Please write your family and given names and **underline** your family name on the front page of your paper.

All answers **must** be typed (preferably in latex), and compiled to a single pdf. All code, output and plots of Q1, Q4 should be *embedded* in latex/pdf. Code and output should be embedded with `fixed-width fonts`, e.g. Courier. Font size of all fonts must be 12. Linespacing set to 1.1 or close. Do **not** use dark background anywhere.
What to submit:
(1) The single pdf file 00A3.pdf (with embedded code, output, plots, spy).
(2) Any source code you wrote (for computation, etc).
(3) Any plot file embedded in the latex/pdf.
Thus, the code and plots will be available within latex/pdf, *as well as* separately.
See course website for example of latex, embedding plots, code, using fixed width fonts, etc.
Some points will be given for the quality of presentation.

1. This question involves the study of the outbreak of an epidemic in a population and its eradication with vaccinations. The study of spreading of infectious diseases is mainly done by numerical Ordinary Differential Equations (ODEs), which are not the topic of CSC336 (but of CSC436). However, numerical ODE solvers, essentially convert a system of ODEs to a sequence of nonlinear systems. Each nonlinear system of the sequence relates unknown quantities at a time point (or step), in terms of known quantities at the previous time point, while the initial state is given (known). In this way, the values of various quantities at several time points are computed and studied.

Assume there is a population of N people, which remains constant over time. Assume that each person belongs to one of the four groups: susceptible, infected/ious, recovered/immune and vaccinated/immune. The infected/ious persons transmit the virus at a given rate β to the susceptible and make them infected, the infected/ious recover at a given rate γ, and stay immune for an average period of $1/omega$, then become susceptible again. Furthermore, the susceptible get vaccinated at a given rate ρ, and become immune immediately after vaccination and stay immune for an average period of $1/omega_v$, then become susceptible again. The population of N persons is replenished at a given death and birth rate μ, and all newborns are susceptible.

In an instance of this problem, at each time point i, the following 4×4 nonlinear system needs to be solved:

$$x_1^{(i)} = x_1^{(i-1)} + h\mu N - h\mu x_1^{(i)} - h\,\frac{\beta}{N}\,x_1^{(i)}x_2^{(i)} - h\rho\,x_1^{(i)} + h\omega\,x_3^{(i)} + h\omega_v x_4^{(i)} \tag{1a}$$

$$x_2^{(i)} = x_2^{(i-1)} - h\mu x_2^{(i)} + h\,\frac{\beta}{N}\,x_1^{(i)}x_2^{(i)} - h\gamma\,x_2^{(i)} \tag{1b}$$

$$x_3^{(i)} = x_3^{(i-1)} - h\mu x_3^{(i)} + h\gamma\,x_2^{(i)} - h\omega\,x_3^{(i)} \tag{1c}$$

$$x_4^{(i)} = x_4^{(i-1)} - h\mu x_4^{(i)} + h\rho\,x_1^{(i)} - h\omega_v x_4^{(i)} \tag{1d}$$

In the above, $\bar{x}^{(i)} = [x_1^{(i)}, x_2^{(i)}, x_3^{(i)}, x_4^{(i)}]^T$ represents the vector of numbers of susceptible, infected/ious, recovered/resistant/immune and vaccinated/immune persons, respectively, at time point i, and these are the four unknowns that need to be computed at the ith time point (step), assuming the quantities of the previous time point $i-1$ are already computed, and the initial state, $\bar{x}^{(0)} = [x_1^{(0)}, x_2^{(0)}, x_3^{(0)}, x_4^{(0)}]^T$ is given. While in practice $x_j^{(i)}$ are integers, we treat them as continuous variables. Also, h is a small stepsize in time, that, in this instance, is given and fixed. Also note that, although some simplifications can be applied to this system, we just treat it as a general 4×4 nonlinear system. Finally, note that it is not required that you understand the details of how this nonlinear system arises. You take the system as granted.

(a) [*5 points*] Write the system (1a)-(1d) in standard form $\bar{f}(\bar{x}^{(i)}) = \bar{0}$. Indicate all components of $\bar{f}$. Write (by hand, i.e. in latex/other) the Jacobian matrix for this system. Indicate all entries (some in terms of $x_j^{(i)}$).

(b) [*8 points*] Consider Newton's method for this system. One iteration of Newton's includes computation of the form
solve $J(\bar{x}^{(i,k-1)})\bar{s}^{(i,k-1)} = -\bar{f}(\bar{x}^{(i,k-1)})$
compute $\bar{x}^{(i,k)} = \bar{x}^{(i,k-1)} + \bar{s}^{(i,k-1)}$
Note that i is the time point index, while k is the index for Newton's iteration, the former remaining constant through all Newton iterations (all k).
Write a matlab or equivalent script that, given $N = 15e6$, $\beta = 0.75$, $\gamma = 0.06$, $\mu = 0.01/365$, $\rho = 150\mu$, $\omega = 2/365$, $\omega_v = 1.5/365$, $h = 1$, $x_1^{(0)} = N - x_2^{(0)} - x_3^{(0)} - x_4^{(0)}$, $x_2^{(0)} = 3800$, $x_3^{(0)} = 589533$, $x_4^{(0)} = 0.8N$, computes $\bar{x}^{(1)}$ by Newton's method with tolerance 10^{-7}. Use `maxit = 10`, and as stopping criterion the infinity norm of the *residual* vector. The Jacobian must be solved using *backslash*. At each Newton iteration, output $\bar{x}^{(1,k)}$ (the four values computed), the sum of the four values and the infinity norm of the residual. See script `testnlv.m` (in the course website) for some template. Comment on the number of Newton iterations and on how the residual behaves as the iterations proceed.

(c) [*10 points*] In (c), you will do four simulations, each with a different value of ρ, to study the effect of vaccination rate to the evolution of the epidemic.
Consider a simulation time period from 0 to $t_{end} = 4 \times 365 + 1$ (can be viewed as a period of 4 years), divided in $nstep = \frac{t_{end}}{h}$ periods of stepsize (length) $h = 1/24$ (or dt), and that, at each time point i, given the state at time point $i-1$, a nonlinear system of the form (1a)-(1d) is solved by Newton's method with tolerance 10^{-7}. The initial guess for Newton's at each time point is the computed solution of the previous time point. Write a script that, given the same parameters as in (b), except that $\beta = 0.25$ (*note the change!*), and $\rho = 150\mu$ (and $h = 1/24$), computes $\bar{x}^{(i)}$, for $i = 1, \cdots, nstep$. Do NOT output and do NOT save the results of each Newton iteration. Save, but do NOT output, the number of Newton iterations at each timestep. Save, but do NOT output, the results ($\bar{x}^{(i)}$) from each time point. It is best to save them in a matrix, say `yi`, of size $nstep \times 4$. They will be used later for plotting.
Output the initial values $\bar{x}^{(0)}$ and their sum, the computed final $\bar{x}^{(nstep)}$ (four values at time t_{end}), and their sum, as well as the maximum number of each type of persons among all time points (steps). Output also which *day* the maximum number of infected persons occurs.
Plot $x_1^{(i)}$, $x_2^{(i)}$, $x_3^{(i)}$ and $x_4^{(i)}$ versus ih, $i = 0, \cdots, nstep$ (index of time point scaled by h), in one plot, using dotted (black), dashed (red), dotted-dashed (blue), solid (green), for susceptible, infected, recovered and vaccinated, respectively. Add proper legend, labels for axes, etc. Use `axis tight;` *after* the plot.
Do the above four times, one with $\rho = 150\mu$ (as above), and then with $\rho = 300\mu$, $\rho = 450\mu$, $\rho = 600\mu$. Thus, you will produce four plots, which you will present in a 2×2 layout (using latex, or other).
Present a table of the final and maximum number of infected and respective values of ρ. (The numbers in the table are excerpts from the output.) Comment on the results.
In another plot, plot the number of Newton's iteration at each timestep, versus the index of the timestep, for each of the four simulations above, in one plot. Add labels for axes. Use `axis([1 nstep 0.9 2.1]);` or other convenient scale *after* the plot. This is a fifth plot. If you saved the number of iterations from all four simulations in a matrix of size $4 \times nstep$, you can do the plot easily. Comment on the results.

(d) [*7 points*] In (d), you will do five simulations, each with a different value of ω_v, to study the effect of immunity period the vaccines offer to the evolution of the epidemic.
All parameters are set as in (b), except $\rho = 300\mu$ (for all (d) simulations), and $\omega_v = 0.5/365$, $\omega_v = 0.75/365$, $\omega_v = 1/365$, $\omega_v = 1.25/365$ and $\omega_v = 1.5/365$. Output the same quantities as in (c). You do not have to do the plots.
Present a table of the final and maximum number of infected and respective values of ω_v. (The numbers in the table are excerpts from the output.) Comment on the results.

Notes: Do not use any symbolic calculation of any sort. The derivatives should be derived by hand and hard-coded into the Jacobian matrix.

2. [To be done by hand/latex/other, no coding.] Consider computing the smallest root of the function $f(x) \equiv xe^{-\frac{x}{2}} - \frac{e^{-1}}{2}$, defined in $\mathbb{R}$. (Note that $e = \exp(1) = 2.7183\ldots$ and $e^{-1} = \exp(-1) = \frac{1}{e} = 0.3679\ldots$.)

(a) [*5 points*] Using mathematical arguments, show that there exists exactly one root of $f(x)$ in $[0, 2]$.
Using mathematical arguments, show that there exists exactly one root of $f(x)$ in $(-\infty, 6]$ (i.e. there are no roots in $(-\infty, 0)$ and no roots in $(2, 6]$). You can use the fact that $f(6) > 0$.
Let r denote the root of $f(x)$ in $(-\infty, 6]$.

(b) [*4 points*] Using $x^{(0)} = 0$ as initial guess, apply (by hand/latex/other) one Newton iteration to compute an approximation $x^{(1)}$ to the root. Indicating how $x^{(1)}$ is computed, simplify as much as you can, and write the result in terms of e^{-1}.

(c) [*4 points*] The equation $f(x) = 0$ can be equivalently written as $x = \frac{e^{-1}}{2} e^{\frac{x}{2}}$, which gives rise to the fixed-point iteration scheme $x^{(k+1)} = g(x^{(k)})$, with $g(x) = \frac{e^{-1}}{2} e^{\frac{x}{2}}$.
Using $x^{(0)} = 0$ as initial guess, apply (by hand/latex/other) one fixed-point iteration to compute an approximation $x^{(1)}$ to the root. Indicating how $x^{(1)}$ is computed, simplify as much as you can, and write the result in terms of e^{-1}.

(d) [*8 points*] Show that the fixed-point iteration scheme converges to r, if started anywhere in $[0, 2]$.

(e) [*6 points*] What is the convergence rate (order) of Newton's if started close enough to r? Explain.
What is the convergence rate (order) of the fixed-point iteration in (c)-(d)? Explain.

(f) [*8 points*] Find the largest interval you can, so that the fixed-point iteration scheme converges, if started within the interval, and so that you justify your answer mathematically. (Note: the interval may expand $[0, 2]$ from both sides.)

3.

(a) [*5 points*] Using the Newton's Divided Differences (NDD) table, construct the polynomial $p_2(x)$ of degree at most 2, that interpolates the function $f(x) = e^x$, at the points $x_0 = -1$, $x_1 = 0$, $x_2 = 1$. Bring the polynomial into monomial form, if it is not already in that form. The coefficients of the polynomial are to be given either in terms of e and e^{-1}, or as (approximate) numerical values.

(b) [*3 points*] Give the error formula for this interpolation problem (i.e. for *this* f and *these* data points). The formula should involve an unknown point ξ. Any other functions involved in the formula should be given explicitly in terms of x.

(c) [*12 points*] Using the polynomial interpolation error formula, give an upper bound for each of $|e^{1.5} - p_2(1.5)|$, $\max\limits_{-1 \le x \le 1} |e^x - p_2(x)|$, $\max\limits_{-1 \le x \le 2} |e^x - p_2(x)|$, $\max\limits_{-2 \le x \le 1} |e^x - p_2(x)|$. Explain how you got the bounds. The bounds are to be given as (approximate) numerical values.
Note: You must use the polynomial interpolation error formula to find the bounds.

4. [*15 points*] Consider the set of data $\{(x_i, y_i), i = 0, \cdots, 20\}$ given by the following piece of MATLAB code:

```
x = [ 0.9  1.3  1.9  2.1  2.6  3.0  3.9  4.4  4.7  5.0  6.0  7.0  8.0  9.2 ...
     10.5 11.3 11.6 12.0 12.6 13.0 13.3];
y = [ 1.3  1.5  1.85 2.1  2.6  2.7  2.4  2.15 2.05 2.1  2.25 2.3  2.25 1.95 ...
      1.4  0.9  0.7  0.6  0.5  0.4  0.25];
```

Consider also the set of equidistant evaluation points $\{(xi)_i, i = 1, \cdots, 100\}$ in the interval [0.87, 13.33]. Write a MATLAB program that

- constructs the not-a-knot cubic spline interpolant of the above data and evaluates it on the evaluation points
- constructs the 20th degree polynomial interpolant of the above data and evaluates it on the evaluation points
- plots the above two interpolants in one graph based on their respective values on the evaluation points.

Comment on what you observe, including any warning obtained by MATLAB.

Notes:

- This question is inspired by the relevant discussion in page 158 of the Burden and Faires textbook (depicting the shape of a duck).
- You can find an electronic version of the above piece of MATLAB code in `http://www.cs.toronto.edu/~ccc/Courses/336/duck_given.m`. (Hence, no excuses for typographical errors...)
- Use `yvs = interp1(x, y, xv, 'spline');` (or `yvs = spline(x, y, xv);`) to get the values of the cubic spline interpolant on the evaluation points. This interpolant uses a type of not-a-knot condition.
  Use the built-in MATLAB functions `polyfit` and `polyval` to get the values of the 20th degree polynomial interpolant on the evaluation points. Note that although the MATLAB built-in function `polyfit` is designed to construct least-squares polynomial approximations, it can be used to construct polynomial interpolants as well, if the number of data and the degree requested are set appropriately.
  If you get a warning (Warning: Polynomial is badly conditioned...") from `polyfit`, you are not necessarily doing anything wrong.
- Use `linspace(0.87, 13.33, 100)` to get the 100 equidistant evaluation points.
- Set the axes scales using `axis([0 14 -6 5])` to view the plot appropriately. (Use `axis` *after* `plot`.)
- Use solid line for the spline interpolant and dashed for the polynomial one. For each data point, write an `o` on the graph.