PHY5340 Laboratory Report 3

Table of Contents

Jeremiah O'Neil, SN6498391

Question 1: Quadrature

Numerical integration methods:

```
type('trap.m')
type('simp.m')
type('emacl.m')
```

```
function [ result ] = trap( f, a, b, N )
x = linspace(a, b, N+1);
result = (b-a)/N * ((f(a) + f(b))/2 + sum(f(x(2:end-1))));
end

function [ result ] = simp( f, a, b, N )
x = linspace(a, b, N+1);
result = (b-a)/(3*N) * (f(a) + f(b) + 4*sum(f(x(2:2:end-1))) ...
                        + 2*sum(f(x(3:2:end-1))));
end

function [ result ] = emacl( f, fprime, a, b, N )
result = trap(f, a, b, N) + ((b-a)/N)^2/12 * (fprime(a) - fprime(b));
end
```

Error function integration and relative error calculation routine:

```
type('num_erf_err')
```

```
function [ err, hh, numerical ] = num_erf_err( h, method_handles )
g = @(y)2/sqrt(pi)*exp(-y.^2);
exact = 0.842700792949715;

N = 2*round(1/(2*h)); % ensuring N is even!
hh = 1/N; % interval size after correction for even integer N
nmethods = size(method_handles, 2);
err = zeros(1, nmethods); numerical = zeros(1, nmethods);
i = 1;
for method_handle = method_handles
    switch method_handle
        case 1
```

```
            numerical(i) = trap(g, 0, 1, N);
        case 2
            numerical(i) = simp(g, 0, 1, N);
        case 3
            gprime = @(y)-2*y.*g(y);
            numerical(i) = emacl(g, gprime, 0, 1, N);
        otherwise
            numerical(i) = 0;
    end
    err(i) = abs((numerical(i) - exact)/exact);
    i = i + 1;
end
end
```

Execute!

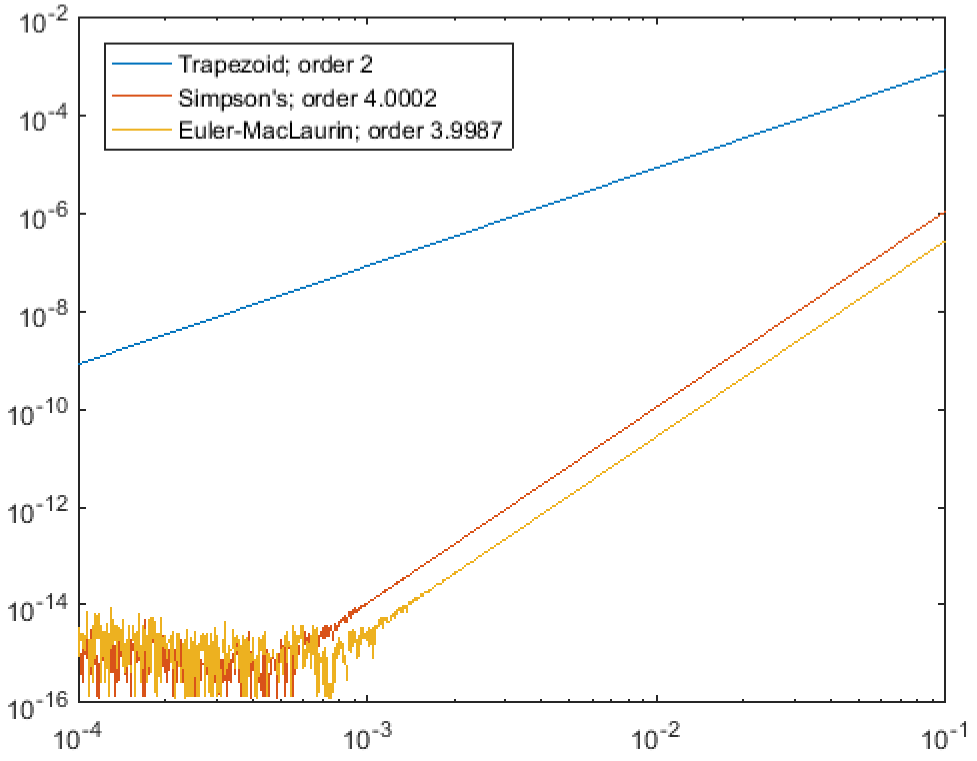
```
type('L3_Q1.m')
L3_Q1


numh = 1000;
H = logspace(-4, -1, numh);

% error for the 3 methods. 1:Trapezoid, 2:Simpson's, 3:Euler-MacLaurin
err = zeros(numh, 3);
i = 1;
for h = H
    % num_erf_err adjusts h to give even int N; reassign H(i) thus.
    [err(i, :), H(i)] = num_erf_err(h, [1, 2, 3]);
    i = i + 1;
end

order = zeros(1, 3);
for i = [1, 2, 3]
    % fit only where error is above precision
    crest = find(err(:, i) > 1e-14, 1);
    loglinfit = polyfit(log(H(crest:end)), log(err(crest:end, i).'),
 1);
    % method order of accuracy: negative slope of log linear fit
    order(i) = loglinfit(1);
end

loglog(H, err)
label1 = ['Trapezoid; order ' num2str(order(1))];
label2 = ['Simpson''s; order ' num2str(order(2))];
label3 = ['Euler-MacLaurin; order ' num2str(order(3))];
legend(label1, label2, label3, 'Location', 'northwest')
```



The plot shows that the trapezoid method is of order 2 while Simpson's method and the Euler-MacLaurin method or of order about 4, with Simpson's method being slightly better asymptotically for this problem --- but the Euler-MacLaurin method having a lower constant error. On the plotted range, only the trapezoid rule does not achieve machine precision.

Question 2: Eigenvalues

Lanczos algorithm

```
type('lanczos.m')
```

```
function eigvals = lanczos( A, n )
% originally my own, with improvements inspired by C4_1.
a = zeros(n, 1); b = zeros(n+1, 1);
v = randn(length(A), 1);
q = 0;
for i = 1:n
    b(i) = norm(v);
    bq_prev = b(i) * q;
    q = v / b(i);
    v = A * q;
    a(i) = q' * v;
    v = v - a(i) * q - bq_prev;
end
b = b(2:n);
```

```
triA = diag(a) + diag(b, -1) + diag(b', 1);
eigvals = sort(eig(triA));
end
```

Execute!

```
type('L3_Q2.m')
L3_Q2


A = [[12 13  2  -4  -7]
     [13 14  7  -4  12]
     [ 2  7  6  10   5]
     [-4 -4 10 -12  -1]
     [-7 12  5  -1 -10]];

precise_eigs = eig(A);
my_eigs = lanczos(A, 5);
err = NaN(5);
c = zeros(5, 2);
for i = 1:5
    err(1:i, i) = (lanczos(A, i) - precise_eigs(1:i)) ./
 precise_eigs(1:i);
    c(i, 1) = det(A - my_eigs(i)*eye(5));
    c(i, 2) = det(A - precise_eigs(i)*eye(5));
end
err
disp('err(n, m) is relative error in nth eigval for m Lanczos
 iterations.')
c
disp(['c(n, m) is det(A - e(n)*I) for e of m=1->lanczos(A, 5) and' ...
      ' m=2->eig(A).'])

err =

   -1.1108   -0.1475   -0.0003   -0.1177    0.0000
       NaN   -1.5273   -1.7663   -1.1198         0
       NaN       NaN    4.7483    1.4918   -0.0000
       NaN       NaN       NaN    1.1804   -0.0000
       NaN       NaN       NaN       NaN   -0.0000

err(n, m) is relative error in nth eigval for m Lanczos iterations.

c =

   1.0e-08 *

    0.0325   -0.0458
   -0.0702    0.0251
    0.0247   -0.0225
    0.1317    0.0168
    0.7044    0.7044

c(n, m) is det(A - e(n)*I) for e of m=1->lanczos(A, 5) and m=2-
>eig(A).
```

The Lanczos algorithm with 5 iterations is as good as MATLAB's eig() function for this problem, judging by the characteristic ('c', above); the difference fluctuates since the Lanczos starting vector is random, so it's too close to call!

Published with MATLAB® R2016a