Review For Final Examination — Math 3607

Tae Eun Kim

Spring 2020

Below is the list of important concepts and topics to review from materials after Midterm 2.

Chapter 12: polynomial interpolation and approximation

- the idea, key types, and implementation of piecewise polynomial interpolation; B-splines will not be on the exam.
- least square approximation and normal equations;
- orthogonality – orthogonal vectors and orthogonal matrices;
- two versions of QR factorization, Moore-Penrose inverse of a matrix, a.k.a. the pseudo-inverse, and the connection to least square problems;
- triangularization of matrices – Gram–Schmidt (review this from linear algebra) and, more importantly, Householder.

Chapter 13: nonlinear rootfinding

- rootfinding tools built in MatLab;
- conditioning of rootfinding problem;
- various iteration methods – their ideas, implementation, and rates of convergence;

Chapter 14: numerical differentiation and integration

- different viewpoints on finite difference formulas (FD, BD, CD), *e.g.*, geometric, interpolation-based, extrapolated, ...;
- accuracy of finite difference formulas;
- determination of optimal step size for finite differences;
- Newton-Cotes family of quadrature formulas and their derivations from various viewpoints, *e.g.*, geometric, interpolation-based, extrapolated, ...;
- error analysis for quadrature methods – local and global;
- implementation of composite methods;

Chapter 15: ordinary differential equations

- significance of 1st-order ODEs (why is 1st-order sufficient?);
- Runge-Kutta methods (multi-step algorithms) – derivation and accuracy
- application/implementation of the algorithms;
- usage of `ode45`;

Problem 1. (Newton's Method)

In this problem, we calculate the value of $\sqrt[4]{7}$ numerically using Newton's method.

(a) Define function $f(x)$ whose root is $\sqrt[4]{7}$ and derive the corresponding Newton's iterative formula.

(b) Now write a MatLab program to implement Newton's method with the following details:

- Set $x_0 = 1.5$.
- You must include a statement defining $f(x)$.
- Utilize the previous result in the iterative steps; do NOT define $f'(x)$.
- Stop the iteration if either $|f(x_{n+1})| \le 10^{-10}$ or $|x_{n+1} - x_n| \le 10^{-8}$.

Solution:

(a) It is clear that $f(x) = x^4 - 7$ has $\sqrt[4]{7}$ as a root. Therefore, we obtain

$$x_{n+1} = x_n - \frac{f(x_n)}{f'(x_n)} = x_n - \frac{x_n^4 - 7}{4x_n^3} = \frac{3}{4}x_n + \frac{7}{4x_n^3}.$$

(b) An example program is presented below.

```
%% Newton's Method to Calculate \sqrt[4]{7}
f = @(x) x.^4 - 7;
xtrue = 7^(1/4);
nriter = 0;
x = 1.5;
fx = f(x);
while true
    nriter = nriter + 1;
    xnew = 1/4*(3*x + 7/x^3);
    fxnew = f(xnew);
    fprintf(' %6d : %16.8g %16.8g %16.8g \n',...
            nriter, xnew, fxnew, abs(xtrue-xnew)); % optional
    if abs(fxnew) <= 1e-10 || abs(xnew-x) <= 1e-8
        xzero = xnew;
        fprintf(' %s \n', 'The iteration is terminated');
        fprintf(' %s : %8.4e \n', 'The error', abs(xtrue-xzero));
        return
    end
    x = xnew;
    fx = fxnew;
end
```

Problem 2. (Trapezoidal Method)

Consider approximating the integral $I\{f\} = \int_a^b f(x)\,\mathrm{d}x = \int_0^\pi \sin x\,\mathrm{d}x$ using composite trapezoidal method

$$I_h^{[t]}\{f\} = h\left(\frac{f(x_1)}{2} + \sum_{i=2}^{n-1} f(x_i) + \frac{f(x_n)}{2}\right)$$

where $x_i = (i-1)h$ and $h = \pi/(n-1)$ for $i \in \mathbb{N}[1,n]$. Write a MATLAB program which calculates $I_h^{[t]}\{f\}$ for $n = 5, 9, 17, 33, 65, \ldots, 2049$. Tabulate the values of h, $I_h^{[t]}\{f\}$, and $I_h^{[t]}\{f\} - I\{f\}$ using `disp` or `fprintf` function.

Solution:

```
%% Composite Trapezoidal Method
n = 5;
nr_h = 10;
a = 0; b = pi;
f = @(x) sin(x);
f_int = @(x) -cos(x);
exact_area = f_int(b) - f_int(a);
M = zeros(nr_h, 3);
for i = 1:nr_h
    x = linspace(a, b, n)';
    h = x(2) - x(1);
    c = ones(n, 1);
    c([1,end]) = 1/2;
    area = c'*f(x)*h;
    M(i, 1) = h;
    M(i, 2) = area;
    M(i, 3) = area - exact_area;
    n = 2*n - 1;
end
fprintf(' %12s %16s %16s \n', 'h', 'trap approx', 'error')
fprintf(' %46s \n', repmat('-', 1, 29))
for i = 1:nr_h
    fprintf(' %12.4e %16.4e %16.4e \n', M(i, :));
end
```

(a) Use the first-order forward difference formula

$$D_h^{[1f]}\{f\}(x) = \frac{f(x+h)-f(x)}{h} = f'(x) + \frac{f''(x)}{2}h + \frac{f'''(x)}{6}h^2 + \mathcal{O}(h^3)$$

and Richardson extrapolation to obtain a second-order forward difference formula $D_h^{[2f]}\{f\}(x)$ for first derivatives. Write down explicitly the leading error term.

(b) Let $a < b$ and $x_j = a + (b-a)j/n$ for $j = 0, 1, 2, \ldots, n$ and suppose that the values $y_j = f(x_j)$, $0 \le j \le n$ are known and saved in MatLab as `ydp`; a, b, and n are also saved as `a`, `b`, `n`, respectively. Nothing else is known about f.

Your mission, should you choose to accept it, is to calculate numerically the derivatives $f'(x_j)$ for $0 \le j \le n$ using second-order methods using MatLab.

Solution:

(a) Write abstractly

$$\begin{aligned} D_h^{[1f]}\{f\}(x) &= f'(x) + c_1 h + c_2 h^2 + \cdots \\ D_{2h}^{[1f]}\{f\}(x) &= f'(x) + 2c_1 h + 4c_2 h^2 + \cdots \end{aligned}$$

Then it follows that

$$2D_h^{[1f]}\{f\}(x) - D_{2h}^{[1f]}\{f\}(x) = f'(x) - 2c_2h^2 + \cdots = f'(x) - \frac{f'''(x)}{3}h^2 + \mathcal{O}(h^3).$$

So the second-order forward difference formula is given by

$$\begin{aligned} D_h^{[2f]}\{f\}(x) &= 2D_h^{[1f]}\{f\}(x) - D_{2h}^{[1f]}\{f\}(x) \\ &= 2\frac{f(x+h)-f(x)}{h} - \frac{f(x+2h)-f(x)}{2h} = \frac{-3f(x)+4f(x+h)-f(x+2h)}{2h}. \end{aligned}$$

(b) An example code:

```
%% Numerical Differentiation Using Data
yprime = zeros(size(ydp));
h = (b-a)/n;
yprime(1) = (-3*ydp(1) + 4*ydp(2) - ydp(3))/(2*h);    % 2nd-order FD
yprime(2:end-1) = ...                                 % 2nd-order CD
    (ydp(3:end)-ydp(1:end-2))/(2*h);
yprime(end) = ...                                     % 2nd-order BD
    (3*ydp(end) - 4*ydp(end-1) + ydp(end-2))/(2*h);
```

Problem 4. (Mechanical Vibration and the Modified Euler Method)

Suppose that the motion of a certain spring-mass system satisfies the differential equation

$$u'' + u' + \frac{1}{5}u^3 = 3\cos\omega t$$

and the initial conditions

$$u(0) = 2, u'(0) = 0.$$

Write a MatLab program to plot the trajectory $u(t)$ for $0 \le t \le 100$ using the *modified Euler method* explained below.

The **modified Euler formula** for the initial value problem $\mathbf{y}' = \mathbf{f}(t, \mathbf{y})$, $\mathbf{y}(t_0) = \mathbf{y}_0$ is given by

$$\mathbf{y}_{n+1} = \mathbf{y}_n + h\mathbf{f}\left(t_n + \frac{h}{2}, y_n + \frac{h}{2}\mathbf{f}(t_n, \mathbf{y}_n)\right).$$

Solution: Since the given ODE solver is applicable to first-order systems, we first turn the given second-order ODE into a (2×2) system. Let $v = u'$ and $\mathbf{y} = (u, v)^T$. Then

$$\frac{d}{dt}\mathbf{y} = \begin{bmatrix} u' \\ v' \end{bmatrix} = \begin{bmatrix} v \\ -v - \frac{1}{5}u^3 + 3\cos\omega t \end{bmatrix} = \mathbf{f}(t, \mathbf{y}).$$

We solve this system using the modified Euler method as below.

```
%% Modified Euler Method
omega = ....;
f = @(t, y) [y(2); -y(2) - 1/5*y(1)^3 + 3*cos(omega*t)];
T = 100;
t = 0:0.1:T;
h = t(2) - t(1);
N = length(t) - 1;
y = zeros(2, N+1);
y0 = [2; 0];

y(:,1) = y0;
for n = 1:N
    fty = f(t(n), y(:,n));
    y(:,n+1) = y(:,n) + h*f(t(n)+h/2, y(:,n) + h/2*fty);
end
plot(t, y(1,:))
grid on, shg
```

Problem 5. (Lorenz Model and `ode45`)

The Lorenz equations are the nonlinear autonomous three-dimensional system

$$\begin{aligned}\dot{x} &= \sigma(y - x)\\ \dot{y} &= x(\rho - z) - y,\\ \dot{z} &= xy - \beta z\end{aligned}$$

where the dot notation indicates the time-derivative $\frac{d}{dt}$. Using

$$\sigma = 10, \quad \rho = 28, \quad \beta = 8/3,$$

plot the three-dimensional trajectory of the particle initially located at $(x, y, z) = (-8, 8, 27)$ for $0 \leq t \leq 10$ using `ode45`.

Solution:

```
%% Lorenz Equations
sigma = 10; rho = 28; beta = 8/3;
lorenz = @(t, w) [sigma*(w(2)-w(1));
                  w(1)*(rho-w(3)) - w(2);
                  w(1)*w(2) - beta*w(3)];
w0 = [-8; 8; 27];
dt = 0.01;
T = 10;
tspan = 0:dt:T;
[t, w] = ode45(lorenz, tspan, w0);

plot3(w(:,1), w(:,2), w(:,3))
grid on, shg
```

The figure below was obtained by running the code with `dt = 0.001` and `T = 100`.