Assignment 1

September 5, 2023 1:01 PM

1. Given $x \in \mathbb{R}$, truncation to k digits is equivalent to truncating to $k-1$ digits after the decimal due to the machine representation of the mantissa.

The truncation of x (denoted $\tilde{x}$) can thus be written as $\tilde{x} = \frac{\lfloor \beta^{k-1} x \rfloor}{\beta^{k-1}}$.

It should also be noted that $x \le \frac{\lceil \beta^{k-1} x \rceil}{\beta^{k-1}}$. Therefore $x - \tilde{x} \le \frac{\lceil \beta^{k-1} x \rceil}{\beta^{k-1}} - \frac{\lfloor \beta^{k-1} x \rfloor}{\beta^{k-1}} = \frac{1}{\beta^{k-1}}\left(\lceil \beta^{k-1} x \rceil - \lfloor \beta^{k-1} x \rfloor\right)$.

Since $\forall a \in \mathbb{R}\ \lceil a \rceil - \lfloor a \rfloor \le 1$ and $\forall x \in \mathbb{R}\ x - \tilde{x} \le \beta^{1-k}$, $\varepsilon_{mach} = \beta^{1-k}$. $\square$

2. $x_1 + dx_2 = 1$ & $dx_1 + x_2 = 0 \implies \begin{pmatrix} 1 & d \\ d & 1 \end{pmatrix}\begin{pmatrix} x_1 \\ x_2 \end{pmatrix} = \begin{pmatrix} 1 \\ 0 \end{pmatrix} \implies \begin{pmatrix} 1 & d \\ 0 & 1-d^2 \end{pmatrix}\begin{pmatrix} x_1 \\ x_2 \end{pmatrix} = \begin{pmatrix} 1 \\ -d \end{pmatrix} \implies \begin{pmatrix} 1 & 0 \\ 0 & 1 \end{pmatrix}\begin{pmatrix} x_1 \\ x_2 \end{pmatrix} = \begin{pmatrix} 1 + \frac{d^2}{1-d^2} \\ -d/_{1-d^2} \end{pmatrix}$.

This means the problem is well posed. Now for the condition number:

$$\text{cond} = \frac{\|f(d+\Delta d) - f(d)\|_\infty}{\|f(d)\|_\infty} \frac{\|d\|_\infty}{\|\Delta d\|_\infty}$$

$$= \left\| \begin{pmatrix} \frac{1}{1-(d+\Delta d)^2} \\ \frac{-(d+\Delta d)}{1-(d+\Delta d)^2} \end{pmatrix} - \begin{pmatrix} \frac{1}{1-d^2} \\ \frac{-d}{1-d^2} \end{pmatrix} \right\|_\infty \frac{|d|}{|\Delta d|\left(\frac{1}{1-d^2}\right)}$$

$$= \frac{|d|(1-d^2)}{|\Delta d|} \left| \frac{1}{(1-(d+\Delta d)^2)(1-d^2)} \right| \left\| \begin{pmatrix} 1-d^2-1+(d+\Delta d)^2 \\ -(1-d^2)(d+\Delta d) + d(1-(d+\Delta d)^2) \end{pmatrix} \right\|_\infty$$

$$= \frac{|d|}{|\Delta d|(1-(d+\Delta d)^2)} \left\| \begin{matrix} 2d\Delta d + \Delta d^2 \\ -(d+\Delta d - d^3 - d^2\Delta d) + (d - d^3 - 2d^2\Delta d - d^2\Delta d^2) \end{matrix} \right\|_\infty$$

$$= \frac{|d|}{|\Delta d|(1-(d+\Delta d)^2)} \left\| \begin{matrix} (2d+\Delta d)\Delta d \\ -\Delta d - d^2\Delta d - d^2\Delta d^2 \end{matrix} \right\|_\infty$$

$$= \frac{|d|}{1-(d+\Delta d)^2} \left\| \begin{matrix} 2d+\Delta d \\ 1+d^2(1+\Delta d) \end{matrix} \right\|_\infty$$

Taking $\lim_{\Delta d \to 0}$, since $|1+d^2| \ge |2d|$, $\text{cond} = \frac{|d|(1+d^2)}{1-d^2}$. $\square$

Since cond is symmetric & monotonic on $(0,1)$, binary search can give $\boxed{|d| < 0.9125...}$ (see q2.m)

3. a. See ForwardSubstitute.m

b.

```
function BackwardSubstitution(U,b)
    x_n ← b_n/U_nn
    for i=n-1 to 1 do
        s ← b_i
        for j=i+1 to n do
            s ← s - U_ij x_j
        x_i ← s/U_ii
    return x
```

7 $\quad$ $x_i \leftarrow s/u_{ii}$

8 $\quad$ return x

c. Line 2: 1 division = 1 FLOPs

Line 6: $(n-i-1)$ multiplications & subtractions $= 2(n-i-1)$ FLOPs

Line 7: 1 division = 1 FLOPs

Total FLOPs:

$$T = 1 + \sum_{i=1}^{n-1}(2n-2i+1) = 1 + (2n+1)(n-1) - n(n-1)$$

$$= n^2 \in O(n^2)$$

d. See BackwardSubstitution.m

e. See LUDecomposition.m

f. See LUSolve.m

4. a. For brevity let $p_i \equiv p(t_i)$, $q_i \equiv (t_i)$, $b_i \equiv b(t_i)$ for $i = 0, \dots, n+1$. Using (2)&(3), (1) becomes:

$$\frac{x_{i+1} - 2x_i + x_{i-1}}{h^2} = p_i \frac{x_{i+1} - x_{i-1}}{2h} + q_i x_i + b_i \;,\; x_0 = \alpha, x_{n+1} = \beta.$$

$$x_{i+1}\left(\frac{1}{h^2} - \frac{p_i}{2h}\right) - x_i\left(\frac{2}{h^2} + q_i\right) + x_{i-1}\left(\frac{1}{h^2} + \frac{p_i}{2h}\right) - b_i = 0$$

$$x_{i+1}\left(\tfrac{h}{2}p_i - 1\right) + x_i\left(h^2 q_i + 2\right) - x_{i-1}\left(\tfrac{h}{2}p_i + 1\right) = -h^2 b_i \implies \boxed{\begin{aligned} a_{i,i-1} &= -\tfrac{h}{2}p_i - 1 \\ a_{i,i} &= h^2 q_i + 2 \\ a_{i,i+1} &= \tfrac{h}{2}p_i - 1 \end{aligned}}$$

b. $L = \|p\|_\infty = \max_{t\in[0,1]} |p(t)|$, $hL < 2$. A is strictly row diagonal dominant when

$$|a_{i,i}| > \sum_{j\neq i} |a_{i,j}| = |a_{i,i-1}| + |a_{i,i+1}|$$

$$|h^2 q_i + 2| > |-\tfrac{h}{2}p_{i+1} - 1| + |\tfrac{h}{2}p_{i-1} - 1| = 2 \quad \left(\text{since } \max_{t\in[0,1]}|p(t)| = L, \tfrac{hL}{2} < 1, \text{ and } |-t-1| + |t-1| = 2\ \forall |t| < 1\right)$$

Since $h^2 > 0$ and $q_i > 0$ the above always holds, so A is strictly row diagonal dominant. □

c. $L = 20$ so $h < \frac{1}{10}$ and $\boxed{n_0 = 10}$. There does appear to be convergence as $n \to \infty$ (see q4c.png)

d. $B_J = I - D^{-1}A \implies B_{J_{i,i}} = 0$, $B_{J_{i,i-1}} = \dfrac{-\frac{h}{2}p_i - 1}{h^2 q_i + 2}$, $B_{J_{i,i+1}} = \dfrac{\frac{h}{2}p_i - 1}{h^2 q_i + 2}$

$$\implies \|B_J\|_\infty = \max_{i\in\{1,\dots,n\}}\left(\frac{2}{h^2 q_i + 2}\right) \implies \boxed{\|B_J\|_\infty < \frac{2}{h^2 q_{min} + 2}}$$

Since $q_{min} > 0$ & $h > 0$ $\frac{2}{h^2 q_i + 2} \in (0,1)$ so it always converges in this case. □

$\boxed{\lim_{h\to 0} \|B_J\|_\infty = 1}$ so it gets very slow as $h \to 0$.

e. This took $\boxed{102958 \text{ iterations } (\|B_J\|_\infty > 0.99992)}$, LU is way faster here. When $q = 10^8(t^2+1)$, $p = \frac{1}{20}\cos(\pi t)$ it took $\boxed{7 \text{ iterations}}$. This is due to the fact that $h^2 q_{min} \sim 10^0$ so $\boxed{\|B_J\|_\infty \approx 1/9.}$ (see q4e1.png & q4e2.png)
The LU method was slower here since it had to perform $O(n^2)$ FLOPs while Jacobi effectively only needed 7 $O(n)$ operations.

5. a. First note that $f(x) = \begin{pmatrix} 1 \\ x \\ x^2 \end{pmatrix} \cdot c$. If $y_i = f(x_i)\ \forall i \in \{1, \dots, n\}$, then $\begin{pmatrix} 1 & x_1 & x_1^2 \\ \vdots & \vdots & \vdots \\ 1 & x_n & x_n^2 \end{pmatrix}\begin{pmatrix} c_0 \\ c_1 \\ c_2 \end{pmatrix} = \begin{pmatrix} y_1 \\ \vdots \\ y_n \end{pmatrix}$ is required.

Since M has more rows than columns, it must be overdetermined. □

Since M has more rows than columns, it must be overdetermined. $\square$

b. $A^T = (M^TM)^T = M^T(M^T)^T = M^TM = A$ $\square$

c. $Mc = y \implies M^TMc = M^Ty \implies \boxed{b = Ay}$

d. See LeastSquaresQuadratic.m.

e. The best fit quadratic was $y = -19.4257 + 5.7451x - 0.2841x^2$ with $\|\tilde{y} - \tilde{y}\| = 2347726$. (see q5e.m).

Question 2

```
function q2()
    fprintf("%d\n", BinarySearch(@(d) abs(d) * (1 + d^2) / (1 - d^2), 10, 0, 1, 1e-6));
end
```

Listing 1: Code for question 2 calls BinarySearch.m.

```
function x  = BinarySearch(f, goal, min, max, tol)
    x = (min+max)/2;
    y = f(x);
    while abs(y-goal) > tol
        if y < goal
            min = x;
        else
            max = x;
        end
        x = (min+max)/2;
        y = f(x);
    end
end
```

Listing 2: Code for Binary search called in q2.m.

Question 3

```
function y = ForwardSubstitute(L, b)
    % Inputs:
    %   L: A square lower triangular matrix
    %   b: a vector of the length as the columns of U
    % Outputs:
    %   y: A vector that is the solution to Ly = b.
    n = size(L, 1);
    % obtain the first element of y
    y = b / L(1, 1);
    % obtain the other elements of y by iterating downwards in L.
    for k=2:n
        y(k) = (1 / L(k, k)) * (b(k) - L(k, 1:k-1)*y(1:k-1));
    end
end
```

Listing 3: Code used for question 3a.

```
function x = BackwardSubstitute(U, b)
    % Inputs:
    %   U: A square upper triangular matrix
    %   b: a vector of the length as the columns of U
    % Outputs:
    %   x: A vector that is the solution to Ux = b.
    n = size(U, 1);
    % obtain the last element of x
    x = b / U(n, n);
    % obtain the other elements of x by iterating upwards in U.
    for k=n-1:-1:1
        x(k) = (1 / U(k, k)) * (b(k) - U(k, k+1:n) * x(k+1:n));
    end
end
```

Listing 4: Code used for question 3d.

```matlab
function A = LUDecomposition(A)
    % Inputs:
    %   A: A square matrix
    % Outputs:
    %   A: A square matrix whose upper and lower triangle (below the
    %   diagonal) are the respective U and L such that A = LU.
    n = size(A, 1);
    for k=1:n-1
        for i=k+1:n
            A(i, k) = A(i, k) / A(k, k);
            for j=k+1:n
                A(i, j) = A(i, j) - A(i, k) * A(k, j);
            end
        end
    end
end
```

Listing 5: Code used for question 3e.

```matlab
function x = LUSolve(A, b)
    % Inputs:
    %   A: A square matrix
    %   b: a vector of the length as the columns of A
    % Outputs:
    %   x: A vector that is the solution to Ax = b done by LU Decomposition.
    A = LUDecomposition(A);
    % Perform the LU Decomposition of A
    L = tril(A, -1) + eye(size(A, 1));
    U = triu(A);
    % Solve Ly = b and Ux = y in order to solve Ax = b
    x = BackwardSubstitute(U, ForwardSubstitute(L, b));
end
```

Listing 6: Code used for question 3f.

Question 4

```matlab
function q4c()
    % Code to produce graphs for A1.4
    n = arrayfun(@(x) 10 * 2^x, 0:5);
    for k=0:length(n)-1
        [T, X, ~] = ODESolve(@(t) 100 * (t^2 + 1), ...
                             @(t) 20 * cos(pi * t), ...
                             @(t) sin(pi * t), ...
                             2, -1, 0, 1, n(k+1), "LU");
        plot(T, X, 'DisplayName',"n = " + k);
        hold on;
    end
    legend()
end
```

Listing 7: Code used for question 4c.

```matlab
1  function q4e()
2      addpath 'C:\Users\rjust\f2023\math578'
3      % Code to produce graphs for A1.4e
4      %[T, X, iter] = ODESolve(@(t) 1000 * (t^2 + 1), ...
5      %                 @(t) 20 * cos(pi * t), ...
6      %                 @(t) sin(pi * t), ...
7      %                 2, -1, 0, 1, 2500, "J");
8      %[T, X, iter] = ODESolve(@(t) 10^8 * (t^2 + 1), ...
9      %                 @(t) (1/20) * cos(pi * t), ...
10     %                 @(t) sin(pi * t), ...
11     %                 2, -1, 0, 1, 2500, "J");
12     [T, X, iter] = ODESolve(@(t) 10^8 * (t^2 + 1), ...
13                    @(t) (1/20) * cos(pi * t), ...
14                    @(t) sin(pi * t), ...
15                    2, -1, 0, 1, 2500, "LU");
16     plot(T, X);
17     fprintf("%i\n", iter);
18 end
```

Listing 8: Code used for question 4e. It was run three times by uncommenting lines 3-6, 7-10, and 12-15 to produce the respective left and right subplots of Figure 2 (The last two blocks produce identical plots).

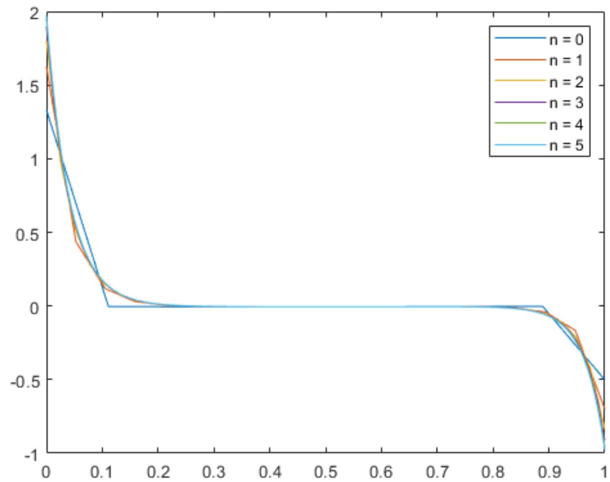

Figure 1: Graph of solution to the ODE for question 4c.

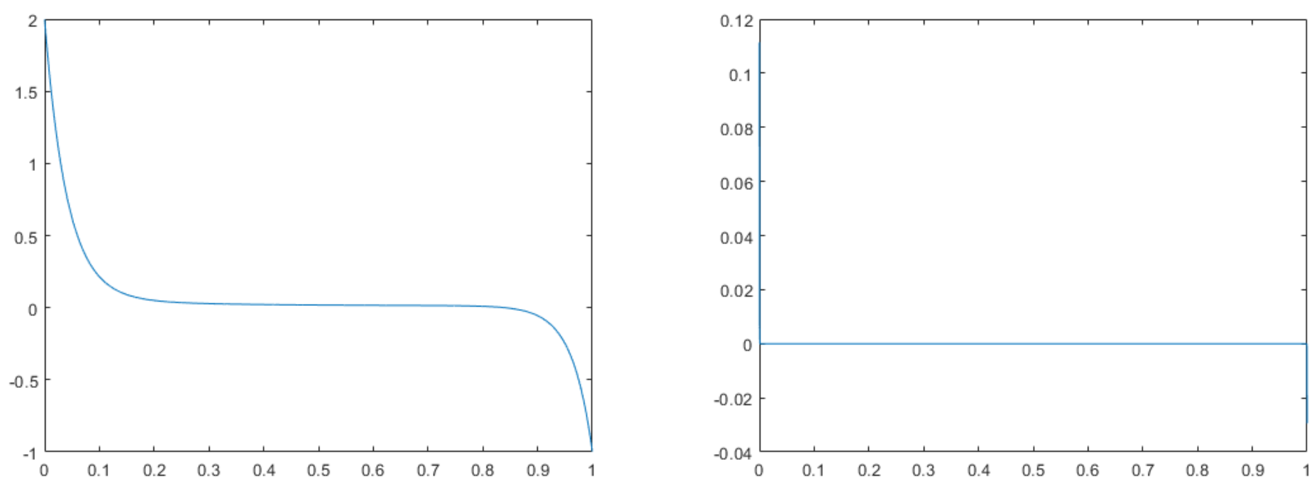

Figure 2: Graph of solution to the respective ODEs for question 4e. The new ODE given (the right subplot) presents much sharper transitions at $x = 0$ and $x = 1$.

Question 5

```
function c = LeastSquaresQuadratic(x, y)
    % Inputs:
    %   x: Vector of independant of the data
    %   y: Vector of dependant of the data
    % Outputs:
    %   c: Vector representing the coefficients of the best fitting
    %   quadratic to the data
    % Form M from x
    M = [ones(size(x)) x x.*x];

    % Solve M^TMc = M^Ty
    c = LUSolve(M.' * M, M.' * y);
end
```

Listing 9: Code used for question 5d.

```
function c = q5e()
    % code to produce graphs and error margins for 1.5e
    % form points
    addpath 'C:\Users\rjust\f2023\math578'
    x = [5, 5.5, 6.5, 8, 8.5, 10.8, 11.5, 13.7, 14.5, 15.9];
    y = [1, 4, 7, 8, 9.5, 9.2, 9, 6, 3, 1];

    % fitting
    c = LeastSquaresQuadratic(x.', y.');
    fit_fn = @(X) c(1) + c(2)*X + c(3)*X^2;

    % calculate residuals
    fprintf("%d\n", sqrt(sum((y - arrayfun(fit_fn, x)).^2)));

    % plot
    scatter(x, y)
    hold on;
    t = linspace(min(x), max(x));
    plot(t, arrayfun(fit_fn, t));
end
```

Listing 10: Code used for question 5e.