Updated on December 7, 2021

Math 3607: Exam 4

Due: 11:59PM, Friday, December 10, 2021

Please read the statements below and sign your name.

Disclaimers and Instructions

- You are **not** allowed to use MATLAB commands and functions **other than** the ones discussed in lectures, accompanying live scripts, textbooks, and homework/practice problem solutions.
- You may be requested to explain your code to me, in which case a proper and satisfactory explanation must be provided to receive any credits on relevant parts.
- You are **not** allowed to search online forums or even MathWorks website for this exam.
- You are **not** allowed to collaborate with classmates, unlike for homework.
- If any code is found to be plagiarized from the internet or another person, you will receive a zero on the *entire* exam and will be reported to the COAM.
- Do not carry out computations using *Symbolic Math Toolbox*. Any work done using `sym`, `syms`, `vpa`, and such will receive NO credit.
- **Notation.** Problems marked with are to be done by hand; those marked with are to be solved using a computer.
- Answers to analytical questions (ones marked with) without supporting work or justification will not receive any credit.

Academic Integrity Statements

- All of the work shown on this exam is my own.
- I will not consult with any resources (MathWorks website, online searches, etc.) other than the textbooks, lecture notes, and supplementary resources provided on the course Carmen pages.
- I will not discuss any part of this exam with anyone, online or offline.
- I understand that academic misconduct during an exam at The Ohio State University is very serious and can result in my failing this class or worse.
- I understand that any suspicious activity on my part will be automatically reported to the OSU Committee on Academic Misconduct (COAM) for their review.

Signature __________________________

1 Optimal Step Size [25 points]

In lecture, the optimal h for the second-order centered difference formula was shown to be about $\boxed{\text{eps}}^{1/3}$. At this optimal h, the leading error is $O(\boxed{\text{eps}}^{2/3})$.

(a) Determine the optimal h for the first-order forward difference formula by following a similar argument. Also determine the leading error at this optimal h.

(b) The following program approximates the Jacobian of $\mathbf{f} : \mathbb{R}^n \to \mathbb{R}^m$ at $\mathbf{x}_0$ using the first-order forward difference formula with the optimal step size determined in the previous part.

```
function J = jacfd(f, x0)
% JACFD Calculates Jacobian using 1st-order forward difference (FD)
% Input:
%   f       function to be differentiated
%           which takes (n-by-1) column vector as an input
%           and produces (m-by-1) column vector as an output
%   x0      evaluation point (n-by-1)
% Output:
%   J       approximate Jacobian (m-by-n)

    h = [BLANK 1];              % optimal step size
    y0 = f(x0);                 % value of f at x0 (m-by-1)
    m = length(y0);             % see specification above
    n = length(x0);             % see specification above
    J = [BLANK 2];              % pre-allocation
    I = eye(n);
    for j = 1:n                 % iterate over columns of J
        J(:,j) = [BLANK 3]; % FD formula for j-th column of J
    end

end
```

Fill in the blanks. No justification is needed, and there is no partial credit.

- BLANK 1: ______________________
- BLANK 2: ______________________
- BLANK 3: ______________________

(c) (Optional; no bonus)

(i) Generalize the argument in part (a) to determine the optimal h for an mth-order forward difference formula, where m is any positive integer. Also determine the leading error at this optimal h.

(ii) Suppose one wants to find the points on the ellipsoid $x^2/25 + y^2/16 + z^2/9 = 1$ that are closest to and farthest from the point $(5, 4, 3)$. The method of Lagrange multipliers implies that any such point satisfies the following nonlinear system

$$\begin{cases} x - 5 = \dfrac{\lambda x}{25}, \\ y - 4 = \dfrac{\lambda y}{16}, \\ z - 3 = \dfrac{\lambda z}{9}, \\ 1 = \dfrac{1}{25}x^2 + \dfrac{1}{16}y^2 + \dfrac{1}{9}z^2 \end{cases}$$

for an unknown value of λ. Solve this system using the multidimensional Newton's method where the Jacobian matrix is *numerically* calculated using `jacfd` from part (b).

Hint for part (b). Recall that

$$\mathbf{J}(\mathbf{x}) = \begin{bmatrix} \dfrac{\partial f_1}{\partial x_1} & \dfrac{\partial f_1}{\partial x_2} & \cdots & \dfrac{\partial f_1}{\partial x_n} \\ \dfrac{\partial f_2}{\partial x_1} & \dfrac{\partial f_2}{\partial x_2} & \cdots & \dfrac{\partial f_2}{\partial x_n} \\ \vdots & \vdots & \ddots & \vdots \\ \dfrac{\partial f_m}{\partial x_1} & \dfrac{\partial f_m}{\partial x_2} & \cdots & \dfrac{\partial f_m}{\partial x_n} \end{bmatrix} \in \mathbb{R}^{m \times n}$$

The jth column of $\mathbf{J}$ consists of all partial derivatives with respect to x_j:

$$\mathbf{J}(\mathbf{x})\mathbf{e}_j = \begin{bmatrix} \dfrac{\partial f_1}{\partial x_j} \\ \dfrac{\partial f_2}{\partial x_j} \\ \vdots \\ \dfrac{\partial f_m}{\partial x_j} \end{bmatrix}$$

This column vector can be approximated by a finite difference formula involving a perturbation only in x_j:

$$\mathbf{J}(\mathbf{x})\mathbf{e}_j \approx \frac{\mathbf{f}(\mathbf{x} + h\mathbf{e}_j) - \mathbf{f}(\mathbf{x})}{h}, \quad j = 1, \ldots, n,$$

where h is optimally chosen according to part (a).

2 Airplane Velocity from Radar Readings [35 points]

The radar stations A and B, separated by the distance $a = 500$ m, track a plane C by recording the angles α and β at one-second intervals.

Successive readings for α and β at integer times $t = 7, 8, \ldots, 14$ are stored in the file `plane.dat`. The columns of the data file are the observation time t, the angle α (in degrees), and the angle β (also in degrees), in that order. At each time t, the Cartesian coordinates of the plane can be calculated from the angles α and β as follows:

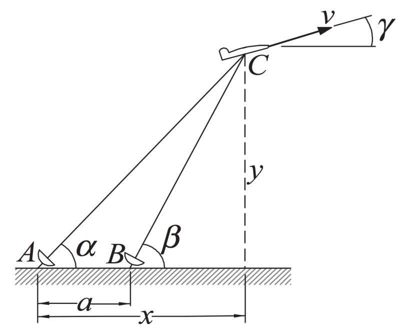

Figure 1: An airplane and two radar stations.

$$x(\alpha, \beta) = a\frac{\tan(\beta)}{\tan(\beta) - \tan(\alpha)} \quad \text{and} \quad y(\alpha, \beta) = a\frac{\tan(\beta)\tan(\alpha)}{\tan(\beta) - \tan(\alpha)}. \tag{1}$$

(**Note.** Be sure to distinguish a and α in the above formulae.)

Denote the position vector of the plane at time t by $\mathbf{s}(t) = (x(t), y(t))^{\mathrm{T}}$. Then the speed at time t, which is the magnitude (or the 2-norm) of the velocity vector, is given by

$$(\text{speed}) = \|\mathbf{s}'(t)\| = \sqrt{x'(t)^2 + y'(t)^2}, \tag{2}$$

and the total distance traveled from $t = a$ to $t = b$ is obtained by integrating the speed, that is,

$$(\text{distance}) = \int_a^b \|\mathbf{s}'(t)\|\ dt = \int_a^b \sqrt{x'(t)^2 + y'(t)^2}\, dt. \tag{3}$$

Your goal, back at the air traffic control, is to determine the position and the speed of the airplane at finer time steps (0.01-second intervals) and to estimate the total distance traveled.

(a) Load the data, convert α and β to radians[1], and then compute the coordinates $x(t)$ and $y(t)$ at each given $t = 7, 8, \ldots, 14$ using (1). Store them as `xdp` and `ydp`. (Do NOT print these out.)

Note.
- To load the data, type `load plane.dat`. To see how the data file is arranged, type `type plane.dat`.
- For consistency of notation and for later use, store the time data $t = 7, 8, \ldots, 14$ as `tdp`.

(b) Let $t_1, t_2, \ldots, t_{701}$ be evenly-spaced points on $[7, 14]$, that is,

$$t_j = 7 + \frac{j-1}{100}, \quad \text{for } j = 1, \ldots, 701.$$

Interpolate `xdp` and `ydp` using (not-a-knot) cubic splines to obtain the coordinates $x(t_j)$ and $y(t_j)$, for $j = 1, \ldots, 701$. Store them as `x` and `y`. (Do NOT print these out.) Then plot

[1] You may ignore this step and use `tand`.

the trajectory of the airplane using x and y, with the positions obtained from radar readings circled in a well-labeled graph; see the figure below. Determine the maximum height and the corresponding time.

Notes.
- Note that t_j's are spaced out by $\Delta t = 0.01$ second. Use either `linspace` or the colon operator to construct them and store it as a vector `t`, for consistency of notation.
- For cubic spline interpolation, you may use either `interp1` or `spline`.
- This is a 2-D spline in which $x = x(t)$ and $y = y(t)$ are independently interpolated with respect to their common parameter, the time t. Do NOT use the (pseudo-)arclength parameter as in homework.

(c) Approximate $x'(t_j)$ and $y'(t_j)$, for $j = 1, 2, \ldots, 701$, using **second-order** finite difference methods. In particular, use the second-order forward difference for $t = t_1$, the second-order backward difference for $t = t_{701}$, and the second-order centered difference for $t = t_2, \ldots, t_{700}$. Then calculate the speed at each t_j using (2). Vectorize your code as much as possible. Store the speed as `spd`. (Do NOT print out $x'(t_j)$, $y'(t_j)$, nor the speed.) Determine the maximum speed and the corresponding time.

(d) Find the total distance traveled by the airplane from $t = 7$ to $t = 14$ using (3). Use the composite Simpson's rule to numerically calculate the integral.

(e) (Optional; bonus) Verify the equations in (1). Comment on the validity of the formulae as either α or β tends to $90°$ (airplane straight above a radar station).

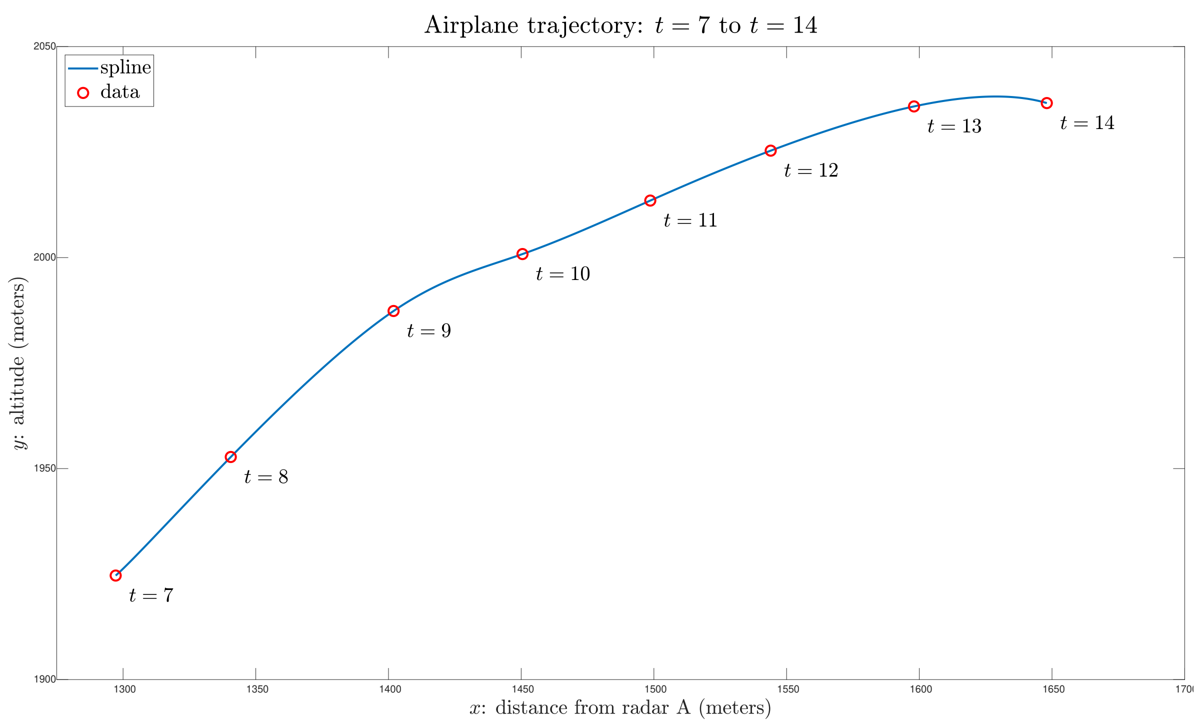

Figure 2: Example output for part (b).

3 Go Away, COVID! [15 points]

Spread of diseases or virus such as seasonal flus or COVID-19 can be modeled by so-called *compartmental models* which partition a given population into different compartments like *susceptible*, *exposed*, *infected*, *recovered*, and the like, and keep track of the transition of population between compartments. One of the simplest such models is the **SIR model** presented by the following ODEs:

$$\begin{cases} \dfrac{dS}{dt} = -\beta S I \\ \dfrac{dI}{dt} = \beta S I - \gamma I \\ \dfrac{dR}{dt} = \gamma I \end{cases} \qquad \text{where} \qquad \begin{cases} S: & \text{proportion of susceptible population} \\ I: & \text{proportion of infected popoulation} \\ R: & \text{proportion of recovered population} \\ \beta: & \text{transmission rate} \\ \gamma: & \text{recovery rate.} \end{cases}$$

The compartmentalization implies that

$$S + I + R = 1. \tag{†}$$

(a) Write a MATLAB function (include it at the end of your live script) that solves the SIR model using `ode45` with the following specifications:

- The function takes as inputs the transmission rate β, the recovery rate γ, the time span $[0, T]$, and the initial compartmentalization profile $(S(0), I(0), R(0))^{\mathrm{T}}$.
- The function must return (numerical) solutions $(S(t), I(t), R(t))^{\mathrm{T}}$ at discrete time steps between $t = 0$ and $t = T$.
- The numerical solution is to be obtained using `ode45` with relative error tolerance of 10^{-6}.
- The function must check that the initial conditions provided by a user satisfy the equation (†). If invalid initial conditions are detected, it must abort the program and send an error message.

(b) Using the function written in part (a), solve the SIR system with $\beta = 0.8$, $\gamma = 0.05$ for $0 \leqslant t \leqslant 100$, with initial values $(S(0), I(0), R(0)) = (0.9, 0.09, 0.01)$.

(c) Using the solutions obtained in part (b), predict the time $\tau \in [0, 100]$ after which the infected population is less than 1% of the total population, that is, $I(t) \leqslant 0.01$ for all $t \geqslant \tau$.

(d) Plot the solutions obtained in part (b) on a well-labeled graph; see below for an example.

(e) (Optional; bonus) Numerically confirm that the solutions obtained in part (b) satisfy $S(t) + I(t) + R(t) = 1$ ("proportions must add up to 1") at all time t using a single MATLAB statement. Your one line of code must output a single number by which one can determine whether or not the property is satisfied at all time. Explain your logic behind the code.

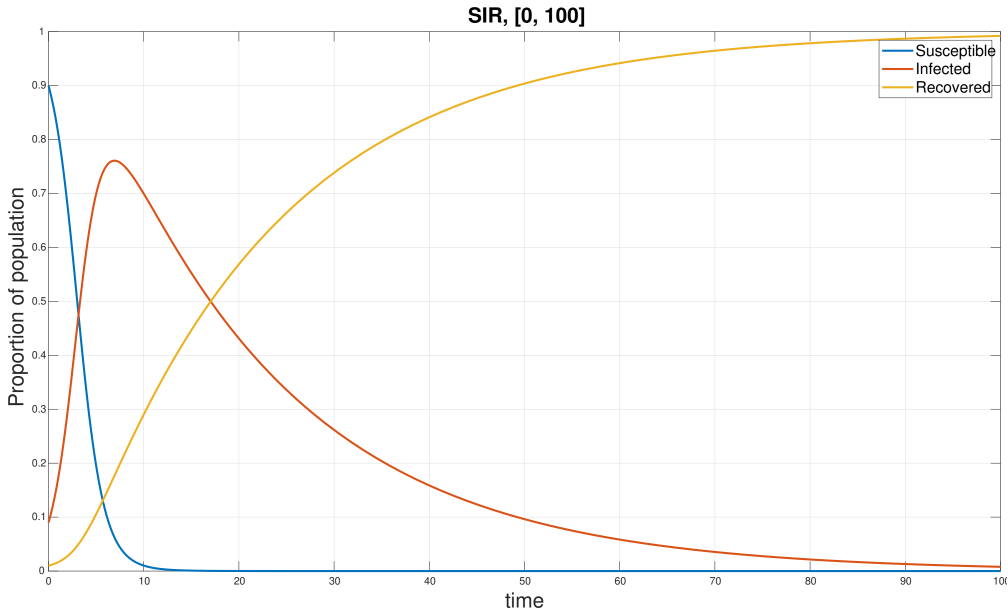

Figure 3: Example output for part (d).