CSC336

A3

RUNCHAO

MAO

1003151938

1(a) $\bar{f}(\bar{x}^{(i)}) = \bar{0}$

$$f = \begin{bmatrix} f_1(x_1^{(i)}, x_2^{(i)}) \\ f_2(x_2^{(i)}, x_1^{(i)}) \\ f_3(x_2^{(i)}, x_3^{(i)}) \end{bmatrix}$$

$$f_1(x_1^{(i)}, x_2^{(i)}, x_3^{(i)}) = x_1^{(i-1)} + h\mu N - h\mu x_1^{(i)} - h\frac{\beta}{N} x_1^{(i)} x_2^{(i)} - x_1^{(i)}$$

$$f_2(x_1^{(i)}, x_2^{(i)}, x_3^{(i)}) = x_2^{(i-1)} - h\gamma x_2^{(i)} - h\mu x_2^{(i)} + h\frac{\beta}{N} x_1^{(i)} x_2^{(i)} - x_2^{(i)}$$

$$f_2(x_1^{(i)}, x_2^{(i)}, x_3^{(i)}) = x_3^{(i-1)} + h\gamma x_2^{(i)} - h\mu x_3^{(i)} - x_3^{(i)}$$

$$J(x) = \begin{bmatrix} -h\mu - h\frac{\beta}{N} x_2^{(i)} - 1, & -h\frac{\beta}{N} x_1^{(i)} & , 0 \\ h\frac{\beta}{N} x_2^{(i)} & , -h\gamma - h\mu + h\frac{\beta}{N} x_1^{(i)} - 1 & , 0 \\ 0 & , h\gamma & , -h\mu - 1 \end{bmatrix}$$

1b

```
% computing the spreading of influenza
% test for Newton's on the first timestep

% beta transmission, gamma recovery, mu death/birth (replenishment)
beta = 1.00; gamma = 0.20; mu = 0.05;

% initial conditions
N = 500; y02 = 10; y0 = [N-y02 y02 0]';

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

maxit = 10; tol = 1e-6; % Newton's parameters
fprintf(' k S            I            R            Total       Residual\n');
y = y0;
yinit = y; % initial guess for Newton's
for k = 1:maxit
    % define vector f and its inf norm
    trans_y0 = transpose(y0);
    trans_y = transpose(y);
    f_1 = trans_y0[1] + Mu*N - Mu*trans_y[1] - Beta*trans_y[1]*trans_y[2]
- trans_y[1]
    f_2 = trans_y0[2] + Gamma*trans_y[2] - Mu*trans_y[2] +
Beta*trans_y[1]*trans_y[2] - trans_y[2]
    f_3 = trans_y0[3] + Gamma*trans_y[2] - Mu*trans_y[3] - trans_y[3]

    f = [f_1, f_2, f_3];
    fnorm = norm(f, inf);
    fprintf('%2d %10.6f %9.6f %9.6f %10.6f %9.2e\n', k-1, y, sum(y),
fnorm);
    % stopping criterion
    if fnorm < tol:
        break
    endif
    % define Jacobian matrix
    r_1 = [-Mu -Beta*trans_y[2] - 1, -Beta*trans_y[1], 0];
    r_2 = [Beta*y[2], -Gamma -Mu + Beta * trans_y[1] - 1, 0];
    r_3 = [0, Gamma, -Mu - 1];
    J = [r1;r2;r3];
    J_inv = inv(J);
    % apply Newton's iteration to compute new y
    y = y - J_inv * transpose(f);
end
```

1c

```
% computing the spreading of influenza

% beta transmission, gamma recovery, mu death/birth (replenishment)
beta = 1.00; gamma = 0.20; mu = 0.05;

% initial conditions
N = 500; y02 = 10; y0 = [N-y02 y02 0]'; tend = 50;

dt = 1/24; nstep = tend/dt; % stepsize in time and number of time points
Beta = dt*beta/N; Gamma = dt*gamma; Mu = dt*mu;

maxit = 10; tol = 1e-6; % Newton's parameters
y = y0; yi(:, 1) = y;
array = []
for i = 1:nstep % nstep*dt days
    yinit = y; % initial guess for Newton's of the i-th step
    y0 = y;
    array = [array i*dt]
    for k = 1:maxit
        % define vector f and its inf norm
        trans_y0 = transpose(y0);
        trans_y = transpose(y);
        f_1 = trans_y0[1] + Mu*N - Mu*trans_y[1] -
Beta*trans_y[1]*trans_y[2] - trans_y[1]
        f_2 = trans_y0[2] + Gamma*trans_y[2] - Mu*trans_y[2] +
Beta*trans_y[1]*trans_y[2] - trans_y[2]
        f_3 = trans_y0[3] + Gamma*trans_y[2] - Mu*trans_y[3] - trans_y[3]
        f = [f_1, f_2, f_3];
        fnorm = norm(f, inf);
        fprintf('%2d %10.6f %9.6f %9.6f %10.6f %9.2e\n', k-1, y, sum(y),
fnorm);
        % stopping criterion
        if fnorm < tol:
            break
        % define Jacobian matrix
        r_1 = [-Mu -Beta*trans_y[2] - 1, -Beta*trans_y[1], 0];
        r_2 = [Beta*y[2], -Gamma -Mu + Beta * trans_y[1] - 1, 0];
        r_3 = [0, Gamma, -Mu - 1];
        J = [r1;r2;r3];
        J_inv = inv(J);
        % apply Newton's iteration to compute new y
        y = y - J_inv * transpose(f);
    end
```

```
    yi(:, i+1) = y;
    %nit(i) = k;
end

t = (0:nstep)*dt; yn = y;
fprintf('         S       I       R       Total\n');
fprintf('initial: %6.2f %6.2f %6.2f %6.2f\n', y0, sum(y0));
fprintf('end     : %6.2f %6.2f %6.2f %6.2f\n', yn, sum(yn));
fprintf('max infected: max %7.2f\n', max(yi(2, :)));

% do the plot
figure
plot(array, yi[1, :], array, yi[2, :], '--',array, yi[3, :], '-.')
```

1d

```
% computing the spreading of influenza

% beta transmission, gamma recovery, mu death/birth (replenishment)
beta = 1.00; gamma = 0.20; mu = 0.05;

% initial conditions
N = 500; y02 = 10; y0 = [N-y02 y02 0]'; tend = 50;

dt = 1/24; nstep = tend/dt; % stepsize in time and number of time points
Beta = dt*beta/N; Gamma = dt*gamma; Mu = dt*mu;

maxit = 10; tol = 1e-6; % Newton's parameters
y = y0; yi(:, 1) = y;
array = []
for i = 1:nstep % nstep*dt days
    if i >= 5/dt:
        beta = 0.15;
        Beta = dt*beta/N;
    end

    yinit = y; % initial guess for Newton's of the i-th step
    y0 = y;
    array = [array i*dt]
    for k = 1:maxit
        % define vector f and its inf norm
```

```
        trans_y0 = transpose(y0);
        trans_y = transpose(y);
        f_1 = trans_y0[1] + Mu*N - Mu*trans_y[1] -
Beta*trans_y[1]*trans_y[2] - trans_y[1]
        f_2 = trans_y0[2] + Gamma*trans_y[2] - Mu*trans_y[2] +
Beta*trans_y[1]*trans_y[2] - trans_y[2]
        f_3 = trans_y0[3] + Gamma*trans_y[2] - Mu*trans_y[3] - trans_y[3]
        f = [f_1, f_2, f_3];
        fnorm = norm(f, inf);
        fprintf('%2d %10.6f %9.6f %9.6f %10.6f %9.2e\n', k-1, y, sum(y),
fnorm);
        % stopping criterion
        if fnorm < tol:
            break
        % define Jacobian matrix
        r_1 = [-Mu -Beta*trans_y[2] - 1, -Beta*trans_y[1], 0];
        r_2 = [Beta*y[2], -Gamma -Mu + Beta * trans_y[1] - 1, 0];
        r_3 = [0, Gamma, -Mu - 1];
        J = [r1;r2;r3];
        J_inv = inv(J);
        % apply Newton's iteration to compute new y
        y = y - J_inv * transpose(f);
    end

    yi(:, i+1) = y;
    %nit(i) = k;
end

t = (0:nstep)*dt; yn = y;
fprintf('          S      I      R      Total\n');
fprintf('initial: %6.2f %6.2f %6.2f %6.2f\n', y0, sum(y0));
fprintf('end    : %6.2f %6.2f %6.2f %6.2f\n', yn, sum(yn));
fprintf('max infected: max %7.2f\n', max(yi(2, :)));

% do the plot
figure
plot(array, yi[1, :], array, yi[2, :], '--',array, yi[3, :], '-.')
```

2. (a) $\frac{a_i}{b_i} = \frac{a_{i-1}+2b_{i-1}}{a_{i-1}+b_{i-1}} = 1+\frac{b_{i-1}}{a_{i-1}+b_{i-1}} = 1+\frac{1}{\frac{a_{i-1}}{b_{i-1}}+1} = 1+\frac{1}{x_{i-1}+1}$

$= x_i$

(b) $x = 1+\frac{1}{x+1}$

$x(x+1) = x+1+1$

$x^2+x = x+2$

$x^2=2$

$x=\sqrt{2}$

(c) if $x_{i-1}=1$, $1+\frac{1}{1+1} > 1$

if $x_{i-1}=2$ $1+\frac{1}{2+1} < 2$

Then we show that it is continuus.

By drawing the graph we know that it is continuus on $(1,2)$

so by Intermediate Value Theorem.

root $x \in (1,2)$

To prove uniqueness.

$f(x) = 1+\frac{1}{x+1} - x$

$f'(x) = -(x+1)^{-2}-1$

if $-(x+1)^{-2}-1=0$.

then $(x+1)^{-2}=-1$

impossible, so

$f'(x) = -(x+1)^{-2}-1 \neq 0$.

Therefore the root is unique.

Over $(1,2)$,

Because $y=x+1$ is monotonic

$z=y^2$ is monotonic

$\frac{1}{z}$ is monotonic.

we know that $f(x)$ is monotonic.

(d) order $=1$.

(e) interval is $(-1, +\infty)$, achieving from graph $f(x) = 1 + \frac{1}{1+x} - x$ because where it is continuous.

3. (a) Using Lagrange

$(\frac{1}{2}, -1), (1, 0), (2, 1)$

$$P_2(x) = -l_0(x) + 0\, l_1(x) + l_2(x)$$

$$= -\frac{(x-1)(x-2)}{(\frac{1}{2}-1)(\frac{1}{2}-2)} + \frac{(x-\frac{1}{2})(x-1)}{(2-\frac{1}{2})(2-1)}$$

$$= -\frac{(x-1)(x-2)}{(-\frac{1}{2})(-\frac{3}{2})} + \frac{(x-\frac{1}{2})(x-1)}{(\frac{3}{2})(1)}$$

$$= -\frac{4}{3}(x-1)(x-2) + \frac{2}{3}(x-\frac{1}{2})(x-1)$$

$$= -\frac{4}{3}(x^2-3x+2) + \frac{2}{3}(x^2-\frac{3}{2}x+\frac{1}{2})$$

$$= -\frac{4}{3}x^2 - 4x - \frac{8}{3} + \frac{2}{3}x^2 - x + \frac{1}{3}$$

$$= -\frac{7}{3} - 5x - \frac{2}{3}x^2$$

(b) $f(x) = \log_2 x \quad f'(x) = \frac{1}{x \ln 2}$

$$f''(x) = -\frac{1}{\ln(2)\, x^2}$$

$$f'''(x) = \frac{1}{\ln(2)\, x^3}$$

$$f(x) - P_2(x) = \frac{\frac{1}{\ln(2)\, \xi^3}}{3!}(x-\frac{1}{2})(x-1)(x-2)$$

(c) let $g(x) = (x-\frac{1}{2})(x-1)(x-2)$

$$= (x^2 - \frac{3}{2}x + \frac{1}{2})(x-2)$$

$$= x^3 - \frac{3}{2}x^2 + \frac{1}{2}x - 2x^2 + 3x - 1$$

$$= x^3 - \frac{7}{2}x^2 + \frac{7}{2}x - 1$$

interval $\in [\frac{1}{4}, 2]$

$x_1 \in$ interval.

$x_2 \in$ interval.

$$g'(x) = 3x^2 - 7x + \frac{7}{2} = 0.$$

$$x_1 = \frac{7+\sqrt{7}}{6} \qquad x_2 = \frac{7-\sqrt{7}}{6}$$

$$|f(x) - P_2(x)| \leq \frac{\left(\frac{98-7\sqrt{7}}{108} - 1\right)}{6 \cdot \ln(2)\, (\frac{1}{4})^3}$$

(d) let $g(x) = (x-\frac{1}{2})(x-1)(x-2)$

$$= x^3 - \frac{7}{2}x^2 + \frac{7}{2}x - 1$$

$$g'(x) = 3x^2 - 7x + \frac{7}{2} = 0.$$

$$x = \frac{7+\sqrt{7}}{6} \quad x_2 = \frac{7-\sqrt{7}}{6} \quad x_1, x_2 \in [\frac{1}{4}, 3]$$

$$|f(x) - P_2(x)| \le \frac{-(\frac{98-7\sqrt{7}}{108} - 1)}{6 \cdot \ln(2)(\frac{1}{4})^3}$$

4

```
a = 1/4; b = 4;
xe = linspace(a, b, 1000);
ye = log2(xe);

disp(['interval (' num2str(a) ',' num2str(b), ')'])
disp('  n     err poly    err lin_spl')
for nn = 1:6
    n = 2^nn;
    xi = linspace(a, b, n+1);
    yi = log2(xi);
    yp = polyval(polyfit(xi, yi, n), xe);
    yl = interp1(xi, yi, xe, 'linear');
    ep = max(abs(ye-yp));
    el = max(abs(ye-yl));
    fprintf('%3d %12.3e %12.3e\n', n, ep, el);
end

fprintf('\n')
disp(['interval (' num2str(a) ',' num2str(b), ')'])
disp('  n    err lin_spl err cub_spl')
for nn = 4:9
    n = 2^nn;
    xi = linspace(a, b, n+1);
    yi = log2(xi);
    yl = interp1(xi, yi, xe, 'linear');
    yc = spline(xi, yi, xe);
    yy = ppval(yc, xi)% use cubic spline interpolation (not-a-knot)
    el(nn) = max(abs(ye-yl));
    ec(nn) = max(abs(ye-yc));
    fprintf('%3d %12.3e %12.3e ', n, el(nn), ec(nn));
    if (nn > 4)
        fprintf('%6.1f %6.1f\n', log(el(nn-1)/el(nn))/log(2), ...
                                 log(ec(nn-1)/ec(nn))/log(2));
    else fprintf('\n'); end
end

a = 1e-2; b = 4;
xe = linspace(a, b, 1000);
ye = log2(xe);

fprintf('\n')
disp(['interval (' num2str(a) ',' num2str(b), ')'])
disp('  n    err lin_spl err cub_spl')
for nn = 4:9
    n = 2^nn;
```

```
    xi = linspace(a, b, n+1);
    yi = log2(xi);
    yl = interp1(xi, yi, xe, 'linear');
    yc = spline(xi, yi, xe);
    yy = ppval(yc, xi)% use cubic spline interpolation (not-a-knot)
    el(nn) = max(abs(ye-yl));
    ec(nn) = max(abs(ye-yc));
    fprintf('%3d %12.3e %12.3e ', n, el(nn), ec(nn));
    if (nn > 4)
        fprintf('%6.1f %6.1f\n', log(el(nn-1)/el(nn))/log(2), ...
                                 log(ec(nn-1)/ec(nn))/log(2));
    else fprintf('\n'); end
end
```