




Spring 2022 Math 3607: Final Exam

Due: 11:59PM, Friday, April 29, 2022

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, supplementary resources provided on the course Carmen pages, or MATLAB's built-in help documentation.
- 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 _____

Notation. Problems marked with  are to be done by hand; those marked with  are to be solved using a computer.

1 Newton's method faster than quadratic? [20 points]

Suppose that f has a simple root at r at which f'' vanishes, that is,

$$f(r) = 0, \quad f'(r) \neq 0, \quad f''(r) = 0.$$

Assuming that f is (at least) three times continuously differentiable near r , determine the rate of convergence of Newton's method. That is, express the relationship between ϵ_{k+1} and ϵ_k in the form

$$\epsilon_{k+1} = C\epsilon_k^p + O(\epsilon_k^{p+1}),$$

where C and p are to be determined. Recall that $\epsilon_k = x_k - r$.

2 Optimal Step Size for Finite Differences

[20 points]

Suppose that f is differentiable at x and $f'(x)$ is to be approximated using the first-order forward difference formula $D_h^{[1f]}\{f\}(x)$ on a computer. Determine the optimal step size h and the corresponding leading error.

3 Embedding Watermark inside Image

[25 points]

Let A and W be m -by- n matrices representing two grayscale images of the same pixel dimensions as shown below. The image W is *embedded* inside the image A using an SVD-based scheme described below.



Figure 1: Two original images. The image represented by A on the left and the image represented by W on the right.

Let $A = U\Sigma V^T$ be an SVD of A . For some small scaling factor $\alpha > 0$, construct a new matrix $\Sigma + \alpha W$ and consider its SVD

$$\Sigma + \alpha W = U_w \Sigma_w V_w^T. \quad (1)$$

Note that both Σ and Σ_w are diagonal matrices and, for a suitably chosen α , $\Sigma \approx \Sigma_w$. Thus, A_w obtained by

$$A_w = U \Sigma_w V^T \quad (2)$$

is approximately equal to A and so represents an image which looks nearly identical to that of A , while containing information about the image W as well.

Your mission, should you choose to accept it, is to establish an algorithm to retrieve (an approximation of) W embedded in A_w given U_w , V_w , Σ , and α .

Begin by loading the data with

```
>> load watermark % download and save 'watermark.mat' first
```

This loads A_w , U_w , V_w , S , and α , which correspond to A_w , U_w , V_w , Σ , and α as described above, respectively. Note that A_w is a matrix of double-precision floating point numbers whose elements range from 0 to 255.

- (a) Write a MATLAB code to find a matrix W_0 , an *approximation* of W , out of A_w .
- (b) Display the images represented by A_w and W_0 side by side using `subplot`. Do they look similar to their respective original images in Figure 2?






4 Air Resistance, Rootfinding, and Lambert W

[25 points]

The function

$$h(t) = -30t + 780(1 - e^{-t/3})$$

models the height of a rocket in the air at time t , subject to air resistance; see the figure below. Let t_{\max} be the time at which the rocket reaches its maximum height and let t_{ground} be the time at which the rocket hits the ground.

-  Find t_{\max} analytically. That is, find an exact expression for t_{\max} . Justify all your steps.
-  Find t_{\max} numerically, using `fzero`. Then compare the result against the analytical answer obtained in part (a).
-  (Optional/Bonus) Find t_{ground} analytically. That is, find an exact expression for t_{ground} . Justify all your steps.
-  Find t_{ground} numerically, using `fzero`. Then compare the result against the analytical answer obtained in part (c), only if you did it.
-  Produce a well-labeled plot of $h(x)$ as shown in the figure. Clearly mark the maximum height and the time the rocket hits the ground.

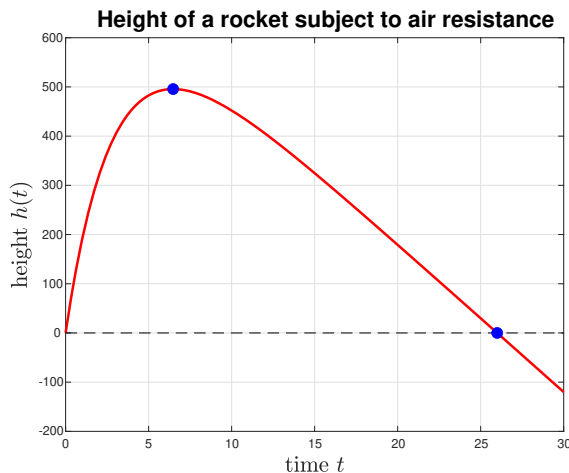


Figure 2: Example output.

Hints for (a) and (c). To find t_{\max} which maximizes the height $h(t)$, use just what you know from Calculus. To find t_{ground} analytically, you need to solve an equation of the form

$$At + e^{Bt} = 1 \tag{3}$$

for t . To do this, introduce a new variable u by the transformation

$$t = -\frac{1}{B}u + \frac{1}{A}. \tag{4}$$

Substitute this into (3) and solve for u , using Lambert W function. Then use the transformation (4) to solve for t .

5 Airplane Velocity from Radar Readings

[30 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 $t = 7, 8, \dots, 14$ 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:

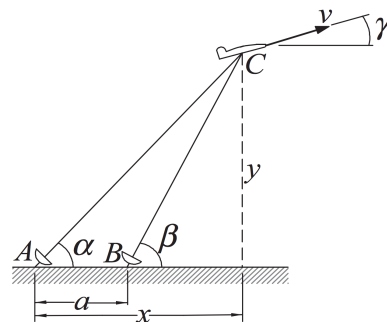


Figure 3: An airplane and two radar stations.

$$x(\alpha, \beta) = a \frac{\tan(\beta)}{\tan(\beta) - \tan(\alpha)} \quad \text{and} \quad y(\alpha, \beta) = a \frac{\tan(\beta) \tan(\alpha)}{\tan(\beta) - \tan(\alpha)}. \quad (5)$$

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

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

$$(\text{speed}) = \|\mathbf{s}'(t)\| = \sqrt{x'(t)^2 + y'(t)^2}. \quad (6)$$

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).

- (a) Load the data, convert α and β to radians¹, and then compute the coordinates $x(t)$ and $y(t)$ at each given $t = 7, 8, \dots, 14$ using (5). 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 $t = 7, 8, \dots, 14$ as `tdp`.

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

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

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

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

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 .

- (c) Approximate $x'(t_j)$ and $y'(t_j)$, for $j = 1, 2, \dots, 701$, 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 $t = t_2, \dots, t_{700}$. Then calculate the speed at each t_j using (6). 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.

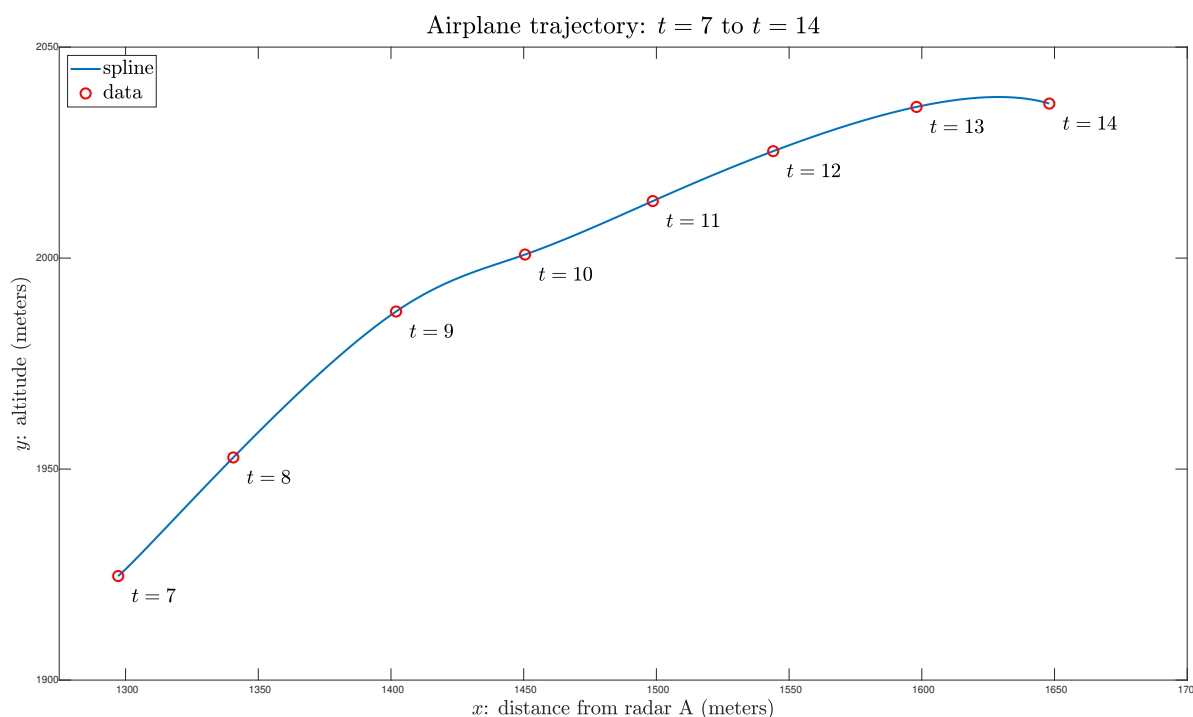


Figure 4: Example output for part (b).