Intro to ODEs in Matlab

Jeremy L. Marzuola

Charles Talbot

Ben Wilson

E-mail address: marzuola@math.unc.edu

Department of Mathematics, UNC-Chapel Hill, CB#3250 Phillips Hall, Chapel Hill, NC 27599 USA

E-mail address: ctalbot@live.unc.edu

Department of Mathematics, UNC-Chapel Hill, CB#3250 Phillips Hall, Chapel Hill, NC 27599 USA

E-mail address: wilsonbn@live.unc.edu

Department of Mathematics, UNC-Chapel Hill, CB#3250 Phillips Hall, Chapel Hill, NC 27599 USA

Abstract. An Introduction to ODEs using Matlab developed to enhance modeling and computational skills in the First Course in Differential Equations at the University of North Carolina.

Dedication

This book is dedicated to Ingrid and Virginia, whose sleeplessness during the night assisted in writing many a chapter. Also, special thanks to Moshe Feldman, Dylan Gooch, Julia Sampson, Aric Wheeler, Yaoxuan Xia, Zhongcheng Xiao and Xiaohan Yi for being willing to stick with the course in its first iteration, pointing out where things could be clarified in early versions of the problem sets and working so hard in the Fall of 2014 to get as much from the first Lab course as possible. The Environmental Engineering worksheet was co-developed with Marc Alperin from Marine Sciences at UNC and we are grateful that he participated and created such an interesting model application. Special thanks to Katie Newhall and Greg Forest for looking over early versions and giving helpful comments. This course was also taken on as part of JLM's NSF Career Grant DMS-1352353.

Contents

CHAPTER 1

Introduction

The goal for this class will be to learn how to approximate solutions to equations (or systems of equations) of the form

$$y' = f(x, y), y(0) = 0. \tag{0.1}$$

Many solutions can be found by looking at an equation such as this by integrating it out and using the initial condition

$$y(x) = y(0) + \int_0^x f(z, y(z))dz. \tag{0.2}$$

This looks like we have not gained much, but as we will see, with a little linear algebra we will see that we have gained quite a lot. To approximate solutions of this form, we will learn some basic algorithms and implement them in Matlab. Throughout, you are welcome work on implementing similar codes in other languages that are good for computation, such as Fortran, Python or C++ if you are interested.

CHAPTER 2

Week 1

1. Discussion Topics

Turning second order equations into systems, approximation of ODEs, linear algebra in ODE

2. Worksheet

1. Download Matlab!! Play with the Help function to learn some of its key features.
2. Check out the tutorials from [**1**] to get a sense of some of the implicit *Matlab* capabilities.

CHAPTER 3

Week 2

1. Discussion Topics

Intro to variables and allocation in *Matlab*, Matrices, Vectors

2. Worksheet

1. Scalar variables in *Matlab* (use implicit *Matlab* components like **exp**, **e**, **i**, **sqrt**, **real**, **conj**, **log**):
a. Create $x = 25.0$;
b. Create $y = 1.4 \times 10^4$;
c. Create $z = e^{i\pi/4}$;
d. Evaluate $(\sqrt{x} + y^{-1/4})^{\pi}$; e. Evaluate $\log(z\bar{z})$.

2. Vector variables in *Matlab* (use implicit *Matlab* components like **linspace**, $.\wedge$, i, π, **plot**:
a. $\vec{x} = .1[-100\ -99\ \ldots\ 99\ 100]$, a 1×201 row vector;
b. $\vec{y} = \begin{bmatrix} 10^{-2} \\ 10^{-1} \\ 10^{0} \\ 10^{1} \\ 10^{2} \end{bmatrix}$, a 5×1 column vector;
c. $\vec{z} = \begin{bmatrix} e^{-i\pi} & e^{-i3\pi/4} & e^{-i\pi/2} & e^{-i\pi/4} & e^{0} & e^{i\pi/4} & e^{i\pi/2} & e^{i3\pi/4} & e^{i\pi} \end{bmatrix}$, a $1x9$ row vector;
d. Create a vector with pointwise components $\vec{b} = \frac{1}{\sqrt{2\pi}} e^{-\vec{x}/2}$;
e. Plot $\vec{b}$ versus $\vec{x}$.

3. Make the following matrix variables in *Matlab* (use implicit *Matlab* components like **linspace**, **ones**, **zeros**, **diag**, **rand**, **floor**, **ceil**):
a. $M_1 = \begin{bmatrix} 1 & \ldots & 1 \\ \vdots & \ddots & \vdots \\ 1 & \ldots & 1 \end{bmatrix}$, a $5x5$ matrix of all 1's;
b. $M_2 = \begin{bmatrix} 1 & 0 & \ldots & 0 \\ 0 & \ddots & 0 & \vdots \\ \vdots & 0 & \ddots & 0 \\ 0 & \cdots & 0 & 10 \end{bmatrix}$, a $10x10$ matrix with non-zero entries only on the diagonal;

c. $M_3 = \begin{bmatrix} 5 & 0 & 5 & 0 \\ 0 & 5 & 0 & 5 \end{bmatrix}$;
d. A $3x3$ matrix with random values between -2 and 2.

CHAPTER 4

Week 3

1. Discussion Topics

Matrices, Vectors, Gaussian Elimination, Solving Linear Systems in *Matlab*

2. Worksheet

1. Solve the linear system of equations:

$$\begin{aligned} 5x_1 + 2x_2 + 3x_3 &= 0, \\ x_1 - 2x_2 + x_3 &= 0, \\ x_1 + x_2 - x_3 &= 0. \end{aligned}$$

In Matlab, use the operation $A\backslash b$ with

$$A = \begin{bmatrix} 5 & 2 & 3 \\ 1 & -2 & 1 \\ 1 & 1 & -1 \end{bmatrix}$$

the "coefficient matrix" and

$$b = \begin{bmatrix} 0 \\ 0 \\ 0 \end{bmatrix}$$

the right hand side. Verify that the output indeed solves the equation.

2. Solve the following linear systems of equations (use implicit Matlab components like \, **lsqr**):
a.

$$\begin{aligned} 5x_1 + 2x_2 + 3x_3 &= 1, \\ x_1 - 2x_2 + x_3 &= 2, \\ x_1 + x_2 - x_3 &= 1; \end{aligned}$$

b.

$$\begin{aligned} x_1 + 2x_2 &= 1, \\ -2x_1 - 4x_2 &= -2; \end{aligned}$$

c. Is the following system solvable?? If so, solve it. If not, find the "best possible" solution.

$$\begin{aligned} 5x_1 + 2x_2 + 3x_3 &= 1, \\ x_1 - 2x_2 + x_3 &= 2, \\ 4x_1 + 4x_2 + 2x_3 &= 1. \end{aligned}$$

3. Create the following functions (use implicit Matlab components like **i**, **sqrt**, **log**, **tanh**):

a. $f_1(x) = x^2 e^{-x^2}$;

b. $f_2(x) = \frac{\tanh(x)}{\sqrt{1+x^2}}$;

c. $f_3(x) = x - \frac{x^3}{3!}$;

d. For $g = 1.0$, $L = 1.0$ with

$$f_4(t, v, h) = \begin{bmatrix} v \\ \frac{g}{L}\sin(h) \end{bmatrix}$$

for g and L ;

e. $f_5(x) = \frac{1}{\sqrt{|x|\ \ |\log x|^2}}$.

CHAPTER 5

Week 4

1. Discussion Topics

Making functions in Matlab

2. Worksheet

1. Create an inline *Matlab* function for a map

$$f : \mathbb{R}^3 \to \mathbb{R}^3$$

such that

$$f\begin{pmatrix} x_1 \\ x_2 \\ x_3 \end{pmatrix} = \begin{pmatrix} \sin(x_1 - x_2) \\ x_3 e^{2x_1} \\ x_1^2 + x_2^2 + x_3^2 \end{pmatrix}.$$

Write the function first assuming one inputs the components of the input vector (x_1, x_2, x_3), then assuming one inputs a general *Matlab* variable x, which one will use as having 3 components.

2. Instead of writing an inline function in your .m file, write the above functions as function files f1.m and f2.m in your worksheet directory.

3. *Matlab* ODE solvers allow one to solve equations of the form

$$y' = f(t, y),\ y(t_0) = y_0$$

by giving the function $f(t, y)$ as input. However, these methods can still be used to solve higher order equations if they are entered correctly by allowing y and f to both be column vectors instead of scalars. For the equation

$$y'' + y' + y^2 = \sin(t)$$

convert the 2nd order ODE into a first order system of ODEs and create the *Matlab* function that inputs y as a 2-vector and outputs $f(t, y)$ as a 2-vector.

CHAPTER 6

Week 5

1. Discussion Topics

Numerical integration, Plotting in *Matlab*, *Matlab* ODE Solvers

2. Worksheet

1. Recall the following function from Week 3 creating $f_1(x) = x^2 e^{-x^2}$;
$f_2(x) = \frac{\tanh(x)}{\sqrt{1+x^2}}$;
$f_3(x) = x - \frac{x^3}{3!}$;
For $g = 1.0$, $L = 1.0$ with

$$f_4(t, v, h) = \begin{bmatrix} v \\ \frac{g}{L}\sin(h) \end{bmatrix}$$

for g and L ;
$f_5(x) = \frac{1}{\sqrt{|x|\ |\log x|^2}}$. Numerically integrate the following (use implicit Matlab components like **sum**, **quad**, **trapz**, **integral**, **abs**):
a. $f_1(x)$ from $x = -10$ to 10;
b. $f_2 * f_3$ from $x = 0$ to 2;
c. Investigate the integral of $(f_5(x))^2$ from $x = 0$ to 1;
d. Do part c with $(f_5(x))^4$???

2. Make the following plots in Matlab (use implicit Matlab components like **quiver**, **polyfit**, **polyval**, **plot**):
a. A cubic and quartic polynomial approximation to $\sin(x)$ on the interval from 0 to 2π. Plot the sin curve and your polynomial approximations on this interval.
b. A direction field plot of the logistic population growth model

$$\frac{dy}{dt} = 10 * (1 - y)(y - 2).$$

3. Use **ode45** or **ode15s** and function f_4 from above to solve the pendulum equation

$$\frac{d^2\theta}{dt^2} + \frac{g}{L}\sin\theta = 0,$$

for $g = L = 1.0$ and $\theta(0) = .1$, $\theta'(0) = 0$.
b. Download **pplane7** from `http://math.rice.edu/~dfield/`. Using that software:
1. Create a phase diagram for the pendulum equation with $g = L = 1.0$.
2. What happens if $g = 1.0$ and $L = 5.0$? $g = .17$ and $L = 1.0$?? In other words,

what happens to a really long pendulum OR if you put a pendulum on the moon??
3. Compare to

$$\frac{d^2\theta}{dt^2} + \frac{g}{L}\theta = 0.$$

Attach the 4 .pdf files to the e-mail sent to the TA.

CHAPTER 7

Week 6

1. Discussion Topics

Determinants, Eigenvalues, Eigenvectors and For Loops in *Matlab*

2. Worksheet

1. Compute the determinants of the following matrices. In your matlab output, use an **if/then** statement to print out either the term "invertible" or "singular" based upon the outcome (use implicit Matlab components like **det**, **eye**, **if**, **then**, **end**, **disp**):
a.

$$M_2 = \begin{bmatrix} 1 & 2 \\ 3 & 4 \end{bmatrix}$$

b.

$$M_3 = \begin{bmatrix} 1 & 0 & 1 \\ 2 & 1 & -1 \\ 4 & 3 & 2 \end{bmatrix}$$

c. $M_4 - I$ for

$$M_4 = \begin{bmatrix} 1 & 1 & 0 & 0 \\ 0 & 1 & 0 & 0 \\ 0 & 0 & 1 & 1 \\ 0 & 0 & 0 & 1 \end{bmatrix}$$

d. Write a matlab function called **mydet** that inputs a $2x2$ matrix and outputs the value of its determinant.

2. Find the eigenvalues and eigenvectors of the following matrices (use implicit Matlab components like **eigs**, **disp**, **det**, **if**, **then**, **end**). In each case, print out whether or not the eigenvectors form a basis for the space $\mathbb{R}^n$ corresponding to M_n:
a. M_2;
b. M_3;
c. M_4 .

3. Use a for loop construction to write your own ODE solver for the forced spring equation

$$y'' = -y + .1\sin(t), \quad y(0) = 0$$

(use implicit Matlab components like **for**, **end**) over the interval $t = 0$ to $t = 10$ implementing 1000 uniform time steps for the schemes:
a. Explicit Euler;
b. Implicit Euler;
c. Solve the same equation using **ode15s** with an error tolerance of $1e - 10$ letting matlab set the number of time steps.
d. Plot your approximations $y(t)$ from all three approximations on the same plot.

CHAPTER 8

Week 7

1. Discussion Topics

More on *Matlab* ODE Solvers and Newton's Method

2. Worksheet

1. Use **ode45** or **ode15s** to solve the following ODEs. Plot their solutions over the given time interval and address the physical question posed:
a. Newton' laws for a falling object of mass 1 kg and coefficient of friction .2 kg/s -

$$\frac{d^2x}{dt^2} + .2\frac{dx}{dt} = 9.8,$$

for $x(0) = -10$, $x'(0) = 0$ over a time interval $t \in [0, 1]$. Print out an approximate time at which the object hits the ground ($x = 0$);
b. The logistic equation

$$\frac{dy}{dt} = 10(10 - y)y,$$

with $y(0) = 5$ over a time interval $t \in [0, 2]$. Print out how close y is to its equilibrium ($y = 10$) at the final time of your simulation. See [**2**, Chapter 2.5] for more on the logistic equation;
c. The first 4 Hermite equations

$$\frac{d^2y}{dx^2} - 2x\frac{dy}{dx} = 2ny,$$

for $n = 0, 1, 2, 3$ with with $y_0(0) = 1$, $y_0'(0) = 1$; $y_1(0) = 0$, $y_1'(0) = 1$; $y_2(0) = 2$, $y_2'(0) = 0$; $y_3(0) = 0$, $y_3'(0) = -12$. Solve both forwards and backwards in time to show all 4 functions on the interval $x \in [-10, 10]$. This is an important set of functions in mathematical physics and will be discussed in many physics texts on quantum mechanics.

2. a. Use a *Matlab* ODE solver to compute solutions to

$$y''(t) + \sin(t)y'(t) + \cos(t)y = 0$$

first for $y_1(0) = 1$ and $y_1'(0) = 0$, then for $y_2(0) = 1$ and $y_2'(0) = 1$ on the interval between $[0, 2\pi]$.
b. Compute the function

$$W(t) = \det \begin{bmatrix} y_1(t) & y_2(t) \\ y_1'(t) & y_2'(t) \end{bmatrix}$$

at the values of t you have approximated.
c. Approximate the ODE

$$z' + \sin(t)z = 0, \quad z(0) = 1$$

on the interval between $[0, 2\pi]$.
d. Plot $W(t)$ and $z(t)$ both on $[0, 2\pi]$ to compare. This is a numerical verification of a famous theorem of Abel, which is discussed at length in [**2**].

3. a. Write a **for** loop root finder implementing Newton's method when a function and its derivative are both known.
b. Use 10 iterations of your Newton solver to approximate the roots (make a first guess by plotting if you like) of

$$f(x) = e^x - 10\cos(x).$$

c. For one of the roots, plot with x's of the first 10 differences of iterates ($|x_1 - x_0|$, $|x_2 - x_1|$, $\ldots$, $|x_{10} - x_9|$).
d. Plot $f(x)$ and the points that you have found from part b. as o's on the curve.

CHAPTER 9

Week 8

1. Discussion Topics

Writing your own ODE solvers

2. Worksheet

1. Use a for loop construction to write your own ODE solver for the following nonlinear equation

$$y' = -y + \sin(y^2) + .1\sin(t), \quad y(0) = 0$$

(use implicit Matlab components like **for**, **end**) over the interval $t = 0$ to $t = 10$ implementing 1000 uniform time steps for the schemes:
a. Explicit Euler;
b. Implicit Euler (this will require solving a nonlinear equation - use Newton's method above;
c. Solve the same equation using **ode15s** with an error tolerance of $1e-10$ letting matlab set the number of time steps.
d. Plot your approximations $y(t)$ from all three approximations on the same plot.

2. Solve the following system of first order equations related to a particle moving in a $2d$ gravitational field from [**8**] using your favorite ODE solver:
a.

$$\begin{aligned} x' &= u \\ y' &= v \\ u' &= -\frac{x}{x^2+y^2} \\ v' &= -\frac{y}{x^2+y^2} \end{aligned}$$

with initial data $(x, y, u, v)(0) = (1, 0, 0, 1)$ for time $[0, 10]$;
b.

$$\begin{aligned} x' &= u \\ y' &= v \\ u' &= -\frac{x}{x^2+y^2} \\ v' &= -\frac{y}{x^2+y^2} \end{aligned}$$

with initial data $(x, y, u, v)(0) = (1, 0, 0, 0)$ for time $[0, 10]$;
c.

$$\begin{aligned} x' &= u \\ y' &= v \\ u' &= -\frac{x}{x^2 + y^2} \\ v' &= -\frac{y}{x^2 + y^2} \end{aligned}$$

with initial data $(x, y, u, v)(0) = (1, 0, 2, 2)$ for time $[0, 10]$;
d. Plot the three orbits $(x(t), y(t))$ from parts a.-c. on the same plot.

3. Solve the following nonlinear 3rd order equation related to the evolution of thin films

$$u_{xxx} + .2u_{xx} + u_x + \frac{1}{2}(u^2) = .4,$$

a. using **ode15s** for 25 initial data in a neighborhood of $(u, u_x, u_{xx}) = \sqrt{.8}, 0, 0)$ on a time interval from $[0, 10]$;
b. Plot the 25 three dimensional orbits $(u(t), u_x(t), u_{xx}(t))$ on a $3d$ line plot.

CHAPTER 10

Week 9

1. Discussion Topics

Approximation of ODEs, Special Functions

2. Worksheet

1. Early convergence tests. Problem 1.a. has an exact solution $(x, y, u, v)(t) = (\cos t, \sin t, -\sin t, \cos t)$. The explicit Euler method is known as 1st order, in that it is accurate only up to order τ for Write an explicit Euler solver with the number of time steps given by $N = 100, 200, 300, 400, 500$ and approximate the solution over the time interval $[0, 4\pi]$ for each N. Make a vector of the maximum error

$$\text{error}_N = \max\{|x_n - \cos t_n|, |y_n - \sin t_n|, |u_n + \sin t_n|, |v_n - \cos t_n|\}.$$

Use **tic** and **toc** to find the time taken per step of each method. Based upon your data points, estimate the constant C such that the error looks like $C\tau$ for $\tau = \frac{4\pi}{N}$ by fitting a linear curve to your 5 data points.

2. The Airy function solves the ODE $y'' - xy = 0$ and was designed to model light moving through a caustic (such as diffraction that forms a rainbow for instance). Note, for $x < 0$, this forms a positive coefficient and for $x > 0$, a negative one. Use an ODE solver of your choice and the *Matlab* function **gamma** for the Γ function to plot a special fundamental solutions given by $y(0) = \frac{1}{3^{2/3}\Gamma(2/3)}$, $y'(0) = \frac{-1}{3^{1/3}\Gamma(1/3)}$ on an interval $x \in [-10, 10]$. Plot the solution and notice that the plots are oscillatory on one side of the origin and notice that they are oscillatory on one side and exponentially decaying on the other. Note, to see this behavior numerically, you might have to be careful with your error tolerances! This is a special function called the Airy function. Compare your function to the implicit *Matlab* function **airy** on the same interval using a plot (note, the Airy function in *Matlab* will have a tiny imaginary component, so you will need to take the real part using **real**).

3. The Bessel functions or order 0 solves the ODE $x^2y'' + xy' + x^2y = 0$ and is related to modes of oscillation on discs and many other physical examples. Use an ODE solver of your choice to solve this equation with $(y, y')(0) = (1, 0)$ and $(y, y')(0) = (0, 1)$. Note, be careful near $x = 0$ given that the coefficient functions are singular at such a point. Compare your functions to the implicit *Matlab* functions **besselj** and **besselk** on the same interval using a plot.

4. (Optional) The Airy function plays a role in solving the Airy equation (linear Korteweg-de Vries equation)

$$u_t + u_{xxx} = 0,\ u(0,x) = u_0,$$

which can be seen to be equivalent to solving the simple first order ordinary differential equation

$$f_t - ik^3 f = 0,\ f(0) = \hat{u}_0(k)$$

for each k using the theory of the Fourier Transform. If you have never seen this before, it is a major tool in scientific computing As a quick demonstration of the correspondence between the Airy equation and the Airy function and an application of the Fourier Transform in Matlab, play with the Matlab function **fft** acting on your solution from part *a* and compare to the complex function $e^{i(2\pi k)^3/3}$ by plotting the real and imaginary parts on the same axis. Note, you will need to plot carefully and use some scaling to make **fft** compare properly due to Matlab conventions.

CHAPTER 11

Week 10

1. Discussion Topics

Stability and Instability in ODE algorithms

2. Worksheet

1. The region of absolute stability for an approximation algorithm to an ODE is the set of $z = h\lambda$'s in the complex plane for which the method when applied to

$$u' = \lambda u,\ u(0) = u_0$$

stays bounded as $n \to \infty$ for any initial data. Note, an exact solution will remain bounded provided the real part of λ is negative. We will numerically explore this region for a variety of methods. To this, choose $y_0 = 1$ and a range of z in a complex box of size $[-10, 10] \times [-10, 10]$ with a grid of size 101×101. For each z, write a **for** loop to test if the given method will remain bounded and plot the region of absolute stability by marking each stable z with a + according to your test:
a. Explicit Euler

$$u_{n+1} = u_n + hf(t_n, u_n) = (1 + h\lambda)u_n = (1 + z)u_n,$$

b. Implicit Euler

$$u_{n+1} = u_n + hf(t_{n+1}, u_{n+1}) = (1 - h\lambda)^{-1}u_n = (1 - z)^{-1}u_n,$$

c. Trapeziodal Rule

$$u_{n+1} = u_n + \frac{h}{2}\left(f(t_n, u_n) + f(t_{n+1}, u_{n+1})\right) = \frac{1 + \frac{h\lambda}{2}}{1 - \frac{h\lambda}{2}}u_n = \frac{1 + z/2}{1 - z/2}u_n.$$

2. Sometimes implicit methods can be approximated by explicit methods. The simplest version of this is an explicit Runge-Kutta scheme, which approximates the implicit dependence of the trapezoid rule with a step of forward Euler. In other words, the simplest Runge-Kutta scheme is of the form

$$\begin{aligned} k_1 &= f(t_n, u_n), \\ k_2 &= f(t_n + h, u_n + hk_1), \\ u_{n+1} &= u_n + h\left(\frac{k_1}{2} + \frac{k_2}{2}\right) \end{aligned}$$

for time step h. This scheme is of order 2. Similar to the Worksheet from Week 9, solve

$$\begin{aligned} x' &= u \\ y' &= v \\ u' &= -\frac{x}{x^2+y^2} \\ v' &= -\frac{y}{x^2+y^2} \end{aligned}$$

with initial data $(x, y, u, v)(0) = (1, 0, 0, 1)$ with the number of time steps given by $N = 100, 200, 300, 400, 500$ and approximate the solution over the time interval $[0, 4\pi]$ for each N. Make a vector of the maximum error

$$\text{error}_N = \max\{|x_n - \cos t_n|, |y_n - \sin t_n|, |u_n + \sin t_n|, |v_n - \cos t_n|\}.$$

Use **tic** and **toc** to find the time taken per step of each method. Based upon your data points, estimate the constant C such that the error looks like $C\tau^2$ for $\tau = \frac{4\pi}{N}$ by fitting a quadratic curve to your 5 data points. Plot the points on a log-log plot.

CHAPTER 12

Week 11

1. Discussion Topics

Implementing Newton's Method in Algorithms and RK4, Lotka-Volterra Model

2. Worksheet

1. The linearly implicit Euler scheme for first order systems implements one step of Newton's method to solve implicit Euler and as a result looks like

$$(I - hDf(t_{n+1}, u_n))(u_{n+1} - u_n) = hf(t_{n+1}, u_n).$$

Use this scheme to write a solver for the forced and damped pendulum problem

$$\frac{d^2\theta}{dt^2} + .05\frac{d\theta}{dt} + \sin\theta = \sin(t), \quad \theta(0) = \pi/4, \ \theta'(0) = 0$$

up to time $T = 12.0$. Plot the orbits of your damped pendulum by plotting $(\cos\theta(t), \sin(\theta(t))$ in the plane.

2. The 4th order Runge-Kutta Algorithm (RK4) attempts to carefully make numerical integration very accurate between time steps by choosing a certain number of points in the interval to evaluate a function. However, those points then need to be approximated by values stemming from the left end-point! The algorithm goes as follows

$$\begin{aligned}
k_1 &= f(t_n, y_n), \\
k_2 &= f(t_n + \frac{h}{2}, y_n + \frac{h}{2}k_1), \\
k_3 &= f(t_n + \frac{h}{2}, y_n + \frac{h}{2}k_2), \\
k_4 &= f(t_n + h, y_n + hk_3), \\
y_{n+1} &= y_n + \frac{h}{6}(k_1 + 2k_2 + 2k_3 + k_4),
\end{aligned}$$

which uses effectively Simpson's Rule for numerical integration, but approximating the mid-point as an average of explicit Euler and linearly implicit Euler approximations. There are a large family of these methods, both explicit as above and implicit, that use the same ideas. Use (RK4) to solve the following Lotka-Volterra

Model from [**7**]

$$x' = x(3 - x - 2y),$$
$$y' = y(2 - x - y),$$
$$(x, y)(0) = (0, .1).$$

Plot the evolution of x and y on a time interval $[0, 10]$. The model listed here has fixed points at

$$(x, y) = (0, 0),\ (0, 2),\ (3, 0),\ (1, 1).$$

In a later worksheet, we will determine how stable each of them is, but for this worksheet, is $(0, 0)$ stable?? Print out using Matlab your response.

3. (Optional) Boundary value problems play a major role in physical applications. One simple example is to find positive solutions to the equation

$$u_{xx} - u + u^3 = 0,\ u(0) = \alpha,\ u(20) = 0.$$

Use an ODE solver of your choice to find α such that by solving forward from $u(0) = \alpha$, $u'(0) = 0$, u is positive and does not cross the x-axis prior to $x = 20$. You could also play with **bvp4**, Matlab's implicit boundary value solver.

CHAPTER 13

Week 12

1. Discussion Topics

Multi-Step Methods and Crank-Nicholson, Lorenz Equations, Optics

2. Worksheet

1. The Adams-Bashforth Algorithm (see [**8**]) is designed around being accurate for polynomials of a chosen order by using more than just nearest neighbor points to create an algorithm. The algorithm is a multi-step method and goes as follows

$$y_{n+2} = y_{n+1} + \frac{3}{2}hf(t_{n+1}, y_{n+1}) - \frac{1}{2}hf(t_n, y_n).$$

Use this simple Adams-Bashforth algorithm below for solving the Lorenz equations

$$\begin{aligned} x' &= (y - x), \\ y' &= x(1 - z) - y, \\ z' &= xy - z, \\ (x, y, z)(0) &= (1, 1, 0) \end{aligned}$$

on time scale $[0, 15]$ with $1e4$ time steps. For more on the Lorenz equations, see [**7**]. Hint: Approximate y_1 with a step of Explicit Euler.

2. The lowest order Crank-Nicholson scheme essentially implements the mid-point rule. The algorithm is

$$y_{n+1} = y_n + \frac{1}{2}h\left(f(t_{n+1}, y_{n+1}) + f(t_n, y_n)\right).$$

Use this for solving problem that actually turns out to be not that far from being a pendulum

$$\begin{aligned} \alpha' &= (1 + 2\alpha^2)\beta, \\ \beta' &= -(4\alpha^2 + 2\beta^2 - 1)\alpha, \\ (\alpha, \beta)(0) &= (0, 1) \end{aligned}$$

on a time interval $[0, 10]$ with $1e4$ time steps. Note, a Newton's Method will be required! Compare the values of α, β to those from solving

$$\begin{aligned} \alpha' &= (1 + 2\alpha^2)\beta, \\ \beta' &= -(1 + 2\alpha^2 - 2A^2)\alpha, \\ A' &= -2\alpha\beta A, \\ (A, \alpha, \beta) &= (0, 0, 1) \end{aligned}$$

using an implicit Matlab solver. Track the quantity $A^2 + \alpha^2 + \beta^2$ throughout your solution. This model was derived in [**3**].

3. (Optional) Solving the Lorenz equations

$$x' = \sigma(y - x), y' = x(\rho - z) - y, z' = xy - \beta z,$$

using any solver of your choosing. The choice of ρ, σ and β determines dynamics. Play with making a $3d$ plot by solving the Lorenz equations over the time interval $I = [0, 15]$ using $(\sigma, \rho, \beta) = (10, 28, 8/3)$ for a family of 1000 different initial conditions (use a for loop!). The figure you are making is a very famous thing in dynamical systems called the "Lorenz attractor." For pictures of these orbits, see [**7**].

CHAPTER 14

Week 13

1. Discussion Topics

Hamiltonians, Newton's Method in higher dimensions, RK Schemes

1.1. Newton's Method in higher dimensions. To implement Newton's Method in all dimensions, recall that Newton's Method centers around making a linear approximation to the function near a root. Hence, we need to recall for a moment how to make a Taylor Series expansion in higher dimensions. Indeed, for a sufficiently regular function

$$F(\vec{x}) = \begin{bmatrix} F_1(\vec{x}) \\ F_2(\vec{x}) \\ \vdots \\ F_d(\vec{x}) \end{bmatrix} : \mathbb{R}^d \to \mathbb{R}^d \tag{1.1}$$

we have the expansion

$$F(\vec{x} + \epsilon\vec{z}) = F(\vec{x}) + \epsilon DF(\vec{x})\vec{z} + \mathcal{O}(\epsilon^2), \tag{1.2}$$

where DF is the Jacobian matrix

$$\begin{bmatrix} \frac{\partial F_1}{\partial x_1} & \frac{\partial F_1}{\partial x_2} & \cdots & \frac{\partial F_1}{\partial x_d} \\ \frac{\partial F_2}{\partial x_1} & \frac{\partial F_2}{\partial x_2} & \cdots & \frac{\partial F_2}{\partial x_d} \\ \vdots & \ddots & \ddots & \vdots \\ \frac{\partial F_d}{\partial x_1} & \frac{\partial F_d}{\partial x_2} & \cdots & \frac{\partial F_d}{\partial x_d} \end{bmatrix} \tag{1.3}$$

and by $\mathcal{O}(\epsilon^2)$ means that all remaining terms can be bounded as operators by $C\epsilon^2$ for some constant $C > 0$. Note, the second order and higher terms relate to important tensor operators. In other words, if we want to find a root

$$F(\vec{x}^*) = 0 \tag{1.4}$$

given some initial guess $\vec{x}_0$, we solve for the root of the linear approximation

$$\vec{0} = F(\vec{x}_0) + DF(\vec{x}_0)(\vec{x}_1 - \vec{x}_0).$$

This gives the relation

$$\vec{x}_1 = -(DF(\vec{x}_0))^{-1}F(\vec{x}_0) + \vec{x}_0.$$

Then, of course to find a more accurate root, we iterate the procedure, solving

$$\vec{x}_{n+1} = -(DF(\vec{x}_n))^{-1}F(\vec{x}_n) + \vec{x}_n.$$

Note, we really want $\vec{x}_0$ to be a good guess in Newton's method for the root so that $\vec{x}_1 - \vec{x}_0$ is small since it is playing the role of $\epsilon\vec{z}$ in (1.2).

EXAMPLE 1.1. *Let us take a simple 2d model problem. Take*

$$\vec{u} = \begin{bmatrix} u_1 \\ u_2 \end{bmatrix} \tag{1.5}$$

and

$$F(\vec{u}) = \begin{bmatrix} (u_1^2 + 1)u_2 \\ (u_1^2 + u_2^2 - 1)u_1 \end{bmatrix}, \tag{1.6}$$

which has roots $(0,0)$ *and* $(1,0)$*. To implement Newton's Method to find said roots, one would need to compute the Jacobian matrix*

$$\begin{bmatrix} \frac{\partial F_1}{\partial u_1} & \frac{\partial F_1}{\partial u_2} \\ \frac{\partial F_2}{\partial u_1} & \frac{\partial F_2}{\partial u_2} \end{bmatrix} = \begin{bmatrix} 2u_1u_2 & u_1^1 + 1 \\ 3u_1^2 + u_2^2 - 1 & 2u_2u_1. \end{bmatrix} \tag{1.7}$$

Note that near $(0,0)$*, the Jacobian takes the form*

$$\begin{bmatrix} 0 & 1 \\ -1 & 0 \end{bmatrix} \tag{1.8}$$

which has determinant 1 *and is hence invertible. Near* $(1,0)$*, the Jacobian takes the form*

$$\begin{bmatrix} 0 & 2 \\ 2 & 0 \end{bmatrix} \tag{1.9}$$

which has determinant -4 *and is hence invertible. As a result, provided our guesses are close enough to the roots, the Newton iteration scheme is well-defined for the problem.*

2. Worksheet

1. Like we saw last week for the near pendulum problem, some ODEs evolve in a way that conserves a specific quantity. For instance, the pendulum problem is of the form

$$\theta'' + \sin(\theta) = 0,$$

which it turns out will conserve the quantity

$$H(\theta, \theta') = \frac{1}{2}(\theta')^2 - \cos(\theta)$$

throughout the pendulum evolution. To see why, just differentiate H and use the pendulum equation! For more on Hamiltonians and conservation laws, see Chapter 1 of [**10**].

Solve the pendulum equation with $(\theta, \theta')(0) = (0, .2)$ on a time scale $[0, 10]$ with $1e4$ time steps using the RK algorithm described in the next paragraph. At each time step, compute $H(\theta, \theta')$ and plot it over the whole time interval. How well does your method do at preserving your conserved quantity??

A variation on the 4th order Runge-Kutta Algorithm (RK4) attempts to carefully make numerical integration very accurate between time steps by choosing a certain number of points in the interval to evaluate a function. However, those

points then need to be approximated by values stemming from the left end-point! The algorithm goes as follows

$$\begin{aligned}
k_1 &= f(t_n, y_n),\\
k_2 &= f(t_n + \frac{h}{3}, y_n + \frac{h}{2}k_1),\\
k_3 &= f(t_n + \frac{2h}{3}, y_n - \frac{h}{3}k_1 + hk_2),\\
k_4 &= f(t_n + h, y_n + hk_1 - hk_2 + hk_3),\\
y_{n+1} &= y_n + \frac{h}{8}(k_1 + 3k_2 + 3k_3 + k_4).
\end{aligned}$$

2. An implicit on the 4th order Runge-Kutta Algorithm (RK4) attempts to carefully make numerical integration very accurate between time steps by choosing a certain number of points in the interval to evaluate a function. However, those points then need to be approximated by values stemming from the left end-point! The algorithm goes as follows

$$\begin{aligned}
k_1 &= f\left(t_n + h\left(\frac{1}{2} - \frac{\sqrt{3}}{6}\right), y_n + k_1\frac{h}{4} + k_2 h\left(\frac{1}{4} - \frac{\sqrt{3}}{6}\right)\right),\\
k_2 &= f\left(t_n + h\left(\frac{1}{2} + \frac{\sqrt{3}}{6}\right), y_n + k_1 h\left(\frac{1}{4} + \frac{\sqrt{3}}{6}\right) + k_2\frac{h}{4}\right)\\
y_{n+1} &= y_n + \frac{h}{2}(k_1 + k_2).
\end{aligned}$$

Use this method to solve the damped pendulum problem

$$\theta'' + .1\theta' + \sin(\theta) = 0,$$

which now will no longer conserve $H(\theta, \theta')$, but actually decreases it

$$\partial_t(H(\theta, \theta')(t) = -.1(\theta')^2 \leqslant 0!!$$

Using the implicit RK scheme, solve the damped equation with $(\theta, \theta')(0) = (0, .2)$ on a time scale $[0, 10]$ with $1e4$ time steps using the RK algorithm described in the next paragraph. At each time step, compute $H(\theta, \theta')$ and plot it over the whole time interval. Do you see the expected behavior for H???

3. In general, a Runge-Kutta method looks like

$$\begin{aligned}
k_j &= f\left(t_n + c_j h, y_n + h\sum_{j=1}^{s} a_{ij}k_j\right),\\
y_{n+1} &= y_n + h\sum_{j=1}^{s} b_j k_j,
\end{aligned}$$

meaning that the RK methods (which CAN be accurate up to order s) can be defined by two s-column vectors $\vec{c}$ and $\vec{b}$ and an $s \times s$ matrix

$$A = [a_{ij}]_{i,j\in\{1,\ldots,s\}}.$$

Print out the $\vec{b}$, $\vec{c}$ and matrix A from Problems 1 and 2 above. To be *consistent* as a numerical scheme means that if we take $h \to 0$, the solutions would converge to the exact solution. For Runge-Kutta schemes, the condition is that

$$c_i = \sum_{j=1}^{s} a_{ij}.$$

To be *stable* as we have discussed before relating to behavior in approximating $y' = \lambda y$, we have the result difference equation

$$y_{n+1} = r(h\lambda)y_n$$

with

$$r(z) = \frac{\det\ (I_s - zA + z\vec{1}(\vec{b})^T)}{\det\ (I_s - zA)}$$

for $\vec{1}$ the $\mathbb{R}^{s\times 1}$ vector of all 1's. Note, since $b \in \mathbb{R}^{s\times 1}$, $b^T \in \mathbb{R}^{1\times s}$ and all entries of above are $s \times s$ matrices and the determinant can be computed. For $\vec{b}$, $\vec{c}$ and matrix A from Problems 1 and 2 above, verify the consistency condition and find $r(z)$.

4. (Optional) Things get even more fun if we have forcing and damping!! Solve the forced-damped pendulum equation using any solver you like

$$\theta'' + \eta\theta' + \sin(\theta) = \sin(\omega t),$$

with initial condition $(\theta, \theta')(0) = (0, 0)$ and a range of η spanning from 0 to 1 and ω spanning from 0 to 20. How does the behavior change from the case of small damping to large damping as the frequency of forcing goes from low to high?? Make some orbit plots of $(\theta, \theta')(t)$ in some illustrative cases.

CHAPTER 15

Week 14

1. Discussion Topics

Stable and Unstable Equilibrium Points, Linearization

2. Worksheet

1. Nonlinear problems can have equilibrium points where the solution cannot move. For instance, in the pendulum problem

$$\theta'' + \sin(\theta) = 0,$$

the initial data points $(\theta, \theta') = (0, 0)$ and $(\pi, 0)$ are fixed points since the right hand side of the matrix equation

$$\begin{pmatrix} \theta \\ \theta' \end{pmatrix}' = \begin{pmatrix} \theta' \\ -\sin\theta \end{pmatrix}$$

would be unable to move. These represent the two configurations where the pendulum is hanging down or standing straight up. Intuitively, you can lightly push a pendulum hanging down and it does not have to change a great deal as it will just oscillate back and forth. However, if a pendulum is standing straight up, a slight push will make it spin all the way around, causing a HUGE change.

Use any ODE solver of your choice (including imbedded Matlab solvers if you like to solve the pendulum equation with initial conditions $(\theta, \theta') = (.01, 0)$ and $(\pi + .01, 0)$ on a time scale of $[0, 6\pi]$ and plot θ vs. t for both solutions and θ' vs. θ for both solutions to see what is going on.

2. Nonlinear problems are generally hard to analyze, but we can look in spots near equilibrium points to see what the "linearized" behavior should be. For the pendulum problem above, near the initial value $\theta = 0$, we have $\sin(\theta) \approx \theta$. This approximation gives

$$\begin{aligned} \begin{pmatrix} \theta \\ \theta' \end{pmatrix}' &= \begin{pmatrix} \theta' \\ -\theta \end{pmatrix} \\ &= \begin{pmatrix} 0 & 1 \\ -1 & 0 \end{pmatrix} \begin{pmatrix} \theta \\ \theta' \end{pmatrix}. \end{aligned} \tag{2.1}$$

For the pendulum problem above, near the initial value $\theta = \pi$, we have $\sin(\pi + \theta) \approx -\theta$. This approximation gives

$$\begin{aligned} \begin{pmatrix} \theta \\ \theta' \end{pmatrix}' &= \begin{pmatrix} \theta' \\ \theta \end{pmatrix} \\ &= \begin{pmatrix} 0 & 1 \\ 1 & 0 \end{pmatrix} \begin{pmatrix} \theta \\ \theta' \end{pmatrix}. \end{aligned} \tag{2.2}$$

Using any *Matlab* ODE solver you like, compare the evolution of the linear equations derived above to the pendulum equation with the same initial data as in Problem 1. Namely, using $(\theta, \theta') = (.01, 0)$ for (2.1) and $(\pi + .01, 0)$ for (2.2) on a time scale of $[0, 6\pi]$, plot the solutions for the same initial data compared to the full pendulum evolution.

3. Using one more term in the Taylor expansion for sin, we end up with a nonlinear, but still simpler equation

$$\theta'' + \theta - \frac{\theta^3}{6} = 0,$$

Using any *Matlab* ODE solver you like, compare the evolution of (2.1), this cubic nonlinear equation and the full pendulum equation using $(\theta, \theta') = (.01, 0)$ on a time scale of $[0, 6\pi]$. Plot all the solutions on the same figure.

4. (Optional) Use the *eigs* command in Matlab to compute the eigenvalues of both matrices

$$\begin{pmatrix} 0 & 1 \\ -1 & 0 \end{pmatrix}, \quad \begin{pmatrix} 0 & 1 \\ 1 & 0 \end{pmatrix}.$$

The eigenvalues, λ, of a matrix play a key role in solving linear systems of ODEs through the function $e^{\lambda t}$. How does this function behave for the eigenvalues of each matrix above and how does that explain the "instability" observed above.

CHAPTER 16

Week 15

1. Discussion Topics

Bifurcations, Matrix Exponentials

2. Worksheet

1. A Hopf bifurcation appears in an ODE when changing a parameter makes a stable equilibrium point become unstable. An example of this is the Van Der Pol oscillator (which appears in electrical engineering, biology, seismology, etc.)

$$x' = \rho(1 - y^2)x - y,$$
$$y' = x,$$

which has equilibrium point $(x, y) = (0, 0)$. The linearization around $(0, 0)$ is

$$\begin{pmatrix} x \\ y \end{pmatrix}' = \begin{bmatrix} \rho & -1 \\ 1 & 0 \end{bmatrix} \begin{pmatrix} x \\ y \end{pmatrix}.$$

Using the *Matlab* function **eigs**, plot the real parts of the eigenvalues of the matrix

$$\begin{bmatrix} \rho & -1 \\ 1 & 0 \end{bmatrix}$$

for 25 values of ρ ranging from -1 to 1. Note, $(0, 0)$ is unstable if the linearized operator has an eigenvalue with positive real part, which leads to exponential growth. For two values of ρ, one on either side of your Hopf bifurcation, use **ode15s** to solve the Van der Pol oscillator problem with $(x, y)(0) = (.01, 0)$ on a time scale $[0, 10]$. Plot both solutions together on the same plot.

2. The equation

$$x' = x(\rho + x^2 - x^4)$$

has an interesting set of fixed points. For 100 values of ρ ranging from -2 to 5, using the *Matlab* function **roots**, find all possible equilibrium points and plot them versus ρ. This is the effect known as hysteresis. Choose an equilibrium point on each branch (note, for some values of ρ, the solution should have 6 total branches, for some 5, some with 3 and some 1) of the solution and solve using **ode45** to test its stability. As a result (trusting that the stability remains constant , re-plot your figure from above with red on stable branches and green solid on unstable.

3. (Optional) Period doubling bifurcations can also happen. The easiest example of this is in a discrete dynamical system

$$x_{n+1} = \rho x_n(1 - x_n),$$

which has fixed points $x = 0$ and $\frac{\rho-1}{\rho}$ as can be seen by finding the roots of

$$\rho x^2 + (1-\rho)x = 0.$$

For $|\rho| < 1$, 0 is stable, and, for $1 < \rho < 3$, $\frac{\rho-1}{\rho}$ is stable. When $\rho > 3$, interesting things happen. In particular, just above 3 (but less than roughly 3.45), there exist stable solutions that can bounce back and forth between two values. To see this, we can look for fixed points of the map under two iterations,

$$x_{n+2} = \rho^2 x_n(1-x_n)(1-\rho x_n(1-x_n)).$$

For $\rho = 3.1$, use **roots** or something like it to find the possible doubly periodic fixed points. For initial data $x_0 = .5$, plot the discrete evaluations up to $n = 100$. Does your solution lock on to the doubly periodic solution?? Note, as ρ increases, even more complex solutions can become stable and very complicated dynamics take over the system. See [**9**] for a more detailed discussion.

CHAPTER 17

Special Topic 1

1. Discussion Topics

Boundary Value Problems - Stokes Flow

2. Worksheet

1. A higher order differential equation that can be used in the modeling of the Stokes flow of a fluid of speed c flowing past a sphere of radius 1 is the 4th order equation

$$\left(\frac{d^2}{dr^2} - \frac{2}{r^2}\right)^2 f = 0,$$

with the boundary conditions $f(1) = f'(1) = 0$ and $f(r) \to \frac{cr^2}{2}$ as $r \to \infty$. Use the *matlab* function **bvp4c** to set up this equation with the given boundary conditions at $r = 1$ and approximate boundary conditions $f(r) = \frac{c25^2}{2}$ at $r = 25$ for $c = 1$ and $c = 5$ and save your solutions as two different vectors.

The actual fluid particles follow streamlines given by a function in polar coordinates

$$\psi(r, \theta) = \sin^2(\theta) f(r), \quad r \geqslant 1, \quad 0 \leqslant \theta < 2\pi.$$

Make a $3d$ plot of ψ as a function of r and θ for both your values of c. Use **contour** to make contour plots displaying what the level sets look like.

2. There is a second order ODE that arises in describing the structure of a vortex ring in a superfluid. It is

$$\frac{d^2 f}{d\rho^2} + \frac{1}{\rho}\frac{df}{d\rho} - \left(\frac{1}{\rho^2} - 1\right) f - s^3 = 0$$

with $f(0) = 0$ and $f(\rho) \to 1$ as $\rho \to \infty$. Using again **bvp4c**, plot the solution to this equation with approximate right boundary condition $f(100) = 1$. See [**5**].

3 (Optional) Use **bvp4c** to approximate the J_0 Bessel function solving

$$x^2 y'' + xy' + x^2 y = 0$$

with $y(0) = 1$ and $y \to 0$ as $x \to \infty$.

CHAPTER 18

Special Topic 2

1. Discussion Topics

Example from Environmental Science - Sedimentation and Gas Exchange

2. Worksheet

As developed with Professor Marc Alperin, we look at an ODE model for gas exchange in the Ocean floor.

Goal: Find the range of ocean water depths at which we might expect to find gassy sediments.

1.

Solve the first-order ODE to predict the sediment depth distributions of organic Carbon remineralization rate ($GRR_{SM} = k_G G$) as a function of ocean water depth over the range of $10 - 1000$ meters:

$$-\omega \frac{dG}{dx} - k_G G = 0, \tag{2.1}$$

where x is depth in the sediment (in units cm), G is the concentration of reactive organic Carbon ($grams\ C/grams_{SM} = gC/g_{SM}$) as a percentage of solid matter, ω is the sediment accumulation rate (in units $cm/year$) and k_G (in $1/year = 1/y$) is the first order rate constant for organic matter remineralization (decomposition). The sedimentation rate and rate constant can be parametrized as a function of the water depth column (z) as follows,

$$\omega\ (cm/y)\ = 3.3 \times 10^{-.875-4.35e-4z}, \quad k_G\ (1/y)\ = .057\omega^{1.94}.$$

The initial condition $G = G_0$ is given by

$$G_0 = \frac{F_G}{F_{sm}}$$

for F_G and F_{sm} the fluxes of reactive organic carbon and solid matter (sm), respectively, to the sediment-water interface,

$$F_G\ \left(mmolC/(cm^2_{WS}y)\right) = 1.8 \times 10^{-.509-3.89e-4z},$$

$$F_{sm}\ \left(g_{sm}/(cm^2_{WS}y)\right) = \omega\rho_{sm}(1-\varphi),$$

where

$$\rho_{sm} = 2.5\ \left(g_{sm}/cm^3_{sm}\right),\ \varphi = .8\left(cm^3_{pw}/cm^3_{ws}\right),$$

for ρ_{sm} the density of dry sediment, φ the sediment porosity with ws and pw representing "wet sediment" and "porewater" respectively.

Goal: Write a **for loop** solving (2.1) from $x = 0$ to a range of 10 depths $x = z$ spanning from $z = 10m$ to $z = 1000m$ and plot the solutions together on the same curve. Note the units carefully throughout!

2.

Using (2.1) from above, solve the following coupled, 2nd order ODE's to find the maximum concentration of methane as a function of ocean depth,

$$D_{ws}^{SO_4^{2-}} \frac{d^2[SO_4^{2-}]}{dx^2} - \omega \frac{d[SO_4^{2-}]}{dx} - \gamma_1 \left(\frac{GRR_{pw}}{2} + k_M[CH_4] \right) = 0,$$
$$D_{ws}^{CH_4} \frac{d^2[CH_4]}{dx^2} - \omega \frac{d[CH_4]}{dx} + \gamma_2 \left(\frac{GRR_{pw}}{2} - k_M[CH_4] \right) = 0,$$

where D_{pw} represents molecular diffusion coefficients in sediment pore water for sulfate or methane and $[SO_4^{2-}]$ and $[CH_4]$ denote sulfate or methane concentration respectively in the units of $(mmol/cm_{pw}^3)$. The diffusion coefficients in whole sediment are given by

$$D_{ws}^j = \varphi^2 D_{pw}^j, \; D_{pw}^{SO_4^{2-}} = 5.6 \times 10^{-6} \; (cm^2/s), \quad D_{ws}^{CH_4} = 9 \times 10^{-6} \; (cm^2/s).$$

The functions γ_1, γ_2 are the error functions that creates organic matter remineralization via sulfate reduction if $[SO_4^{2-}] > .2$ or methane production if $[SO_4^{2-}] < .2$ given by

$$\gamma_1 = \frac{1}{2} \left(1 + \text{erf} \left(\frac{[SO_4^{2-}] - .2}{.05} \right) \right),$$
$$\gamma_2 = \frac{1}{2} \left(1 - \text{erf} \left(\frac{[SO_4^{2-}] - .2}{.05} \right) \right).$$

Note, use the **erf** implicit *Matlab* function here.

Changing units to the porewater setting, we have

$$GRR_{pw} = \frac{GRR_{sm} \rho_{sm} (1 - \varphi)}{\varphi} \frac{10^9 nmolC}{12gC}.$$

Using that

$$[SO_4^{2-}] = 28 \; mM, \quad [CH_4] = 10^{-3} \text{ at } x = 0,$$

and

$$\frac{d[SO_4^{2-}]}{dx} = \frac{d[CH_4]}{dx} = 0 \text{ at } x = \frac{5\omega}{k_G},$$

solve the system using the boundary value solver **bvp4c** in *Matlab*. Note, the right boundary $x = \frac{5\omega}{k_G}$ is chosen such that $> 99\%$ of G is remineralized by that point.

Plot your 10 solutions for $[SO_4^{2-}]$ on the same plot. Note that your boundary conditions change with the water column depth, so set the **xlim** command to be on the scale of your widest simulation. Do the same for $[CH_4]$ as well.

3. Gas will form sediments if the methane concentration exceeds saturation, where

$$[CH_4]_{sat} = \frac{1000\beta(P + 1)}{RT},$$

with

$$\beta = .0389 \; (cm_{g,STP}^3/(atm \cdot cm_{pw}^3)), \quad RT = 22.414 \; (mmol/cm_{g,STP}^3),$$

and P the pressure where $1atm$ is equivalent to $10m$ of water depth. Here, β is the Bunsen solubility of methane in 4 degree C seawater. Display a list of your water column depths in which gassy sediment will occur.

CHAPTER 19

Special Topic 3

1. Discussion Topics

Example from Physics - Many Body Dynamics

2. Worksheet

Let us look at a simple gravitational model in $3d$ with interaction between 3 planets. The planets will have masses m_1, m_2, and m_3. Similarly, we will assume that they start at positions $\vec{x}_1 = (x_1, y_1, z_1)(0) = (x_{10}, 0, 0)$, $\vec{x}_2(0) = (x_{20}, 0, 0)$ and $\vec{x}_3(0) = (0, 0, 0)$ with initial velocity components $\vec{v}_1(0) = (u_1, v_1, w_1)(0) = (0, v_{10}, 0)$, $\vec{v}_2(0) = (0, v_{20}, 0)$ and $\vec{v}_3(0) = (0, 0, 0)$. The ODE system for $\vec{x}_1, \vec{v}_1$ is modeled by

$$
\begin{aligned}
x_1' &= u_1 \\
y_1' &= v_1 \\
z_1' &= w_1 \\
u_1' &= -Gm_2 \frac{x_1 - x_2}{|\vec{x}_1 - \vec{x}_2|^3} - Gm_3 \frac{x_1 - x_3}{|\vec{x}_1 - \vec{x}_3|^3} \\
v_1' &= -Gm_2 \frac{y_1 - y_2}{|\vec{x}_1 - \vec{x}_2|^3} - Gm_3 \frac{y_1 - y_3}{|\vec{x}_1 - \vec{x}_3|^3} \\
w_1' &= -Gm_2 \frac{z_1 - z_2}{|\vec{x}_1 - \vec{x}_2|^3} - Gm_3 \frac{z_1 - z_3}{|\vec{x}_1 - \vec{x}_3|^3}
\end{aligned}
$$

with similar equations for $\vec{x}_2, \vec{v}_2, \vec{x}_3, \vec{v}_3$ and G here is the Newtonian gravitational constant. Set up the system for an earth, sun, moon model by finding the necessary distances and initial velocities for a model orbit in consistent units, then solve it using the *ode15s* ODE solver and plot the $3d$ orbits for each planet on a time scale of several orbits. Why could we have stuck with a $2d$ model for this particular initial data?? Keep track of the quantity

$$
H_{grav} = \sum_{j=1}^{3} \frac{m_j |\vec{v}_j|^2}{2} + \sum_{1 \leqslant i < j \leqslant 3} \frac{m_i m_j}{|\vec{x}_j - \vec{x}_j|} = K + U,
$$

where K is the kinetic energy and U is the gravitational potential energy. Also, compare to the ODE equation for the inertia

$$
\frac{d^2 I}{dt^2} = 2K(t) - U(t),
$$

where

$$I(t) = \sum_{i=1}^{3} m_i |\vec{x}_i|^2.$$

Use **plot3** to track your orbits in $3d$ space!! Play around with other initial data too if you like!

This is the infamous 3-body problem and is TERRIBLY sensitive to initial conditions!

CHAPTER 20

Special Topic 4

1. Discussion Topics

Example from separation of scales - The Stommel Model from Climate Science, FitzHugh Nagumo

2. Worksheet

1. The model from [**6**] is of the form

$$\begin{aligned}\frac{dX}{dt} &= \frac{1}{\epsilon}(A - X^3 + 3X - K_0),\\ \frac{dA}{dt} &= K_1(X - X_0)^2 - K_2 - K_3 A - (K_4 + A) + H,\\ \frac{dH}{dt} &= K_5(K_4 + A - H),\end{aligned}$$

for X the volume of continental ice, A the amount of Carbon in the atmosphere and H the amount of Carbon in the mixed layer of the ocean and includes a number of parameters. For $\epsilon = .1, .01$ and $.001$ with $X_0 = .8$, $K_1 = 3$, $K_2 = 2.1$, $K_0 = 4$, $K_5 = 1$, $K_3 = 1$, $K_4 = 1$, solve with $(X, A, H)(0) = (0, 0, 0)$ on a time scale $T = [0, 10]$. Compare the three components for each value of ϵ on a set of 3 plots with a legend identifying the particular simulation.

2. A baby version of the FitzHugh-Nagumo equations (see [**4**]) related to models for neurons is:

$$\begin{aligned}\frac{dv}{dt} &= v(1 + v/\sqrt{3})(1 - v/\sqrt{3}) - w + .1,\\ \frac{dw}{dt} &= \epsilon(v - .01w).\end{aligned}$$

Solve this system using any implicit Matlab solver on a time interval $[0, 100]$ and $(v, w) = (0, 0)$ with $\epsilon = .001$, $.01$ and $.1$. Compare all three solutions for v at each ϵ value to the case where $\epsilon = 0$, $w = 0$ on a single plot with a legend identifying the particular simulation.

3. (Bonus) Separating scales can be an important part of analyzing coupled system of equations in boundary value problems as well. A model for this phenomenon comes from stationary states for the FitzHugh-Nagumo equations (see [**4**])

related to models for neurons:

$$\begin{aligned}\frac{du}{dx} &= v,\\ \frac{dv}{dx} &= -u(u-1/4)(1-u)+w,\\ \frac{dw}{dx} &= \epsilon(u-.01w).\end{aligned}$$

Solve this system using **bvp4c** on an interval $[-10,10]$ and (u,v) are near $(0,0)$ on the boundary when $\epsilon = 0$, $w = 0$. To get good initial data for u, v, you can use *pplane* for the system with $\epsilon = w = 0$. Then, do the same for the full system with $\epsilon = .001$, $.01$ and $.1$ and w vanishing at a boundary. Compare all three solutions for u at each ϵ value on a plot with a legend identifying the particular simulation.

APPENDIX A

Model Codes

0.1. Example 1 - ODE Function Files.

1. Example 2 - Newton's Method

```
%Main file to run newtode.m to solve the 3-body problem in
    classical
%celestial mechanics
%define initial conditions and paramter values for the Sun,
     Earth, and Moon
%using empirical values to simulate the 3-body system.

%sun initial conditions
%position components
y0(1)=0;
y0(2)=0;
y0(3)=0;
%velocity components
y0(4)=0;
y0(5)=0;
y0(6) =0;

%earth initial conditions
%position components
y0(7)=149600000*10^3;%400;
y0(8)=0;
y0(9)=0;
%velocity components
y0(10)=0;
y0(11)=2.9806e+04;
y0(12)=0;

%moon initial conditions
%position components
y0(13)=y0(7)+384400*10^3;
y0(14)=0;
y0(15)=0;
%velocity components
```

```
y0(16)=0;
y0(17)=1.023*10^3+y0(11);
y0(18)=0;

%Define time domain for solution
tspan=[0 1e9];
[T,Y]=ode15s(@newtode,tspan,y0);

close all
%plot trajectories of:
%sun (black)
%earth (blue)
%moon (red)
figure(1);
plot3(Y(:,1),Y(:,2),Y(:,3),'k-o');hold on
plot3(Y(:,7),Y(:,8),Y(:,9),'b');
plot3(Y(:,13),Y(:,14),Y(:,15),'r');

%plot 3D phase space
figure(2);
plot3(Y(:,13)-Y(:,7),Y(:,14)-Y(:,8),Y(:,15)-Y(:,9),'r');
```

```
function dydt=newtode(t,y)
%System of ODEs corresponding to Newton's Law of
    Gravitation using measured
%quantities of the Sun, Earth, and Moon to solve the 3-body
     problem from
%classical physics.

%universal gravitational constant
G=6.67384*10^(-11);
%masses in kilograms, multiplied by G
ms=G*1.989e30;
me=G*5.972e24;
mm=G*7.3477*1022;

%define distances
```

```
%distance from sun to earth
rse=[y(1),y(2),y(3)]-[y(7),y(8),y(9)];
%distance from sun to moon
rsm=[y(1),y(2),y(3)]-[y(13),y(14),y(15)];
%distance from earth to sun
res=-rse;
```

```
%distance from earth to moon
rem=[y(7),y(8),y(9)]-[y(13),y(14),y(15)];
%distance from moon to earth
rme=-rem;
%distance from moon to sun
rms=-rsm;

%cubed norms of distances
_____________________________________________________
nrse3=norm(rse,2)^3;
nrsm3=norm(rsm,2)^3;
nrem3=norm(rem,2)^3;

%equations of motion governing (coupled) dynamcis of the
    sun
dydt1=y(4);
dydt2=y(5);
dydt3=y(6);
dydt4=-me*rse(1)/nrse3-mm*rsm(1)/nrsm3;
dydt5=-me*rse(2)/nrse3-mm*rsm(2)/nrsm3;
dydt6=-me*rse(3)/nrse3-mm*rsm(3)/nrsm3;


%equations of motion governing (coupled) dynamcis of the
    earth
dydt7=y(10);
dydt8=y(11);
dydt9=y(12);
dydt10=-ms*res(1)/nrse3-mm*rem(1)/nrem3;
dydt11=-ms*res(2)/nrse3-mm*rem(2)/nrem3;
dydt12=-ms*res(3)/nrse3-mm*rem(3)/nrem3;

%equations of motion governing (coupled) dynamcis of the
    moon
dydt13=y(16);
dydt14=y(17);
dydt15=y(18);
dydt16=-ms*rms(1)/nrsm3-me*rme(1)/nrem3;
dydt17=-ms*rms(2)/nrsm3-me*rme(2)/nrem3;
dydt18=-ms*rms(3)/nrsm3-me*rme(3)/nrem3;

dydt=[dydt1 dydt2 dydt3 dydt4 dydt5 dydt6 dydt7 dydt8 dydt9
    dydt10 dydt11 dydt12 dydt13 dydt14 dydt15 dydt16 dydt17
    dydt18];
dydt=dydt';
```

2. Example 3 - The Jacobian Matrix

3. Example 4 - Boundary Value Solvers

```
function vfout=LotkaBVP(a,c,n)
% Purpose: compute the solution to the boundary value
    problem (BVP) consisting of
% the Lotka-Volterra system of ODES:
% x'=x(alpha-beta*y)
% y'=-y*(gamma+delta*x)
% with boundary conditions
% x(0)=a
% x(1)=c
% over a domain [0,1] discretized with n grid points

%Example:
% solve the above system with initial prey population a=1,
    and final prey
% population c=4, using a time domain over [0,1]
    discretized by 100 points.
% LotkaBVP(1,4,100)

%% Initialize Parameters, initial conditions as global
    variables
%define global parameters for initial and final populations
     of prey,
%model parameters alpha, beta, gamma, and delta
global u0 uf alpha beta gamma delta
u0=a;
uf=c;

%Fixed parameter values for Lotka-Volterra ODE system
alpha=1.5;
beta=1.0;
gamma=3.0;
delta=1.0;

%% Solution by bvp4c (see functions below)
%solution grid (independent variable)
x_init = linspace ( 0.0, 1.0, n);
%guess for solution to ODE
y_init = [a;c];
%initialize BVP solution
solinit = bvpinit ( x_init, y_init );
%compute solution using bvp4c or bvp5c
sol = bvp5c ( @twoode, @twobc, solinit );
```

```
38
39 %% Plot Solution
40 %plot the solution for each component
41 figure(1);
42 plot (sol.x, sol.y(1,:), 'r-', 'Linewidth', 2 );hold on
43 plot (sol.x, sol.y(2,:), 'b-', 'Linewidth', 2 );
44 title('Population over Time of Competing Species, solved as
       Boundary Value Problem')
45 xlabel('Time (Years)')
46 ylabel('Species Population')
47 legend('Species (Prey) Pop.','Species 2 (Predator) Pop.')
48 hold off
49
50 %plot one trajectory in the phase plane
51 figure(2);
52 plot (sol.y(1,:), sol.y(2,:), 'b-', 'Linewidth', 2 );hold
      on
53 title('Populations of Competing Species over Time, solved
      as Boundary Value Problem')
54 xlabel('Prey Population')
55 ylabel('Predator Population')
56
57 %Return the solution
58 vfout=[sol.x sol.y(1) sol.y(2)];
59
60
61 %% Definition of ODE and Boundary Conditions
62
63 %system of differential equations (Lotka-Volterra Predator-
      Prey model) to solve
64 function dydt = twoode(t,y)
65 global u0 uf alpha beta gamma delta
66 dydt = [y(1).*(1.5 - 1.*y(2)); -y(2).*(3-1.*y(1))];
67
68 %boundary conditions, here:
69 %     ya is vector of values at left boundary
70 %     yb is vector of values at right boundary
71 %     so, for example, ya(1) is the prey (species/y(1)
      population) at the initial time
72 %     yb(1) is the prey (species/y(1) population) at the
      final time
73 function res = twobc(ya,yb)
74 global u0 uf
75 res = [ ya(1) - u0
76         yb(1) - uf];
```

Bibliography

[1] *MIT Open Courseware* (2008).
[2] *Differential Equations and Boundary Value Problems.* Boyce and DiPrima (2013).
[3] *Josephson tunneling in a Hamiltonian ODE from Optics.* Goodman, Marzuola and Weinstein (2014).
[4] *Dynamical Systems.* L. Arnold, C.K.R.T. Jones, K. Mischaikow and G. Raugel (1994).
[5] *Vortex ring web-site.* Kevrekidis (2014).
[6] *Mixed mode oscillations in a conceptual climate model.* Roberts, Widiasih, Wechselberger, Jones (2014).
[7] *Nonlineary dynamical systems and chaos.* Steven Strogatz (1994).
[8] *Discussion Topics notes on numerical ODE* John Strain (2004).
[9] *Computational Methods in Engineering* Gilbert Strang (2010).
[10] *Ordinary Differential Equations* Michael E. Taylor (2013).