VANDERBILT UNIVERSITY
CE 2989
NUMERICAL METHODS IN CIVIL & ENV. ENGINEERING

TAKE HOME FINAL EXAM
Ahmad F. Taha
April 18, 2025

Firm due date of this take home final exam: May 1st, 2025 at 11:59:59 CT.

- No extensions will be granted as the final grades are Due on May 3rd.
- You should show your code but also show the outcome of your codes, along with the needed mathematical derivations and some discussions regarding the observed results/outcomes. For example, when asked to compute eigenvalues, you are supposed to show the derivations and/or the computations.
- While I know that there is no way of enforcing it, I would appreciate it if you **do not discuss** this take home final exam with your classmates.
- You submitting the solutions explicitly entails that you have independently worked on the take home final exam, without using any NLP tools or searching for solutions online (not that you will find any).
- The quality and presentation of this take home final exam also matters, so please do not include sloppy figures or elusive code that goes out of the margins.
- I reiterate: You need to present adequate discussions on the results. Show the formulations, write the math down, discuss why the results make sense.
- Finally, each one of you is entitled to a zoom call with me to discuss any questions related to this exam. I am happy to accommodate that, so make sure you have a lot of questions for me (perhaps, ideally, couple of days before it is due or after you've spent a good amount of time solving the problems). You can also send a quick email if you suspect that there is a typo in the document.

1. *Lipschitz Constants.*

   We learned in class that computing Lipschitz constants, tight constants that upper bound the rate of changes of ODEs/IVPs, is fundamental in solving differential equations and proving existence and uniqueness. You are given this system of nonlinear ODEs $\dot{x}(t) = f(x,t)$ as follows:

$$\begin{aligned}
\dot{x}_1(t) &= -2x_1(t) - x_4^2(t) + 5x_2^2(t) + \cos(x_1(t)) \\
\dot{x}_2(t) &= -x_2(t) + x_3(t) \\
\dot{x}_3(t) &= -4x_3^2(t) + \sin(x_2(t)) \\
\dot{x}_4(t) &= -x_4(t)
\end{aligned}$$

   (a) *Analytical Lipschitz Constants.* Investigate the analytical computation of the Lipschitz constant K for $x \in \mathcal{X} \subset \mathbb{R}^4$ where $\mathcal{X}$ is the space containing all possible $x(t)$ defined as follows: $\mathcal{X} = \{0 \leq x_i \leq 2,\ i = 1,2,3,4\}$. That is, all the states are confined to a region $[0,2]$. Are you analytically able to find that Lipschitz constant? Make assumptions as needed.

   (b) *Analytical Bounds on the Jacobian.* An alternate way for computing the Lipschitz constant is to use the Jacobians (or partial derivatives, which yields a matrix in this case) of the vector-valued mapping $f(x)$ to compute an upper bound on K, as detailed in Slide 5 of Module 8. Can you analytically compute such K for the above system given the same $\mathcal{X}$ in the previous part?

   (c) *Numerical Computations of Lipschitz and Jacobian Constants.* Another approach to compute Lipschitz constants is via a numerical evaluation of the Lipschitz constant, as we discussed in class. You can use the Jacobians or the Lispchitz constant definition to compute a K numerically by basically searching through sampling across all the feasible points $x(t) \in \mathcal{X}$ and then evaluating the Lipschitz constant and the Jacobian. This yields a numerical approximation of the Lipschitz constant. Perform this task for the above system by heavily sampling thousands of data-points from $\mathcal{X}$ and showcase the computed constants and how they change as the number of sampling points change. This is an open ended problem that is meant to make you deeply think about the problem so I would like to see some innovation.

   Include your detailed codes and make sure you clearly explain what you are doing. Dumping only your codes in your solution is also not good enough; I need to see discussions, figures, and so on. I will not be running your code so showing your results and discussing them is very important.

   Hint: The paper https://ieeexplore.ieee.org/stamp/stamp.jsp?tp=&arnumber=8815016 basically shows some of these results for a nonlinear model of a power systems. In particular, Algorithm 1 in this paper demonstrates how to numerically compute the bounding constants. You might find this paper helpful, but reading it is not mandatory.

2. *Solving System of ODEs Analytically and Numerically.*

You are given the following system of linear ODEs with 4 states and 2 control inputs:

$$\begin{aligned}
\dot{x}_1(t) &= -2x_1(t) + 4x_2(t) + 3x_3(t) + u_1(t) \\
\dot{x}_2(t) &= -x_2(t) + 4x_4(t) \\
\dot{x}_3(t) &= -3x_3(t) + u_2(t) \\
\dot{x}_4(t) &= -8x_4(t).
\end{aligned}$$

(a) Write the above system as a matrix-vector ODE:

$$\dot{x}(t) = Ax(t) + Bu(t).$$

(b) Compute the eigen-decomposition of A **manually**. Note that this matrix is upper triangular, so you do not (and should not) use Matlab to compute the eigenvalues. You should also manually compute the eigenvectors resulting in this decomposition of matrix A: $A = TDT^{-1}$.

(c) Compute the matrix exponential of A given as e^A. What is e^{At} where t is the time scalar?

(d) Given that

$$x(t_0) = x(0) = \begin{bmatrix} 1 & -1 & 2 & 0 \end{bmatrix}^\top, \quad u(t) = \begin{bmatrix} u_1(t) \\ u_2(t) \end{bmatrix} = \begin{bmatrix} 1 - e^{2t} \\ 2\cos(t) \end{bmatrix}$$

compute $x(t)$ analytically. Show your work.

(e) Plot the four states $x_{1,2,3,4}(t)$ for a timespan of $t = [0, 5]$ seconds with an increment of $h = 0.01$ sec.

(f) An alternative to the analytical computation of $x(t)$ in the previous part (which is probably tedious), is to compute the value of the integration

$$\int_0^t e^{A(t-\tau)} Bu(\tau) d\tau = \int_0^t \begin{bmatrix} h_1(\tau, t) \\ h_2(\tau, t) \\ h_3(\tau, t) \\ h_4(\tau, t) \end{bmatrix} d\tau = \begin{bmatrix} \int_0^t h_1(\tau, t) d\tau \\ \int_0^t h_2(\tau, t) d\tau \\ \int_0^t h_3(\tau, t) d\tau \\ \int_0^t h_4(\tau, t) d\tau \end{bmatrix}$$

which entails integrating four functions of τ separately. Instead of analytically integrating, numerically integrate via your numerical integration method of choice considering again that the timespan is $t \in [0, 2]$ seconds. For this problem you can pick a different h. It is up to you. Then, plot the resulting numerical $x(t)$ solution for $t = [0, 2]$ seconds. For this problem, you can automate the process of computing the numerical integration. By the way, do not forget the additional term $e^{At}x(0)$.

(g) Compare the plots you got in Part (e) to the plots you got in (f). If you picked a good numerical integration method, your plots should be nearly identical. You should show your code too.

(h) Another alternate to everything you have done in the past few parts is to just use the ode45 solve on Matlab. Test the ode45 solver to solve the above system of ODEs and plot the states $x_{1,2,3,4}(t)$, with the given initial conditions, control inputs, and time-span. Does the ode45 solver on Matlab results in identical state trajectories to your analytical solution?

3. *Derivation of Numerical Solutions to ODEs.*

   In class, we derived the Adam-Bashforth two-step explicit method. Here, I want you to derive the Adam-Bashforth four-step **explicit** method:

$$y_{i+1} = y_i + \frac{h}{24}(55f_i - 59f_{i-1} + 37f_{i-2} - 9f_{i-3}).$$

   and the implicit Adam-Moulton two-step **implicit** method:

$$y_{i+1} = y_i + h\left(\frac{1}{12}f_{i-1} - \frac{2}{3}f_i + \frac{5}{12}f_{i+1}\right).$$

   What are the error order of these methods?

4. *4D Chess.*

The objective of this problem is to get you to investigate the performance of implicit multi-step methods versus explicit ones in solving ODEs. This problem gets you to think of the four dimensions of this problem and the impact of: sampling time h, order of the multi-step method m (i.e., how many past terms are considered), method to solve the implicit nonlinear system of equation, and the numerical method used to solve the problem (RK-2, RK-4, forward Euler, etc...). This problem is indeed lengthy but cant teach you a lot about numerical solutions of ODEs.

To approach this problem, we will consider a case study based on a problem from the textbook (Problem 28.32). The problem described here is different from the textbook problem. A pond drains through a pipe, as shown in the below figure.

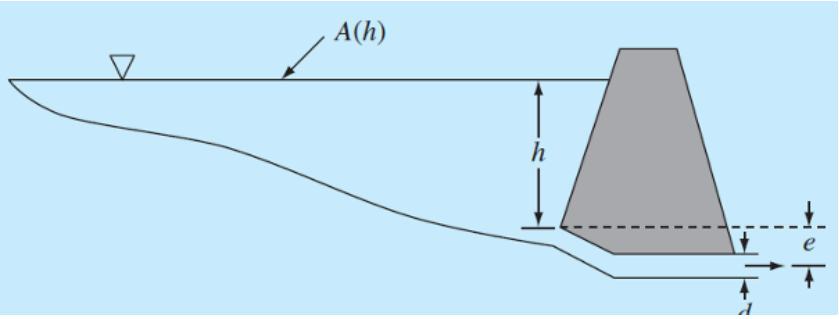

Figure 1: Pond draining through a pipe.

Under a number of simplifying assumptions, the following differential equation describes how depth changes with time:

$$\dot{h}(t) = -\frac{\pi d^2}{4A(h)}\sqrt{2g(h(t)+e)}$$

where $h(t)$ is depth (m) in t defines time (s), d is the pipe diameter (m), $A(h(t))$ is the pond surface area as a function of depth (m^2), g is the gravitational constant (equals 9.81 m/s^2), and e is depth of pipe outlet below the pond bottom (m). Consider the following parameters of this problem:

$$h(t_0) = h(0) = 6m,\ d = 0.25, e = 1.$$

Because this is indeed a long problem that will take a day or two to complete, I am showing you what the solution should look like in the below figure:

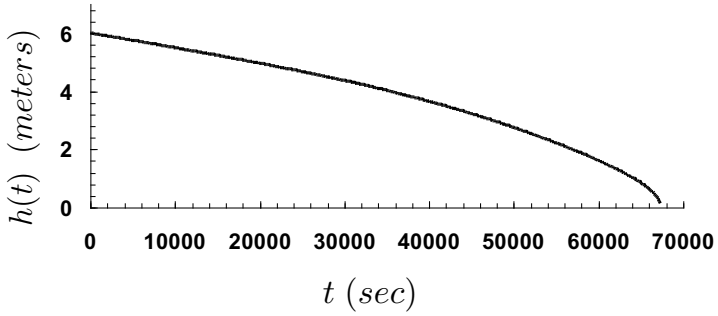

Figure 2: The solution to the IVP for around 70,000 seconds.

Table 1: Evaluations of the area $A(h(t))$ in 10^4 meter squares for different values of $h(t)$.

h in meters	6	5	4	3	2	1	0
$A(h)$ in 10^4 m^2	1.17	0.97	0.67	0.45	0.32	0.18	0

(a) The above table showcases the relationship between the area $A(h(t))$ in 10^4 meter squares for different values of $h(t)$. Obtaining closed form expression for this is a bit more difficult. Given that, and using a numerical method of your choice (Least Squares, Lagrange, Newton Divided Difference, etc..) obtain a closed form polynomial-based expression of $A(h)$ as a function of h. It is recommended that you use a fifth degree polynomial. If I were you, I'd start with obtaining a polynomial regression based on least squares then followed by testing another numerical interpolation method such as Newton Divided Differences. Note that this polynomial will have no constant terms in it as (you can see from the table) $A(0) = 0$.

(b) Plot a figure that demonstrates the data-points given in the table and the regression-based (or interpolation) model $A(h) = h^5 + a_1 h^4 + \dots$ that you obtained in the previous part.

(c) Now that you have a model for $A(h)$, you can plug that into the above nonlinear ODE. Write the newly obtained ODE down and solve it via Matlab's ode45 solver. Run the simulations for around 18 to 19 hours (be careful with the units).

(d) How how long did it take for the pond to be almost empty starting with the provided initial condition $h(0) = 6$? That is, what is t^* for which $h(t^*) \approx 0$? This is tricky because you end up getting singularity at $h(t) = 0$, so you can instead solve for $h(t^*) \approx \epsilon$ where ϵ is a smaller number like 0.01 m. You can solve this problem via running a nonlinear system of equation solver to obtain that t^* or via observing the plot in the previous problem (i.e., a graphical method).

(e) Can you compute the analytical solution of this nonlinear ODE? Try the Matlab dsolve command www.mathworks.com/help/symbolic/solve-a-single-differential-equation.html which obtains analytical solutions for linear/nonlinear ODEs. I frankly am not sure it will work here but I want you to try it out and report the result. Use the given intiail value $h(0) = 6$.

(f) Now that we have solved the ODE via Matlab, it is time to start investigating how the nonlinear ODE (with the obtained polynomial expression $A(h)$) can be solved numerically via the methods we learned in class.

First, and for different values of $h = 0.01, 0.1, 1, 10, 100$, test the following **seven numerical methods**: *(i)* first, second, and third order Taylor series approximation, *(ii)* modified Euler, *(iii)* midpoint Runge-Kutta method, and *(iv)* RK-4 and RK-5 methods. All of these methods were discussed in class, and their implementations are relatively straightforward.

(g) Now, considering that the ground truth is either your ode45-based solution in part (c), or the analytical solution in part (e), plot the absolute and relative error values for all of the methods tested in part (f). You should produce one plot for each h used. Basically, you should obtain five figures here, and each figure has seven plots for the seven numerical methods. Make sure that the figures are clear, the colors of your legend are nicely chosen, and that you save your figures as .EPS or .PDF files and not PNGs or JPGs; it is no longer high school.

Is there a consistent trend? What do we learn here?

(h) We are almost done with the class here, which gets us to the most important part of the class where we analyze the performance of implicit and explicit, multi-step methods in solving the nonlinear ODE.

In particular, I want you to investigate whether it is possible to use a much larger sampling time (for example, but not necessarily, $h = 1000$ seconds for example) for an implicit multi-step method in comparison with the a smaller time-step for an explicit multi-step method. I also want you to investigate the impact of the numerical method used to solve for the

To solve this problem, do the following:

i. Using the first few initial conditions via the RK-4 or modified Euler, and implementing the explicit and implicit 4-step methods on Slide 21/30 of Module 8, analyze the performance of these two methods in solving the above IVP with different sampling times $h = 0.1, 1, 10, 100, 1000$. When using the implicit method, use Newton's method to solve for the $i+1$ value of $h(t)$. Do you encounter any issues? Report the results (and compare with the analytical/ODE solution) and tabulate the computational time required by the two methods for various values of h.
   A. Which method required more computational time? And why?
   B. Does the choice of the method used to find the initial conditions (RK-4 or modified Euler) impact the state trajectory of $h(t)$? Does it make any meaningfully tangible difference?

ii. Repeat the previous problem but instead of Newton's method (for the implicit iteration), use another method (for solving nonlinear system of equations) of your choice to solve the problem. Do you see any notable difference? What would it be?

iii. Now instead of this approach, you can use a very common approach called the *predictor-corrector method*. I did not mention it as much in the class, but the main idea is highlighted in the last few bullet points of Slide 24/30 from Module 8 and then illustrated via the figure on Slide 25/30. Implement the Fourth-Order Adams-Bashforth predictor-corrector method from Slide 25/30 and report the results for different values of h.

iv. Is this method more computationally efficient than utilizing Newton's method at each iteration? Compare the computational time as well as the errors that result from this method.

The class is henceforth complete. Thank you for taking this class and for tolerating my bad jokes and terrible takes. I know it was a lot of hard work to take the class, and I am very proud of your hard work and the effort you put. I am excited to see your solutions and to give feedback on how you have done in class and on the take home final.