CSC 336 - Fall 2021 - Assignment 3

Songheng Yin, 1004762303

Question 1.

(a)

The standard form is

$$(h\mu + h\rho + 1)x_1^{(i)} + h\frac{\beta}{N}x_1^{(i)}x_2^{(i)} - h\omega x_3^{(i)} - h\omega_v x_4^{(i)} - x_1^{(i-1)} - h\mu N = 0 \tag{1}$$

$$(h\mu + h\gamma + 1)x_2^{(i)} - h\frac{\beta}{N}x_1^{(i)}x_2^{(i)} - x_2^{(i-1)} = 0 \tag{2}$$

$$(h\mu + h\omega + 1)x_3^{(i)} - h\gamma x_2^{(i)} - x_3^{(i-1)} = 0 \tag{3}$$

$$(h\mu + h\omega_v + 1)x_4^{(i)} - h\rho x_1^{(i)} - x_4^{(i-1)} = 0 \tag{4}$$

The system is $\bar{f}(x_1^{(i)}, x_2^{(i)}, x_3^{(i)}, x_4^{(i)}) = \bar{0}$, i.e. 4 components.

$$J = \begin{pmatrix} h\mu + h\rho + 1 + h\frac{\beta}{N}x_2^{(i)} & h\frac{\beta}{N}x_1^{(i)} & -h\omega & -h\omega_v \\ -h\frac{\beta}{N}x_2^{(i)} & h\mu + h\gamma + 1 - h\frac{\beta}{N}x_1^{(i)} & 0 & 0 \\ 0 & -h\gamma & h\mu + h\omega + 1 & 0 \\ -h\rho & 0 & 0 & h\mu + h\omega_v + 1 \end{pmatrix}$$

(b)

MATLAB script:

```
% computing the spreading of influenza
% test for Newton's on one (large) timestep

% beta transmission, gamma recovery, mu death/birth (replenishment)
beta = 0.75; gamma = 0.06; mu = 0.01/365;
% rho vaccination, 1/omega, 1/omegaV immunity from recovering, vaccination
rho = 150 * mu; omega = 2/365; omegaV = 1.50/365;

% initial conditions
N = 15e6;
h = 1;
y02 = 3800; y03 = 589533; y04 = 0.80*N;
y01 = N - y02 - y03 - y04;
```

```
y0 = [y01 y02 y03 y04]';

dt = 1; % stepsize for time is h (or dt)
Beta = dt*beta/N; Gamma = dt*gamma; Mu = dt*mu; % for convenience
Rho = dt*rho; Omega = dt*omega; OmegaV = dt*omegaV;

maxit = 10; tol = 1e-7; % Newton's parameters
fprintf(' k   S         I       R        V        Total     Residual\n');
y = y0;
yinit = y; % initial guess for Newton's
for k = 1:maxit
    % define vector f and its inf norm
%     f = [
%         y(1) - yinit(1) - h * mu * N + h * mu * y(1) + h * beta * y(1) * y(2) / N + h * rho
%         y(2) - yinit(2) + h * mu * y(2) - h * beta * y(1) * y(2) / N + h * gamma * y(2);
%         y(3) - yinit(3) + h * mu * y(3) - h * gamma * y(2) + h * omega * y(3);
%         y(4) - yinit(4) + h * mu * y(4) - h * rho * y(1) + h * omegaV * y(4);
%     ];
    f = [
        y(1) - yinit(1) - Mu * N + Mu * y(1) + Beta * y(1) * y(2) + Rho * y(1) - Omega * y(3)
        y(2) - yinit(2) + Mu * y(2) - Beta * y(1) * y(2) + Gamma * y(2);
        y(3) - yinit(3) + Mu * y(3) - Gamma * y(2) + Omega * y(3);
        y(4) - yinit(4) + Mu * y(4) - Rho * y(1) + OmegaV * y(4);
    ];
    fnorm = norm(f, inf);
    fprintf('%2d %9.0f %6.0f %9.0f %6.0f %10.0f %9.2e\n', ...
        k-1, y, sum(y), fnorm);
    % stopping criterion
    if fnorm <= tol
        break;
    end
    % define Jacobian matrix
    J = [
        Mu + Rho + 1 + Beta * y(2), Beta * y(1), -Omega, -OmegaV;
        -Beta * y(2), Mu + Gamma + 1 - (Beta * y(1)), 0, 0;
        0, -Gamma, Mu + Omega + 1, 0;
        -Rho, 0, 0, Mu + OmegaV + 1
    ];
    s = J \ f;
%         fprintf('s is %f %f %f %f', s);
    % apply Newton's iteration to compute new y
    y = y - s;
end
```

Table 1: (b) Results

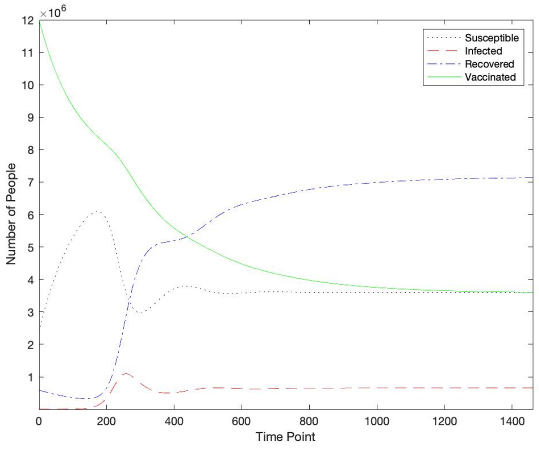
k	S	I	R	V	Total	Residual
0	2406667	3800	589533	12000000	15000000	4.25e+04
1	2448819	4052	586546	11960583	15000000	5.32e-01
2	2448818	4053	586546	11960583	15000000	1.70e-08

Comments: The infinity norm of the residual keeps decreasing during our experiment iterations.

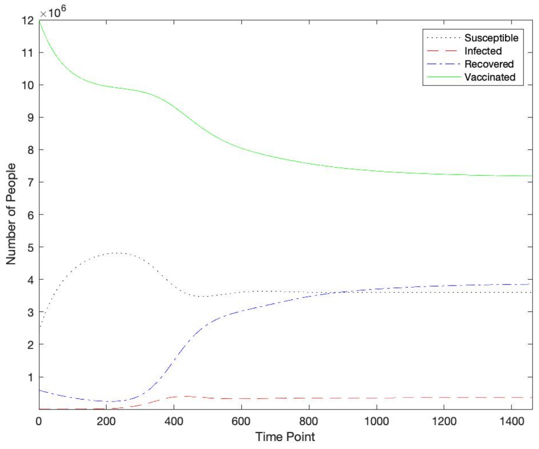
(c)

MATLAB script:

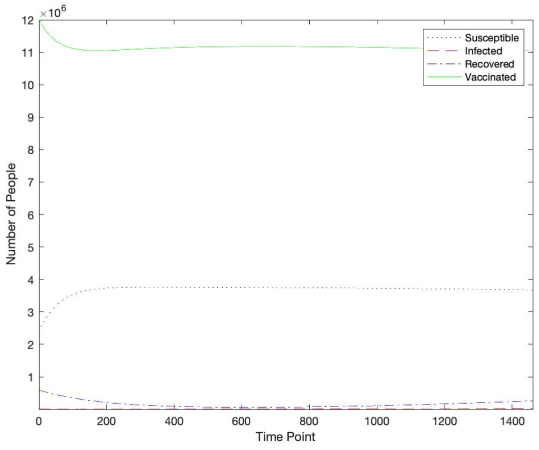
```
% beta transmission, gamma recovery, mu death/birth (replenishment)
beta = 0.25; gamma = 0.06; mu = 0.01/365;
%1/omega, 1/omegaV immunity from recovering, vaccination
omega = 2/365; omegaV = 1.50/365;
% initial conditions
N = 15e6;
h = 1 / 24;
y02 = 3800; y03 = 589533; y04 = 0.80*N;
y01 = N - y02 - y03 - y04;
y0 = [y01 y02 y03 y04]';

dt = 1/24; % stepsize for time is h (or dt)
Beta = dt*beta/N; Gamma = dt*gamma; Mu = dt*mu; % for convenience
Omega = dt*omega; OmegaV = dt*omegaV;

t_end = 4 * 365 + 1;
nstep = t_end / h;

% difference
num_Newton = zeros(4, nstep); % store the number of Newton's iteration
for rho_index = 1:4
    rho = rho_index * 150 * mu;
    Rho = dt * rho;
    fprintf('rho = %d * mu\n', rho_index * 150);
    % result matrix
    yi = zeros(nstep + 1, 4);   % row i <---> time at i-1
    yi(1, :) = y0;
    fprintf(' k      S         I       R        V        Total\n');
    fprintf('%d %9.0f %6.0f %9.0f %6.0f %10.0f   Initial\n', rho_index * 150, y0, sum(y0));
    for i = 1:nstep
        maxit = 10; tol = 1e-7; % Newton's parameters

        y = yi(i, :);  % the initial guess is the solution of the previous time
        for k = 1:maxit
            % define vector f and its inf norm
               f = [
                y(1) - yi(i, 1) - Mu * N + Mu * y(1) + Beta * y(1) * y(2) + Rho * y(1) - Omega
                y(2) - yi(i, 2) + Mu * y(2) - Beta * y(1) * y(2) + Gamma * y(2);
                y(3) - yi(i, 3) + Mu * y(3) - Gamma * y(2) + Omega * y(3);
                y(4) - yi(i, 4) + Mu * y(4) - Rho * y(1) + OmegaV * y(4);
            ];
            fnorm = norm(f, Inf);
            % stopping criterion
            if fnorm <= tol
                break;
            end
            % define Jacobian matrix
            J = [
                Mu + Rho + 1 + Beta * y(2), Beta * y(1), -Omega, -OmegaV;
                -Beta * y(2), Mu + Gamma + 1 - (Beta * y(1)), 0, 0;
                0, -Gamma, Mu + Omega + 1, 0;
                -Rho, 0, 0, Mu + OmegaV + 1
            ];
            s = J \ (-f);
            % apply Newton's iteration to compute new y
            y = y + s.';
        end
        yi(i + 1, :) = y;
        num_Newton(rho_index, i) = k - 1;
    end
    % done for one rho, output
    fprintf('%d %9.0f %6.0f %9.0f %6.0f %10.0f   Final\n', rho_index * 150, yi(nstep + 1, :),

    fprintf('%d %9.0f %6.0f %9.0f %6.0f    MAX\n', rho_index * 150, max(yi(1:nstep + 1, 1)), m
    [argvalue, argmax] = max(yi(:, 2));
    fprintf('The max # infected occurs on day %d\n', (argmax - 1)*h);
    fprintf('\n');
    % plot
    X = ((1:nstep + 1) - 1) * h;
    figure();
    plot(X, yi(:, 1), ':k', ...
        X, yi(:, 2), '--r', ...
        X, yi(:, 3), '-.b', ...
        X, yi(:, 4), '-g');
    legend('Susceptible', 'Infected', 'Recovered', 'Vaccinated');
    xlabel('Time Point');
    ylabel('Number of People');
    axis tight;
end
% plot the 5th figure
figure();
plot((1:nstep), num_Newton(1, :), ':k', ...
     (1:nstep), num_Newton(2, :), '--r', ...
     (1:nstep), num_Newton(3, :), '-.b',...
     (1:nstep), num_Newton(4, :), '-g');
legend('150', '300', '450', '600');
xlabel('Time Point');
ylabel('Number of Newton Iteration');
axis([1 nstep 0.9 2.1]);
```



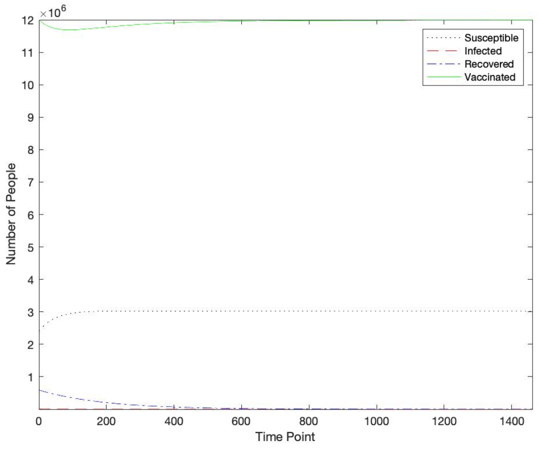
(a) $\rho = 150\mu$



(b) $\rho = 300\mu$



(c) $\rho = 450\mu$



(d) $\rho = 600\mu$

Figure 1: x value plotsx

ρ	Object	S	I	R	V	Total
150μ	Initial	2406667	3800	589533	12000000	15000000
	Final	3601878	656826	7137068	3604228	15000000
	Maximum	6086550	1094926	7137068	12000000	-
300μ	Initial	2406667	3800	589533	12000000	15000000
	Final	3602145	355941	3857097	7184816	15000000
	Maximum	4813525	394995	3857097	12000000	-
450μ	Initial	2406667	3800	589533	12000000	15000000
	Final	3677212	30243	258355	11034189	15000000
	Maximum	3766726	30243	589533	12000000	-
600μ	Initial	2406667	3800	589533	12000000	15000000
	Final	3015997	0	201	11983802	15000000
	Maximum	3025157	3800	589533	12000000	-

The maximum number of infected person occurs on day $257, 429, 1461, 0$ respectively for $\rho = 150\mu, 300\mu, 450\mu, 600\mu$

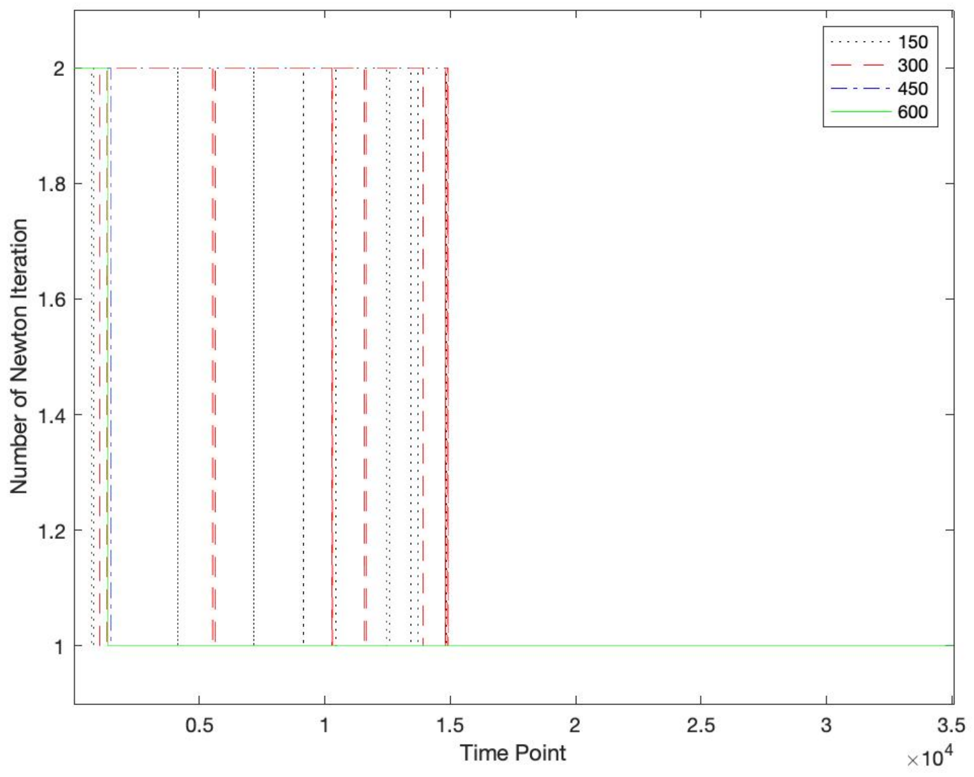


Figure 2: Number of Newton's Iteration

Comment:

The larger the susceptible get vaccinated rate is, the more vaccinated people will be and the fewer infection will occur. The number of susceptible people will also decrease as the ρ increases, but the speed will be much slower than the drop of the number of infection. The number of the recovered people will be smaller as well due to fewer infection exist.
Based on the day number where the maximum infected number shows up, we know we need the vaccinated rate to be larger than some threshold (in this case, somewhere between 450μ and 600μ) to make the plague under control (i.e. the maximum number appear at day 0 or near).
In our 4 different ρ values, the larger the value is, the earlier the number of Newton's iteration converges to 1. Also the larger ρ is, the less oscillation of the number of Newton's iteration has.

(d)

MATLAB script:

```
% beta transmission, gamma recovery, mu death/birth (replenishment)
beta = 0.75; gamma = 0.06; mu = 0.01/365;
%1/omega, 1/omegaV immunity from recovering, vaccination
rho = 300 * mu; omega = 2/365;
% initial conditions
N = 15e6;
h = 1;
dt = 1; % stepsize for time is h (or dt)
y02 = 3800; y03 = 589533; y04 = 0.80*N;
y01 = N - y02 - y03 - y04;
y0 = [y01 y02 y03 y04]';
Beta = dt*beta/N; Gamma = dt*gamma; Mu = dt*mu; % for convenience
Omega = dt*omega; Rho = dt * rho;
t_end = 4 * 365 + 1;
nstep = t_end / h;
% difference
for experiment_index = 1:5
    omegaV = 0.25 * (experiment_index + 1) / 365;
    OmegaV = dt*omegaV;
    fprintf('rho = %.2f /365 \n', 0.25 * (experiment_index + 1));
    % result matrix
    yi = zeros(nstep + 1, 4);   % row i <---> time at i-1
    yi(1, :) = y0;
    fprintf('365omegaV  S      I       R        V        Total\n');
    fprintf('%.2f %9.0f %6.0f %9.0f %6.0f %10.0f   Initial\n', 0.25 * (experiment_index + 1),
    for i = 1:nstep
        maxit = 10; tol = 1e-7; % Newton's parameters
        y = yi(i, :);  % the initial guess is the solution of the previous time
        for k = 1:maxit
            % define vector f and its inf norm
               f = [
                y(1) - yi(i, 1) - Mu * N + Mu * y(1) + Beta * y(1) * y(2) + Rho * y(1) - Omega
                y(2) - yi(i, 2) + Mu * y(2) - Beta * y(1) * y(2) + Gamma * y(2);
                y(3) - yi(i, 3) + Mu * y(3) - Gamma * y(2) + Omega * y(3);
                y(4) - yi(i, 4) + Mu * y(4) - Rho * y(1) + OmegaV * y(4);
            ];
            fnorm = norm(f, Inf);
            % stopping criterion
            if fnorm <= tol
                break;
            end
            % define Jacobian matrix
            J = [
                Mu + Rho + 1 + Beta * y(2), Beta * y(1), -Omega, -OmegaV;
                -Beta * y(2), Mu + Gamma + 1 - (Beta * y(1)), 0, 0;
                0, -Gamma, Mu + Omega + 1, 0;
                -Rho, 0, 0, Mu + OmegaV + 1
            ];
            s = J \ (-f);
            % apply Newton's iteration to compute new y
            y = y + s.';
        end
        yi(i + 1, :) = y;
    end
    % done for one rho, output
    fprintf('%.2f %9.0f %6.0f %9.0f %6.0f %10.0f   Final\n', 0.25 * (experiment_index + 1), yi

    fprintf('%.2f %9.0f %6.0f %9.0f %6.0f    MAX\n', 0.25 * (experiment_index + 1), max(yi(1:n
    [argvalue, argmax] = max(yi(:, 2));
    fprintf('The max # infected occurs on day %d\n', (argmax - 1)*h);
    fprintf('\n');
end
```

ω_v	Object	S	I	R	V	Total
$0.5/365$	Final	1203187	516637	5449638	7830538	15000000
	Maximum	2406667	516637	5449638	12146155	-
$0.75/365$	Final	1201764	739803	7918436	5139997	15000000
	Maximum	2585903	739803	7918436	12000000	-
$1/365$	Final	1201030	852790	9210562	3735619	15000000
	Maximum	2817789	987398	9210562	12000000	-
$1.25/365$	Final	1200717	917504	9956063	2925717	15000000
	Maximum	3036954	1217115	9956063	12000000	-
$1.5/365$	Final	1200598	958957	10429355	2411090	15000000
	Maximum	3239673	1429541	10429355	12000000	-

The maximum number of infected person occurs on day $1461, 1461, 90, 83, 78$ respectively for $\omega_v = 0.5/365, 0.75/365, 1/365, 1.25/365, 1.5/365$ respectively.

Comment: With a shorter immunity lasting period, both the final and maximum infected number will increase. More people will become susceptible and infected (so that more people will get recovered also). Because fewer people are vaccinated in this case.

Question 2.

(a)

$$\begin{aligned} f(0) &= -\frac{e^{-1}}{2} < 0 \\ f(2) &= 2e^{-1} - \frac{e^{-1}}{2} = \frac{3}{2}e^{-1} > 0 \end{aligned}$$

Also since $f(x)$ is continuous in $[0, 2]$, by Bolzano theorem, there exist at least one root in the interval.

$$\begin{aligned} f'(x) &= e^{-\frac{x}{2}} - \frac{1}{2}xe^{-\frac{x}{2}} \\ &= e^{-\frac{x}{2}}(1 - \frac{x}{2}) \\ &> 0 && \text{, for } x \in (0, 2) \end{aligned}$$

Clearly f is differentiable because it is elementary function, by theorem, the root must be unique in $(0, 2)$. It is also the unique root in $[0, 2]$.

For $x \in (-\infty, 0)$, $f(x) = xe^{-\frac{x}{2}} - \frac{e^{-1}}{2} < 0$. No root exists here.
For $x \in (2, 6]$, $f'(x) = e^{-\frac{x}{2}}(1 - \frac{x}{2}) < 0$, which implies f is decreasing on the interval to $f(6) > 0$. Obviously $f(x) > 0$ for any $x \in (2, 6]$.
Hence no roots exist in $x \in (-\infty, 0)$ and $x \in (2, 6]$, i.e. there exists exactly one root in $(-\infty, 6]$.

(b)

$$\begin{aligned} f(x^{(0)}) &= f(0) = -\frac{e^{-1}}{2} \\ f'(x^{(0)}) &= f'(0) = 1 \end{aligned}$$

By Newton's iteration,

$$\begin{aligned} x^{(1)} &= x^{(0)} - \frac{f(x^{(0)})}{f'(x^{(0)})} \\ &= 0 + \frac{e^{-1}}{2} = \frac{e^{-1}}{2} \end{aligned}$$

(c)

By fixed-point iteration scheme,

$$\begin{aligned} x^{(1)} = g(x^{(0)}) &= \frac{e^{-1}}{2}e^{0} \\ &= \frac{e^{-1}}{2} \end{aligned}$$

(d)

The fixed point of g, which is also the root of f, $r \in [0, 2]$.

$$\begin{aligned} g'(x) &= \frac{1}{4}e^{\frac{x}{2}-1} && \text{which is increasing} \\ g'(0) \le g'(x) &\le g'(2) && \text{where, } x \in [0,2] \\ 0 < \frac{1}{4}e^{-1} \le g'(x) &\le \frac{1}{4} \\ |g'(x)| &< \frac{1}{4} \end{aligned}$$

Hence g is contractive in $[0, 2]$ with $\lambda = \frac{1}{4}$.
Also, $g'(x) > 0$ implies g is increasing on $[0, 2]$, so for $x \in [0, 2]$,

$$\begin{aligned} g(0) \le g(x) &\le g(2) \\ \frac{e^{-1}}{2} \le g(x) &\le \frac{1}{2} \\ 0 \le g(x) &\le 2 \end{aligned}$$

that is, $g(I) \subseteq I$ for $I = [0, 2]$.
By theorem 4b, $\forall x^{(0)} \in [0, 2]$, the iteration scheme $x^{(k+1)} = g(x^{(k)})$ converges to r.

(e)

By we calculated in (b), define

$$\begin{aligned} \gamma(x) &= x - f/f' \\ &= \frac{x - \frac{e^{-1}}{2}e^{\frac{x}{2}}}{1 - \frac{x}{2}} \end{aligned}$$

because r is simple and γ is at least twice differentiable near r with non-zero second-order derivative, the rate of convergence is 2 for the Newton's method.

Since $g'(x) = \frac{1}{4}e^{\frac{x}{2}-1}$,

$$g'(r) = \frac{1}{4}e^{\frac{r}{2}-1} > 0 \qquad \text{showed in (d)}$$

By theorem 6, the rate of convergence of $x^{(k+1)} = g(x^{(k)})$ is 1

(f)

First solve the inequality

$$\begin{aligned} |g'(x)| = \frac{1}{4e}e^{\frac{x}{2}} &< 1 \\ e^{\frac{x}{2}} &< 4e \\ x &< 2\ln 4e \end{aligned}$$

Hence g is contractive in $(-\infty, 2\ln 4e)$. (1)
By $g'(x) > 0$ we know g is increasing, for $x \in (-\infty, 2\ln 4e)$,

$$\begin{aligned} g(-\infty) \leq g(x) &< g(2\ln 4e) \\ 0 \leq g(x) &< \frac{1}{2e}e^{\ln 4e} = 2 \\ 0 \leq g(x) &< 2\ln 4e \end{aligned}$$

Thus g maps $(-\infty, 2\ln 4e)$ to iteself. (2)
By theorem, (1) and (2) imply any initial guess from $(-\infty, 2\ln 4e)$ must make the fixed-point iteration converge.
To determine if there exists any point outside the interval and after a few iterations get into the interval. Define

$$h(x) = g(x) - x$$

We want know the domain such that $h(x) < 0$ so that function g maps x to a smaller value.

$$h'(x) = 0$$

has one root $2\ln 4e$ which means h is increasing on $(2\ln 4e, \infty)$.
Use a numercial solver we get the root to $h(x) = 0$ is approximate 7.38.
By experiment, 7.38 will turn into much less than 1 after 6 iterations.
Therefore, the interval where the fixed-point iteration converges is $(-\infty, 7.38)$

Question 3.

(a)

$f_0 = f[x_0] = e^{-1}, f_1 = f[x_1] = 1, f_2 = f[x_2] = e$
NDD table:

$$\begin{array}{ccccc} -1 & e^{-1} & & & \\ & & 1-e^{-1} & & \\ 0 & 1 & & \frac{e+e^{-1}-2}{2} & \\ & & e-1 & & \\ 1 & e & & & \end{array}$$

Thus,

$$\begin{aligned} p_2(x) &= e^{-1} + (1-e^{-1})(x+1) + \frac{e+e^{-1}-2}{2}(x+1)x \\ &= \frac{e+e^{-1}-2}{2}x^2 + \frac{e-e^{-1}}{2}x + 1 \end{aligned}$$

(b)

Since $f(x) = e^x \in C^\infty$, by the error in polynomial interpolation theorem, the error is

$$\begin{aligned} f(x) - p_2(x) &= \frac{f^{(3)}(\xi)}{3!}\prod_{j=0}^{2}(x-x_j) \\ &= \frac{e^\xi}{6}x(x+1)(x-1) \end{aligned}$$

where $\xi \in ospr\{x, x_0, x_1, x_2\}$

(c)

I.
By (b), Let $x = 1.5$,
In this case, $ospr\{x, x_0, x_1, x_2\} = ospr\{1.5, -1, 0, 1\} = (-1, 1.5)$, thus $-1 < \xi < 1.5$

$$\begin{aligned} |e^{1.5} - p_2(1.5)| &= |\frac{e^\xi}{6}1.5 \times 2.5 \times 0.5| = |\frac{5}{16}\xi| \\ &< \frac{5}{16} \times 1.5 = 0.469 \end{aligned}$$

II.
Let $-1 \le x \le 1$, $ospr\{x, x_0, x_1, x_2\} = ospr\{x, -1, 0, 1\} = (-1, 1)$, thus $-1 < \xi < 1$
Maximize

$$|W(x)| = |x(x-1)(x+1)| = |x^3 - x|$$

Since $W'(x) = 3x^2 - 1 = 0$ has roots $x = \pm\sqrt{\frac{1}{3}}$
The local maximum and minimum occur at among the roots and the boundary,

$$\begin{aligned} W(\sqrt{\frac{1}{3}}) &= -\frac{2\sqrt{3}}{9} \approx -0.385 \\ W(\sqrt{-\frac{1}{3}}) &= \frac{2\sqrt{3}}{9} \approx 0.385 \\ W(1) &= 0 \\ W(-1) &= 0 \end{aligned}$$

Thus

$$\max_{-1\le x\le 1}|W(x)| \le 0.385$$

So

$$\begin{aligned}\max_{-1\le x\le 1}|e^x - p_2(x)| &\le |\frac{e^\xi}{6}\times 0.385|\\ &< \frac{e^1}{6}\times 0.385\\ &< 0.174\end{aligned}$$

III.
For $-1 \le x \le 2$, we only need to consider $1 \le x \le 2$ in addition to II.
Let $1 \le x \le 2$, $ospr\{x, x_0, x_1, x_2\} = ospr\{x, -1, 0, 1\} = (-1, 2)$, thus $-1 < \xi < 2$
Maximize the same $W(x)$ as II., we know $W(x)$ increases on $(\sqrt{\frac{1}{3}}, \infty)$
Also by

$$\begin{aligned}W(1) &= 0\\ W(2) &= 6\end{aligned}$$

Thus

$$\max_{1\le x\le 2}|W(x)| \le 6$$

So

$$\begin{aligned}\max_{1\le x\le 2}|e^x - p_2(x)| &\le |\frac{e^\xi}{6}\times 6|\\ &< \frac{e^2}{6}\times 6\\ &< 7.39\end{aligned}$$

Combine the answer in II.

$$\max_{-1\le x\le 2}|e^x - p_2(x)| < \max\{7.39, 0.174\} = 7.39$$

IV.
For $-2 \le x \le 1$, we only need to consider $-2 \le x \le -1$ in addition to II.
Let $-2 \le x \le -1$, $ospr\{x, x_0, x_1, x_2\} = ospr\{x, -1, 0, 1\} = (-2, 1)$, thus $-2 < \xi < 1$
Maximize the same $W(x)$ as II., we know $W(x)$ increases on $(-\infty, -\sqrt{\frac{1}{3}})$
Also by

$$\begin{aligned}W(-1) &= 0\\ W(-2) &= -6\end{aligned}$$

Thus

$$\max_{-2\le x\le -1} |W(x)| \le 6$$

So

$$\begin{aligned}\max_{-2\le x\le -1} |e^x - p_2(x)| &\le |\frac{e^\xi}{6}\times 6| \\ &< \frac{e}{6}\times 6 \\ &< 2.72\end{aligned}$$

Combine the answer in II.

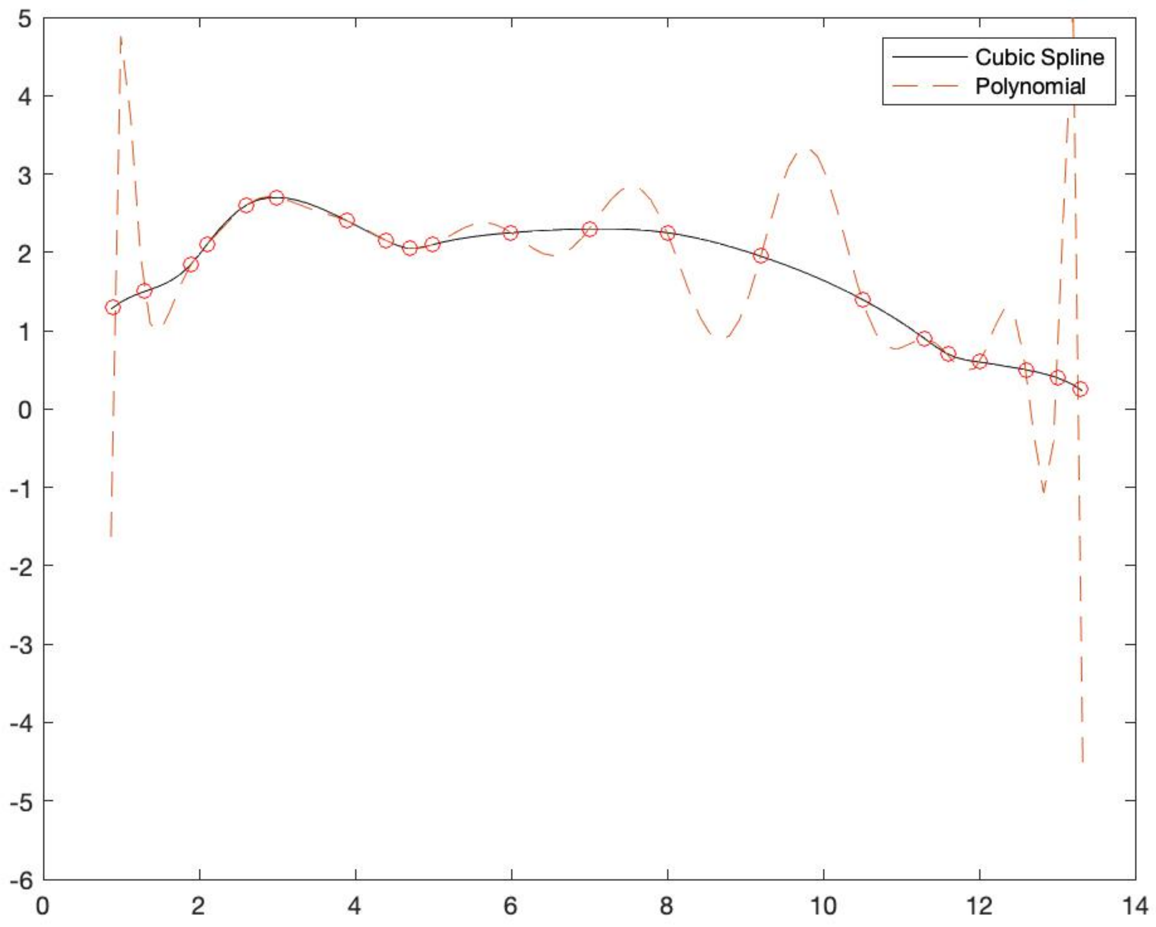
$$\max_{-2\le x\le 1} |e^x - p_2(x)| < \max\{2.72, 0.174\} = 2.72$$

Question 4.

MATLAB script:

```
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];
xi  = linspace(0.87, 13.33, 100);

% not-a-know cubic spline interpolant
yvs = spline(x, y, xi);

% 20th degree polynomial interpolant
p = polyfit(x, y, 20);
polyfits = polyval(p, xi);

% plot
plot(xi, yvs, 'k-', ...
    xi, polyfits, '--', ...
    x, y, 'ro');
axis([0 14 -6 5]);
legend('Cubic Spline', 'Polynomial');
```

Figure 3: Result



Comment: The interpolation function given by the cubic spline is much more stable and smoother, especially at the knots or the boundary. The polynomial interpolant oscillates a lot so that the accuracy is not high given the very high degree (20). Consider the MATLAB warning: ‘Polynomial is badly conditioned.’, i.e. the result is not numerically stable. In a nutshell, Runge’s phenomenon can be witnessed.