels, but just at the last two.

If you send your final code to me by 7 p.m. one day after the day when this problem is assigned to be attempted, it will save you coding time for Problem 5; see the first paragraph of that Problem.

[16] Do **not** try to program your code to compute the solutions for all values of κ in a loop. Rather, obtain the results for one value of κ at a time. See the *Technical note* in Problem 1 of HW 12 and *Very important note* for Problem 8 of HW 8 for the general principle behind this.

[17] What you measure here is the global error. Recall from Section 12.2 that the global and discretizaion errors have the same dependence on κ and h.

Task 2:

The results that you will have obtained prompt a few questions (please read to the end of this problem before attempting to answer them):

(i) The magnitude of the error for $\kappa = 0.1$ does not jive with our intuition for a quantity that is $O(\kappa^2)$. Why is this error so large?

(ii) (**0.25 point**) Why does the error change sign when going from cases (a), (b) to case (c)?

(iii) (**0.25 point**) Why does the error increase when going from case (c) to case (d)?

Below I will answer question (i), while you will need to answer the other two questions.

First, note that the notation $E = O(\kappa^2)$ means that the error $E = C \cdot \kappa^2 +$ 'terms which decrease faster than κ^2 as $\kappa \to 0$'. The constant C, however, can be arbitrary. E.g., it can be 0.1, or it can be 100. Obviously, the errors produced in these two cases will differ by a factor of 10^3. Next, for the Crank–Nicolson scheme, the leading terms of the discretization error are

$$E_{\text{CN}} = -\frac{1}{12}(h^2 u_{xxxx} + \kappa^2 u_{xxxxxx})$$

(this can be shown by extending your work for problem Bonus-1 of this HW). Note that h^2 is multiplied by the 4th derivative of u, while κ^2 is multiplied by the *6th* derivative. For the IBVP in question, the exact solution was listed in Problem 1 of HW 12; it is $u_{\text{exact}} = \sin(\pi x)\exp(-\pi^2 t)$. Therefore, $u_{\text{exact},\,6x} = -\pi^6 \cdot u_{\text{exact}}$. Substituting this into the formula for E_{CN} above, we see that for $\kappa = 0.1$, the part of the error due to κ^2 has the size of about 80% of the exact solution!

Side note: The above analysis should not be (mis)interpreted to the effect that the discrepancy between the exact and numerical solutions must be on the order of 80 %. A more sophisticated analysis[18] confirms the discrepancy of ~ 35 %, which you should have found in this problem for $\kappa = 0.1$.

Problem 5 (worth **2.5 points + 0.25 bonus point**)

Download the code `hw13_p5_skeleton.m` from the page "Codes for examples and selected homework problems". In the indicated place inside this code, insert your own sub-code that can implement a method from the θ-family of methods (13.16) or (13.17). The value of θ should be specified at the beginning of the code. The initial and boundary conditions are already set inside the code.

If you send me your final code for Task 1 in Problem 4 by the deadline indicated there, I will email you my code for Problem 5, and you will not need to code anything, just do the assigned calculations and answer questions.

(**0.75 point**) Redo Problem 4(b) of HW 12 for the Crank–Nicolson ($\theta = 1/2$) and backward-difference ($\theta = 1$) methods. Run simulations for $p = 0,\ 1,\ 2,\ 3$. Since there is no stability threshold value of r for these methods, instead of $r = 0.5,\ 0.4,\ 0.3$, consider instead values $\kappa = h/2$ (i.e., $r = 1/(2h)$) and $\kappa = h/4$ (i.e., $r = 1/(4h)$). Summarize your results in a table as described in Problem 4(b) of HW 12.

Also, for each case, *in a separate table* record the maximum relative errors (for all three h values), i.e. $\max_x |U_{\text{numeric}} - U_{\text{exact}}| / \max_x |U_{\text{exact}}|$, of the solution U_{numeric}. This error for all three values of h is computed and displayed at the end of the code using a very close approximation to U_{exact}.

Answer the following questions about your results:

(q1) (**0.25 point**) For each value of θ, list those values of r, h, and p where the solution appears to be jagged (to the naked eye).

(q2) (**0.75 point**)

Preamble:

Recall that the Crank–Nicolson method, unlike the simple explicit method, is unconditionally

[18] based on seeking $u_t = u_{xx} \cdot (1 + \epsilon)$, where ϵ is a small factor due to E_{CN}

stable. Then it may seem strange that the same jagged harmonic that dominated the unstable solution of the simple explicit method (e.g., in Problem 1 of HW 12) also dominates the Crank–Nicolson solution in some cases considered in the present problem.

Below you will be asked to explain *how* that harmonic can dominate a stable CN solution. You will also see why this harmonic does not appear in the solution obtained by the backward Euler method. For the sake of this explanation, let us focus on the discontinuous initial condition ($p = 0$), in which the amount of the harmonic with $\beta h = \pi$ is the largest among all p.[19] We can also restrict our attention to one value of h, say, $h = 0.05$.

Assignment:
Use the code posted in "Codes for examples and selected homework problems" on the course website to plot, in the same figure, the amplification factors ρ for the methods with $\theta = 1$ and $\theta = 1/2$ for each of the two values of r versus (βh). In addition, plot the amplification factor corresponding to the exact solution:

$$\rho_{\text{exact}} = \exp\big[-r(\beta h)^2\big]$$

(if you have taken a course on partial differential equations, this formula should make sense; otherwise, just take it on faith). Thus, you should have a total of six curves in your figure, three for each value of r; use different line styles and thicknesses for them.

Now use the above figure to explain why you have seen the fastest harmonic in some plots but not in others (for $h = 0.05$). (Recall the meaning of the amplification factor; specifically, see the line before Eq. (12.34) in Lecture 12.)

FYI: As you have seen in this problem, the backward Euler method smoothens a discontinuous initial condition much more efficiently than the Crank–Nicolson method. This fact is the foundation for a technique called *Rannacher smoothing*, used to handle a non-smooth initial condition in Heat Equation-type problems.

(q3) (**0.25 point**) Now look at both tables (one for γ and one for the relative errors) that you recorded for this problem. Do values of γ and the relative error "tell the same story" about the accuracy of the solution? (The answer here may possibly depend on the θ, r, and p values.) Whose story (if they are different) would you believe, and why?[20]

Do not try to guess a "correct" answer. I will give credit mostly for your logic and consistency rather than for that presumed "correctness".

(q4) (**0.5 point**) Suppose you are to solve a parabolic PDE whose initial condition is either discontinuous ($p = 0$) or has a discontinuous first derivative ($p = 1$). *Based on your results*, explain what considerations could affect your choice of the numerical parameters: θ, r, and h. Be specific in your explanations *for each parameter*. For example, you may say "Under circumstances A, I'll use $\theta = 1$ (or 0), while under circumstances B I'll use $\theta = 1/2$" (it is possible that you will never use one of those values).
Remember that there is no "universally best" numerical scheme. Schemes are selected by their performance in specific circumstances. So, you should discuss when one scheme is preferred over the other.

Note: This question, where you don't need to do any new work, is worth half a point. This is because to answer it, you need to think how to summarize what you have learned from Problems 4 and 5. Therefore, I will grade not only correctness of your answers but also their clarity and coherence.

[19] This fact is shown in a course on Fourier analysis.
[20] This should depend on your goal, and so you should state it first.

(Bonus) (**0.25 point**) Why is $\gamma \approx 1$ for the backward-difference method, even though we stated in Lecture 13 that the discretization error for this method is $O(\kappa + h^2)$? (That is, naively interpreting the exponent of h, one could conclude that $\gamma = 2$.)

Bonus-1 (**1.5 points**)

Obtain Eqs. (13.13) and (13.14) from Eqs. (13.11) and (13.12) of the notes.

Note: Doing calculations formally and using the last terms ($O(k^4 + \ldots)$) in (13.11) and (13.12), one may naively conclude that the error in (13.14) should contain terms $O(\kappa^4/h^2 + \kappa^3/h + \kappa h)$. You need to explain that such terms will *not* appear.

To that end, write what these terms are explicitly, relating them to the higher-order derivatives of u. Show which of them cancel out. (When marking mutually cancelling terms in your work, please use different cancellation symbols, such as single, double, etc. slashes or front and back slashes.)

Bonus-2

Use the von Neumann analysis to show that the DuFort–Frankel method is unconditionally stable.

Note: As before, your conclusion must be valid for all values of βh, not just for selected few. Here, however, the notation z introduced in Problem 3(c) of HW 12 will not appear. Alternatively, you can plot the absolute value of ρ as a function of r and βh using either Matlab's command `mesh` or Mathematica's command `Plot3D`. An equivalent way is to make one 2D plot displaying curves $|\rho|$ as a function of βh only for several representative values of r.

Bonus-3 (worth **0.5 point**)

This problem is intended to reinforce your mastery of the way to estimate the exponent in the error dependence on the discretization step, as presented in Problem 4 of HW 12.

Run your code for Problem 4 of *this* HW to find the errors of the numerical solutions at $t = 0.5$ and $x = 0.5$ for cases (b), (c), (d) in Task 1 of that Problem. You assignment is to show that for a fixed h, these errors actually scale as $O(\kappa^2)$, as the CN theory predicts.[21]

Note that one can write the error as

$$E_j = E(\kappa_j) + E(h), \qquad j = \text{(b), (c), (d)} \quad \text{as per Problem 4}$$

where E_j is the total error and $E(\kappa)$ and $E(h)$ are the errors due to the $O(\kappa^2)$- and $O(h^2)$-terms. Note that h is the same in all three cases (b), (c), (d) above, and thus you can eliminate $E(h)$ and establish that $E(\kappa)$ is indeed approximately $O(\kappa^2)$ by using the formula of Problem 4(a) of HW 12.

HW # 14

Due: 04/27/22

Before you begin coding for this and especially for the next HW, please review my suggestion about staying close to the lectures' notations, made in HWs 1 and 8.

An additional point may also be worth mentioning. Sometimes you may be tempted to simplify some of the formulas derived in the lecture notes before programming those formulas into your code. RESIST THAT TEMPTATION! The main reason is the same as before: If you want me to check your code for mistakes, I can only check the notations that I used and hence readily understand. Different notations will reduce both my willingness and ability to help you. (A secondary reason is that Matlab is designed to do calculations better than you; so leave to experts their job.)

Compare your plots in Problems 1 and 2 with those found under "Codes for examples and selected homework problems" on the course webpage.

[21] We skip case (a) because, as we showed in Problem 4, the error there is too large and hence may have a noticeable deviation from the $O(\kappa^2)$ dependence.

Problem 1

Use the method described in the notes to solve the IBVP

$$\begin{aligned} &u_t = u_{xx}, \qquad x \in (0,1), \quad t > 0 \\ &u_x(0,t) + u(0,t) = \pi, \qquad u(1,t) = 0, \quad t > 0 \\ &u(x,0) = \sin(\pi x), \quad x \in (0,1). \end{aligned}$$

(Note that the initial condition matches with both boundary conditions.) Use the Crank–Nicolson scheme with $h = 0.02$ and $\kappa = h/2$. Display the solution at $t = 2$.

Verify that your solution satisfies the boundary condition at $x = 0$ [22] by using formula (8.35) from the notes (see Technical note 2 below).

Technical notes:

1. Program your code using general notations p and q, as in (14.3), (14.11), and (14.12). (Use the fact that they are time-independent.)

2. According to (8.35), u_0' can be computed with the second-order accuracy using the forward difference quotient, $(U_1 - U_0)/h$, and the second derivative. As an approximation for the second derivative term, use

$$\delta_x^2 U_1 = h^2 \, (u_{xx})_1 + O(h^4) = h^2 \, (u_{xx})_0 + O(h^3) \, .$$

Bonus (i) (**0.75 point**) This problem provides an example of how a boundary condition can cause the Heat equation to have an unstable *analytical* solution. To see this, set $p < 1$ in the above IBVP (in the original one, $p = 1$) and examine the numerical solution at several values of $t_{\max}$. Now repeat for $p = 2$. You should see the instability.

Show that for $p > 1$, the homogeneous version (i.e., that with $q = 0$) of the above IBVP has an unstable analytical solution. To that end, first seek a solution in the form

$$u(x,t) = (A \, e^{kx} + B \, e^{-kx}) \, e^{\lambda t},$$

where A and B are to be determined from the boundary conditions. Next, show that for $p > 1$, one can have $k \in \mathbb{R}$. Finally, explain why this means that this solution is unstable.

Bonus (ii) (**0.5 point**) Show how a nonhomogeneous IBVP (i.e. that with $q \neq 0$) can be transformed into a homogeneous one when $p \neq 1$.[23] (That is, both the boundary conditions and the PDE must become homogeneous.) Incidentally, your transformation should explain your numerical results for $p < 1$.

Hint: See the Hint for Problem 1 of HW 9.

Problem 2 (worth **1.5 points**)

Solve the IBVP with variable coefficients:

$$\begin{aligned} &u_t = \frac{\partial}{\partial x} \left((1+x) u_x \right) - \frac{x^2}{1+t^2} u, \qquad x \in (0,1), \quad t > 0 \\ &u(0,t) = 0, \qquad u(1,t) = 0, \quad t > 0 \\ &u(x,0) = \sin(\pi x), \quad x \in (0,1). \end{aligned}$$

Use the Crank–Nicolson scheme with $h = 0.02$ and $\kappa = h/8$. Display the solution at $t = 0.5$.

Technical notes:

1. Your scheme will have the form (14.10), but, of course, will have different matrices A and B and vector $\vec{\mathbf{b}}$ than there.

2. You first need to decide whether you will code your scheme as (14.17) or as (14.19). There is no correct or wrong answer here; it depends on which of the schemes you find *easier to code*. If you decide

[22] The boundary condition at $x = 1$ should be apparent from your plot.

[23] The case $p = 1$ is special and is somewhat harder, so you do not need to consider it.

to use (14.17), then, in your code, use notations like `a_pde` for a in (14.17), etc., because otherwise you will confuse these coefficients with the three diagonals `a`, `b`, `c` of the counterparts of matrices A and B.

3. Another decision you want to make is whether you will work with vector `U` defined on the full interval `x` (i.e., the one which includes end points) or that defined on the `x`-interval with the end points being chopped off. These two different approaches are illustrated in the posted codes for HW 12 and 13. Again, each of them is correct. However, once you have chosen one of these approaches, stay with it (within this problem) and do *not* switch back and forth, as this will only confuse you.

4. *Very carefully* work out the entries on each diagonal of your matrices, especially those terms that come either from a_m's (if you use (14.17)) or $\alpha_{m\pm1/2}$, if you use (14.19). If you misalign indices in some way (even by one), this may drastically change the solution. Indeed, let us use (14.17) to illustrate this. While having a_m in the place of a_{m+1}, or vice versa, leads only to the error $a_m - a_{m+1} = O(h)$, that error gets multiplied by $1/h^2$: see the second line in (14.17). The result is a large — $O(1/h)$ — change in the matrix entries.

As always, begin with sketching the grid and labeling the coefficients which appear in the diagonals of your matrices. Keep in mind that, also as always, the indices in (14.17) or (14.19) are off by 1 from those in a Matlab code.

5. Write out your scheme in the matrix form (14.10), explicitly showing the first couple terms on each diagonal.

When coding matrix B, you may use command `diag` instead of `spdiags` if you find it *easier to work with*;[24] both of them were introduced in Problem 4 of HW 8. If you decide to use `spdiags`, review its syntax (found in that Problem) and, in particular, which is the first entry of the vector that makes it into the matrix.

Once you have set up B, you can compute the r.h.s. vector as shown in the posted code for HW 13.

6. Follow the rule stated in Technical note 4 for Problem 1 of HW 12.

7. If you find that your code does not work properly, you may introduce auxiliary factors like `debug1`, `debug2`, etc., into your coefficients, so that setting them both to 0 would recover the plain Heat equation. For example, if you use (14.19), you would define $\alpha = 1 +$ `debug1 x`, $\beta =$ `debug2*-x.^2/(1+t.^2)`. You can then turn them on and off individually to trace likely locations of errors in your code.

Problem 3 (worth **2 points**)

Preamble: If you did what was asked after Eq. (14.26) and in the paragraph before Eq. (14.34), you can now breath out and pad yourself on the back, as the following assignment will only ask you to write down what you have already done. If, on the other hand, you had decided to charge ahead through the text without stopping and doing calculations where it was requested to do so (which, ahem, has happened with some students), now comes the moment of truth. You will need to go back and do those calculations.

The assignment below is not that hard, and on its face value would be worth 1 point. The reasons that it is valued at 2 points are: (i) These calculations check your *understanding* of how second-order accurate schemes are constructed, and (ii) With a greater weight, I will be able to grade on an effectively a finer scale and subtract points for sloppy derivations.

Assignment:

(a) Read the text below Eq. (14.26) and present a careful[25] explanation of each term in that equation and its accuracy. Include a sketch of the stencil in your derivation. Conclude with what the overall accuracy of scheme (14.26) is.

(b) Read the text before Eq. (14.34) and present a careful[26] explanation of each term in that equation and its accuracy. Include a sketch of the stencil in your derivation. Conclude with what the overall accuracy of scheme (14.34) is.

[24] Do not worry about the speed of your code. Your first concern is that it must work correctly, and that the reader could understand its logic. Everything else, e.g., speed of execution, would be a nice, but not necessary, feature.

[25] Note that the issue requiring "more accurate work," as pointed out in the footnote after Eq. (14.19), is relegated to a Bonus problem below.

[26] Of course, you can and should refer to your results in part (a). However, a sloppy/hand-waving/corner-cutting work on your part will entail logical consequences in regards to your grade.

Problem 4 (worth **1.5 points**)
Solve the nonlinear IBVP:

$$
\begin{aligned}
&u_t = (u-1)u_x + \gamma u_{xx}, \qquad x \in (-5,5), \quad 0 \le t \le 3 \\
&u(-5,t) = 0, \qquad u(5,t) = 2, \quad 0 \le t \le 3 \\
&u(x,0) = \frac{4}{\pi} \arctan\left(\exp[7x]\right), \quad x \in (-5,5)
\end{aligned}
$$

for $\gamma = 0.125$, 0.25, and 0.5 (i.e., you need to solve three IBVPs that differ from one another by the value of γ). Compare your result with the asymptotically exact[27] solution $u_{\text{as}-\text{ex}} = 1 + \tanh[x/(2\gamma)]$ by plotting *both* the solution and the error.

Now, regarding the numerical method: You are free to choose whichever *non-explicit* method with accuracy $O(\kappa^2 + h^2)$ (that is, any method from Secs. 14.4.2 or 14.5) that you feel most comfortable with. So, to make an informed decision, review both of these sections.
Also, you will need to decide which value of h will allow you to resolve the behavior of both the initial condition and final solution (this should be clear from their plots). Having decided on the h, choose a reasonable value for r based on your work for HW 13.

Technical notes:

1. See Technical note 2 for Problem 2 above.

2. Your errors should have the shape of the function $\operatorname{sech}(ax)\tanh(ax)$, where constant a changes with γ.

FYI: The above PDE is the celebrated Burgers equation, which arises in many applications in viscous fluid or gas dynamics.

Bonus-1 (worth **1.5 points**)
In this problem, you will be asked to generalize a certain scheme, derived for the simple Heat equation, to a linear parabolic PDE with variable coefficients. First, the derivation for the simple Heat equation will be presented to you. Then, you will be asked to do your part. No coding will be needed.

Derivation of a new method:
Note that the Heat equation can be written in the form:

$$
\underline{\text{at time level } n+1:} \qquad \left(U^{n+1}\right)_{xx} = \left(U^{n+1}\right)_t . \tag{1}
$$

In words, the second x-derivative at any node on level $n+1$ equals the first t-derivative on this level. We know how to approximate the l.h.s. of the above equation with accuracy $O(h^2)$: simply replace the continuous operator $\partial^2/\partial x^2$ with the discrete one δ_x^2/h^2. Thus, to obtain a scheme with the error $O(\kappa^2 + h^2)$, it remains to approximate the r.h.s. of the above equation with accuracy $O(\kappa^2)$. This can be done as follows. Note that for any continuously-differentiable $f(t)$, one has:

$$
\begin{aligned}
f(t+\kappa) \quad &= \quad f\left(t+\frac{\kappa}{2}\right) + \frac{\kappa}{2} f_t\left(t+\frac{\kappa}{2}\right) + O(\kappa^2) \;=\; f\left(t+\frac{\kappa}{2}\right) + \frac{\kappa}{2}\left(f_t(t) + O(\kappa)\right) + O(\kappa^2) \\
&\overset{(14.15b)}{=} \quad f\left(t+\frac{\kappa}{2}\right) + \frac{1}{2}\left(f\left(t+\frac{\kappa}{2}\right) - f\left(t-\frac{\kappa}{2}\right)\right) + O(\kappa^2)
\end{aligned}
$$

(please verify). Next, if $f(t) = u_t(t)$, then the above equation implies that

$$
\begin{aligned}
u_t(t+\kappa) \quad &= \quad u_t\left(t+\frac{\kappa}{2}\right) + \frac{1}{2}\left(u_t\left(t+\frac{\kappa}{2}\right) - u_t\left(t-\frac{\kappa}{2}\right)\right) + O(\kappa^2) \\
&= \quad \frac{3}{2} u_t\left(t+\frac{\kappa}{2}\right) - \frac{1}{2} u_t\left(t-\frac{\kappa}{2}\right) + O(\kappa^2) \\
&\overset{(14.15a)}{=} \quad \frac{3}{2}\left[\frac{u(t+\kappa)-u(t)}{\kappa} + O(\kappa^2)\right] - \frac{1}{2}\left[\frac{u(t)-u(t-\kappa)}{\kappa} + O(\kappa^2)\right] + O(\kappa^2)
\end{aligned}
$$

[27] This solution becomes exact when $t \to \infty$ and the x-interval is the entire line.

whence scheme (1) becomes

$$\frac{U_{m+1}^{n+1} - 2U_m^{n+1} + U_{m-1}^{n+1}}{h^2} = \frac{3}{2}\frac{U_m^{n+1} - U_m^n}{\kappa} - \frac{1}{2}\frac{U_m^n - U_m^{n-1}}{\kappa}, \tag{2}$$

with its error being $O(\kappa^2 + h^2)$.
Side note: Notice the similarity of the right-hand side of this equation with the multi-step method (3.5) of Lecture 3.

Your assignment:

(a) (**1 point**) Perform the von Neumann analysis of scheme (2) and verify that it is stable for any r (i.e., unconditionally stable).
Hint: You should be able to do so by plotting a quantity related to the amplification factor ρ versus the usual parameter z, which you have used in previous von Neumann analyses. Make sure to use an interval of z values that will convince both you and me that the method under study is indeed *unconditionally* stable.

(b) (**0.25 point**) Use the plot from part (a) to predict which method, this or the Crank–Nicolson, would better smoothen a discontinuous initial condition (see Problem 5 in HW 13).

(c) (**0.25 point**) Propose a generalization of scheme (2) for the PDE (14.18) of the notes. Of course, your generalization must have the same accuracy as the original scheme (2).
Hint: This is quite easy. Eqs. (14.19) of the notes should be helpful.

Bonus-2 (worth **0.5 point**)
Perform the "more accurate work" described in the footnote found a few lines below Eq. (14.19). (Of course, you should begin by stating which expression you are analyzing.)

Bonus-3
First, write down the scheme that results when formulae (14.34) are substituted into PDE (14.25). Then, derive Eq. (14.37) of the notes. Guidelines for this derivation are in the text following that equation. Once you get the idea, the details of the derivation will no longer look as gory as they may appear at first.

Bonus-4 (worth **2 points**)
Analyze numerical stability of the semi-implicit method (14.51) on the background of the constant solution $u \equiv C$. Follow these steps. Substitute into (14.51a) an ansatz:

$$U_m^n = C + \epsilon\, \rho^n\, e^{i\beta m h}, \qquad \epsilon \ll 1, \tag{3}$$

where the second term represents a small numerical error to the exact solution $u = C$ and ρ is our usual notation for the amplification factor in the von Neumann stability analysis. Linearize the expression inside the large parentheses in (14.51a). This requires four substeps. (i) Show that

$$U_m^{n+\frac{1}{2}} = C \; + \; \epsilon \left(\frac{3}{2}\rho - \frac{1}{2}\right) \rho^{n-1} e^{i\beta m h}. \tag{4}$$

(ii) Find the linearized version of $\left(U_m^{n+\frac{1}{2}}\right)^2$ following the idea of (14.39). (iii) Find the expression for $(U_m^n + U_m^{n+\frac{1}{2}})/2$ (this is very simple). (iv) Simplify the required term in (14.51a) using the idea explained after (14.40).

Now that you have a finite-difference scheme for a linear equation for the $O(\epsilon)$-term, you can apply to it the standard von Neumann analysis. At the end, you should answer this question: *Is scheme (14.51) unconditionally stable on the background of the constant solution*?

HW # 15
Due: 05/06/22

Compare your plots in Problems 1 and 2 with those found under "Codes for examples and selected homework problems" on the course webpage; see also a note on Problems 8 and 9 there.

Problem 1 (worth **1.5 points**)
Solve the IBVP

$$u_t = u_{xx} + u_{yy}, \qquad x \in (0,1), \quad y \in (0, y_{\max}), \qquad t > 0, \quad y_{\max} = 2.5$$
$$u(0,y,t) = 0, \quad u(1,y,t) = 0, \quad u(x,0,t) = 0, \quad u(x,y_{\max},t) = 0, \qquad t > 0$$
$$u(x,y,0) = \sin(\pi x)\, \sin(\pi y/y_{\max}), \qquad x \in (0,1), \quad y \in (0, y_{\max}).$$

Use the simple explicit method with $h = 0.1$ and $\kappa = 0.002$ to obtain the solution at $t = 0.5$. Plot it using one of Matlab's 3-dimensional plotting commands: `surf, mesh` or `waterfall`. Compare your result with the exact solution $u_{\text{exact}} = \sin(\pi x)\, \sin(\pi y/y_{\max})\, \exp\big(-(1+y_{\max}^{-2})\pi^2\, t\big)$ by plotting the error at $t = t_{\max}$.

Technical notes for Problems 1 and 2:

1. Right after defining the `x`- and `y`-vectors at the beginning of your code, use the `meshgrid` command (see Sec. 15.1) to define matrices `X` and `Y`.

2. Define the initial condition in terms of these matrices: `U0 = sin(pi*X) ? sin(pi*Y/ymax)`. Here the `?` is either `*` or `.*` ; you need to decide which one it is (read the description of each of these command in the Matlab Primer if you are not sure).

3. Define the solution `U(ell,m)` on the entire grid, i.e., on that that *includes* the boundaries. See *Technical note* 2 for Problem 1 of HW 12.

4. I also recommend that you *not* keep track of your solution at all time levels, but only at the nth and $(n+1)$st levels for any given n. See *Technical note* 3 for Problem 1 of HW 12 and also the example of updating the solution within the loops over m and l found in Sec. 15.1.

5. Make sure to label the axes in your plot.

Problem 2
Repeat the assignment of Problem 1 for the following BVP with *nonzero* boundary conditions:

$$u_t = u_{xx} + u_{yy}, \qquad x \in (0,1), \quad y \in (0, y_{\max}), \qquad t > 0, \quad y_{\max} = 2.5$$

$$u(0,y,t) = \sin(2\pi y/y_{\max})\, \exp\big(-(2\pi/y_{\max})^2 t\big),$$
$$u(1,y,t) = \cos(2\pi y/y_{\max})\, \exp\big(-(2\pi/y_{\max})^2 t\big),$$
$$u(x,0,t) = x\, \exp\big(-(2\pi/y_{\max})^2 t\big),$$
$$u(x,y_{\max},t) = x\, \exp\big(-(2\pi/y_{\max})^2 t\big);$$

$$u(x,y,0) = 10\, \sin(\pi x)\, \sin(\pi y/y_{\max}) + \sin(2\pi y/y_{\max})\,(1-x) + \cos(2\pi y/y_{\max})\, x\,.$$

The corresponding exact solution is

$$u_{\text{exact}} = 10\, \sin(\pi x)\, \sin(\pi y/y_{\max})\, e^{-(1+1/y_{\max}^2)\pi^2 t} + \Big(\sin(2\pi y/y_{\max})\,(1-x) + \cos(2\pi y/y_{\max})\, x\Big)\, e^{-(2\pi/y_{\max})^2 t}\,.$$

Technical note:
Since the boundary conditions are time-dependent, you need to set them up inside the loop over time steps, one level per step.

Problem 3 (worth **0.5 point**):
Obtain Eqs. (15.12) and (15.13). All this should take is a direct application of operators δ_x^2 and δ_y^2 (see Eqs. (15.9)) to the harmonic (15.11). You will also need to use the half-angle formula $1 - \cos\alpha = 2\sin^2(\alpha/2)$.

Problem 4 (worth **0.5 point**)
Write down the generalization of Eq. (15.24) for the inhomogeneous Heat equation,

$$u_t = a(u_{xx} + u_{yy}) + f(x, y, t).$$

Assume zero boundary conditions (i.e., ignore them) for simplicity.

Problem 5
(a) Following the suggestion in the text, verify that the Douglas method (15.48) is equivalent to scheme (15.33).
(b) Using the result of Problem 3, verify both lines in (15.49). In particular, make sure to demonstrate the second equation in the second line.

Problem 6
The purpose of this and the next problems is to prepare you for programming the Peaceman–Rachford method in Problems 8 and 9, respectively.

Write down equations (15.37) and (15.39) of the Peaceman–Rachford method *column by column* for $M = 4$ and $L = 5$, assuming *zero* boundary conditions at the sides of the rectangle in the (x, y)-plane. Also write down the counterparts of (15.37) for $l = 1$ and $l = L - 1$, which can be deduced from (15.62).

Technical notes for Problems 6 and 7:

1. Write out both matrices and vectors *explicitly*, i.e. entry by entry.[28]
2. Begin by drawing the grid and visualizing which vectors you set up and solve for at each step.
3. Set up the boundary values[29] for U and $\overset{*}{U}$ first. Then you will *not* need to treat the rows and columns that are next to the boundaries separately from the inner rows and columns. (But, of course, the *explicit* form of the equations for those next-to-the-boundary rows and columns will be different than the equations for the points inside the bulk of the grid.)

Problem 7
Write down the steps of the algorithm (found in Section 15.6) of the Peaceman–Rachford method with general (i.e., time-dependent) Dirichlet boundary conditions column by column for the case $M = 4$ and $L = 5$.
Strong suggestion 1: Read the first couple of pages of Section 15.6, where that algorithm is motivated.
Strong suggestion 2: Provide as much details as reasonably needed. In Problem 9 below, you will need to put those steps into a code, so providing sufficient details here will pay off when you get to Problem 9.

Problem 8 (worth **1.5 points**)
Solve Problem 1 of this HW using the Peaceman–Rachford method.
Based on your previous experience with the Crank–Nicolson method in HW 13, choose a value for κ and justify your choice in writing. (Yes, I will grade this part.)

Technical notes for Problems 8 and 9:

1. Begin by re-reading the *Technical notes* for Problems 1 and 2. All of them also apply here.
2. In addition, the auxiliary variable `Ustar` should also be defined on the entire grid (i.e., including the boundaries). Then you will be able to program Eq. (15.66) as one command, without considering the next-to-the-boundary rows and columns separately. Of course, do not forget about the vector $\vec{\overset{*}{\mathbf{b}}}_{;\,l}$. A similar comment also pertains to Eq. (15.67).

Problem 9 (worth **1.5 points**)
Solve Problem 2 of this HW using the Peaceman–Rachford method. (Take the same value for κ as in Problem 8.)

Additional Technical note for Problem 9:

[28] You do *not* need to write the same vectors and matrices explicitly more than one time. For example, once you have written out a vector for the first time, you can then refer to it in one of the abbreviated forms (15.20), (15.36), or (15.38).
[29] In Problem 6 they are all zero.

Modern versions of Matlab have a treacherously "convenient" feature whereby one can add a $(1 \times n)$ row to a $(m \times 1)$ column and obtain a $(m \times n)$ matrix. This feature can derail your code if, e.g., in (15.66), the first four terms on the r.h.s. are rows but the fifth one is set up as a column. (A similar thing can happen with (15.67).) Therefore, I recommend that you keep in mind Matlab's conventions that these commands define a row:

```
a(ell, 1:M) = ...
```

or

```
for m = 1:M;  a(m) = ...;  end;
```

and these commands define a column:

```
a(1:L, m) = ...
```

or

```
a(1) = ...;   a(L) = ... .
```

When in doubt, check the dimensions of your arrays with the command `size`.

Problem 10 (worth **0.5 point + 1.5 bonus points**):

(a) (**0.5 point**) Obtain boundary conditions for the two-dimensional Douglas and D'yakonov methods (i.e., obtain Eqs. (15.74) and (15.75), respectively).

(b) (**1.5 bonus points**) Obtain boundary conditions for the three-dimensional Douglas method (15.50).

Notes for parts (a) and (b):

1. To begin, draw the grid. (In (a), you may use the same picture for both methods.)
2. Referring to your picture, explain:

(i) What boundary values are needed for each of the equations (starting with the first equation), and

(ii) How one obtains those values.

For part (b), it may help to use a color pencil/marker to label the required faces (or edges) of the grid.

Problem 11:

Verify that the forms (15.81) and (15.83) of the Craig–Sneyd scheme are indeed equivalent.

Bonus-1 (worth **0.5 point**)

(a) Obtain Eq. (15.51) from (15.50). (Use *Mathematica* to perform algebraic manipulations.)

(b) Show that the expression in (15.51) is no less than -1.

Hint: It suffices to show (you must explain why) that for $X, Y, Z > 0$, $(X + Y + Z)/(\,(1 + X)(1 + Y)(1 + Z)\,) < 1$.

Bonus-2

Write down the generalization of the Peaceman–Rachford method (15.34) for the inhomogeneous Heat equation,

$$u_t = u_{xx} + u_{yy} + f(x, y, t).$$

Assume zero boundary conditions (i.e., ignore them) for simplicity.

Suggestion 1: It is more convenient to work with form (15.34) for the individual components (i.e., $U^n_{m,l}$) of the solution than with the vector form (15.62), (15.63) of the same equations.

Suggestion 2: You may want to start with the case when f does not depend on t.

HW # 16

Due:

Problem 1

Suppose that in the initial conditions (16.7), $\phi(x) \equiv 0$ and

$$\psi(x) = \begin{cases} 1, & 0 \le x \le 2 \\ 0, & \text{otherwise.} \end{cases}$$

(i) Use D'Alembert solution (16.8) with $c = 1$ to visualize $u(x,t)$ as a function of x for several values of t. See *technical notes* below. Print out (or draw by hand) as many *qualitatively different* profiles of the solution as there exist in this problem. Please be mindful of the amount of paper that you will use: combine several plots on the same page.
(ii) Predict (i.e., give a simple formula for) how wide the support[30] of the solution is for a given value of t. *Hint*: This is related to the characteristics of the Wave equation.

Technical notes:

1. An easy way to visualize the solution is with Mathematica, whose command for defining a piecewise function is `Piecewise`. Look it up in the Help menu. To specify that $0 < x < 2$, type `x>0 && x<2`. The command for the integration, which you will need to use to define $u(x,t)$, is `Integrate`. In it, you have to set `Assumptions->{Im[x]==0 && Im[t]==0}`; otherwise Mathematica cannot perform the integration.

2. The most convenient way to scan through several plots for different values of t is by using command `Animate`; see examples in the Help entry for this command. Vary t from 0.1 through 10 with a step 0.2. Plot u on the interval $x \in [-10, 10]$. Set the plotting option `PlotRange -> {-0.5, 2.5}` for easy viewing; also set the option `AnimationRunning -> False`.

Problem 2

Starting with (16.6) and (16.7), obtain (16.28) and then (16.8), as outlined in Appendix 1.

Problem 3 (worth **1.5 point**)

The Wave equation (16.3) can be solved by the method of Lectures 12 and 13: Simply replace each second derivative by its central-difference approximation.
(a) Write down the corresponding scheme.
(b) Perform its von Neumann stability analysis. Your result should be a condition on a parameter $\mu = c\kappa/h$ under which (the condition) this scheme is stable. This condition plays a fundamental role in numerical solution of hyperbolic equations and is called the Courant–Friedrichs–Lewy (CFL) condition.
Note 1: See the Note for Problem 10 of HW 5. I still expect you to present all the relevant calculations here.
Note 2: In analogy to HWs 12 and 13, denote $z = \mu^2\big(-1+\cos(\beta h)\big)$. You will need to consider cases $|1+z| < 1$ and $|1+z| > 1$ separately when analyzing the modulus of complex roots.

HW # 17

Due:

Problem 1 (worth **2 points**)

Use scheme (17.17) to obtain and plot the solution of problem (17.13), (17.14) with $a = 1$ at $t = 0, 0.25, 0.5, 0.75, 1$. Use $h = \kappa = 0.05$. *Interpret* your results. (For this part, googling for "shock wave equation" should help.)

[30] i.e., the interval of x where $u(x,t)$ is nonzero