Answer to Homework Set No. 4

1. Trapezoid and Simpson’s methods

   a). The trapezoid rule for uniform grid is:

   \[ T(f; h) = h \cdot \left[ \frac{1}{2} f(x_0) + \frac{1}{2} f(x_n) + \sum_{i=1}^{n-1} f(x_i) \right]. \]

   For our example, \( n = 4 \), and you get 0.55250538.

   b). The Simpson’s rule for uniform grid is:

   \[ S(f; h) = \frac{h}{3} \left[ f(x_0) + f(x_{2n}) + 4 \sum_{i=0}^{n-1} f(x_{2i+1}) + 2 \sum_{i=1}^{n-1} f(x_{2i}) \right]. \]

   For our example, \( n = 2 \), and you get 0.55067591.

   c). The exact value is \( e^0 - e^{-0.8} = 0.55067104 \). The absolute error with trapezoid rule is 0.00183435, with Simpson’s rule is 0.00048716. The Simpson’s method gives better result.

   d). Since \( f''(x) = e^{-x} \), then \(|f''| \leq 1\) on the interval \([0, 0.8]\). Then, the error bound for trapezoid rule is

   \[ |E_T| \leq \frac{0.8}{12} h^2 \]

   and the error bound for Simpson’s rule is

   \[ |E_S| \leq \frac{0.8}{180} h^4. \]

   If the error should be less than \( 10^{-4} \), then for the trapezoid rule, we require:

   \[ \frac{0.8}{12} h^2 \leq 10^{-4}, \quad \rightarrow \quad h \leq 0.0387298, \quad \rightarrow \quad n \geq 20.6 \]

   so we need at most \( n + 1 = 22 \) points. For the Simpson’s rule, we get

   \[ \frac{0.8}{180} h^4 \leq 10^{-4}, \quad \rightarrow \quad h \leq 0.3873, \quad \rightarrow \quad 2n \geq 2.06 \]

   and we need at most \( 2n + 1 = 4 \) points.

   We see that Simpson’s method need much fewer points to give the same accuracy as the trapezoid method with many more points.
2. Matlab problem: Trapezoid rule

The integrating function is defined in the file `funItg.m` as

```matlab
function y=funItg(x)
% function y=funItg(x)
% the function to be integrated in num itg.
y= exp(-x);
```

My version of the trapezoid rule reads like this:

```matlab
function v=trapezoid(f,a,b,n)
% function v=trapezoid(f,a,b,n)
% Compute the numerical integration with trapezoid rule
% Input: a,b: interval
% n: number of sub-intervals
% f: the given function to be integrated
% Output: v: the num itg of f.

h=(b-a)/n;
v=(feval(f,a)+feval(f,b))/2;
x=a; % the x-coordinate
for i=1:1:n-1,
x=x+h;
v=v+feval(f,x);
end
v=v*h;
```

If one uses Matlab function `sum`, the code could be much simpler. For example,

```matlab
function v=trapezoid(f,a,b,n)
    h=(b-a)/n;
fval=feval(f,[a:h:b]);
v=((fval(1)+fval(n+1))/2 + sum(fval(2:n)))*h;
end
```

Here is the script to run it:

```matlab
% a script to use the trapezoid.m
n=[4,8,16,32,64,128];
error=zeros(size(n));
a=0; b=0.8;
exact=exp(-a)-exp(-b); % the exact value
for i=1:1:length(n),
    error(i)=abs(trapezoid('funItg',a,b,n(i))-exact);
```
And I get the following plot:

Figure 1: Trapezoid rule

We see that when \( n \) doubles, which means \( h \) halves, the error is reduced by a factor of 4.

3. Matlab problem: Simpson’s rule

My version of the Simpson’s rule reads:

```matlab
function v=simpson(f,a,b,n)
% function v=simpson(f,a,b,n)
% Compute the numerical integration with Simpson rule
% Input: a,b: interval
%       n: number of sub-intervals
%       f: the given function to be integrated
% Output: v: the num itg of f.
```
h=(b-a)/2/n;
v=feval(f,a)+feval(f,b);
x=a+h;
for i=1:1:n, % sum over the odd index
    v=v+4*feval(f,x);
    x=x+h+h;
end
x=a+h+h;
for i=1:1:n-1, % sum over the even index
    v=v+2*feval(f,x);
    x=x+h+h;
end
v=v*h/3;

Or, if one uses directly some Matlab functions, it could be much simplified.

function v=simpson(f,a,b,n)
    h=(b-a)/2/n;
    xodd=[a+h:2*h:b-h]; % x_i with odd indices
    xeven=[a+2*h:2*h:b-2*h]; % x_i with even indices
    v=(h/3)*(feval(f,a)+4*sum(feval(f,xodd))+2*sum(feval(f,xeven))+feval(f,b));
end

% a script to use the trapezoid.m
n=[4,8,16,32,64,128];
error=zeros(size(n));
a=0; b=0.8;
exact=exp(-a)-exp(-b); % the exact value
for i=1:1:length(n),
    error(i)=abs(simpson('funItg',a,b,n(i))-exact);
end
loglog(n,error)
grid, title('Errors with trapezoid rule')
print -depsc simpsError.eps

And I get the following plot:

We see that when n doubles, which means h halves, the error is reduced by a factor of 16. The error reduces much faster here than the one of trapezoid rule.

4. Romberg algorithm

    a) My version of the file romberg.m looks like the following:
function R = romberg(f,a,b,n)
    h = b-a;
    R = zeros(n,n);
    R(1,1) = (feval(f,a) + feval(f,b))*h/2;
    for i=2:n
        h = 0.5*h;
        sum = 0;
        for k = 1:2:2^(i-1)-1
            sum = sum + feval(f,a+k*h);
        end
        R(i,1) = 0.5*R(i-1,1) + sum*h;
        for j = 2:i
            R(i,j) = R(i,j-1) + (R(i,j-1)-R(i-1,j-1))/(4^(j-1)-1);
        end
    end

b) Use the files:

    function y = f1(x)    function y = f2(x)
    y = sin(x);           y = sqrt(x);

    f1.m                    f2.m
And for f1.m, I get with $n = 4$:

\[
\begin{array}{cccc}
0.00000000000000 & 0 & 0 & 0 \\
1.57079632679490 & 2.09439510239320 & 0 & 0 \\
1.89611889793704 & 2.00455975498442 & 1.99857073182384 & 0 \\
1.97423160194555 & 2.00026916994839 & 1.99998313094599 & 2.00000554997967 \\
\end{array}
\]

In order to get the error along the diagonal, I run the following script:

```matlab
format short e;
R=romberg('f1',0,pi,7);
error=abs(diag(R)-2)
```

and I get:

```
error =
 2.0000e+00
 9.4395e-02
 1.4293e-03
 5.5500e-06
 5.4127e-09
 1.3212e-12
 8.8818e-16
```

And for f2.m, I repeat the same thing, and get the error:

```
error =
 2.1175e+00
 2.8863e+00
 2.9959e+00
 3.0285e+00
 3.0396e+00
 3.0434e+00
 3.0448e+00
```

We see that this is very poor result.

c) Romberg’s algorithm was developed based on the Euler-Maclaurin’s formula. This formula is valid if the function $f(x)$ is smooth such that all the derivatives are continuous and bounded. If this condition fails, then the Euler-Maclaurin’s is no longer valid, and therefore the method developed by it will no longer work. This is exactly what happened here. The function $f(x) = \sqrt{x}$ does not have a bounded derivative at $x = 0$.

d) Try the following:

```matlab
>> format long
>> res = quad('f1',0,pi,1e-9)
```
res = 
1.99999999999913
>> res = quadl('f1',0,pi,1e-9)
res = 
2.00000000000000
>> res = quad('f2',0,1,1e-9)
res = 
0.66666666157807
>> res = quadl('f2',0,1,1e-9)
res = 
0.66666666564636

We see that quadl works better than quad in general, but both functions have problem with the last integral.

5. More problems on numerical integration:

a) Let $T(h)$ be the trapezoid’s approximation to $J$ with equal spaced sub intervals $h$.

\[
T(1) = \frac{1}{2} [f(0) + f(1)] \\
T\left(\frac{1}{3}\right) = \frac{1}{3} \left[\frac{1}{2} f(0) + f\left(\frac{1}{3}\right) + \frac{2}{3} f(1)\right] \\
T\left(\frac{1}{9}\right) = \frac{1}{9} \left[\frac{1}{2} f(0) + f\left(\frac{1}{9}\right) + f\left(\frac{2}{9}\right) + f\left(\frac{3}{9}\right) + f\left(\frac{4}{9}\right) + f\left(\frac{5}{9}\right) \\
+ f\left(\frac{6}{9}\right) + f\left(\frac{7}{9}\right) + f\left(\frac{8}{9}\right) + \frac{1}{2} f(1)\right]
\]

The numbers of evaluations of $f$ are: 2 in $T(1)$, 2 more new ones in $T(1/3)$ and 6 more new ones in $T(1/9)$.

Totally 10 evaluations of the function $f$. This gives:

\[
T(1) = 0.920735493 \\
T(1/3) = 0.943291429 \\
T(1/9) = 0.945773189
\]

b) We start with the Euler-Maclaurin’s formulae:

\[
J = T(h) + a_2 h^2 + a_4 h^4 + a_6 h^6 + \cdots \tag{A}
\]

Since this is valid for all $h$, it also holds for $h/3$, and we get:

\[
J = T\left(\frac{h}{3}\right) + a_2 \left(\frac{h}{3}\right)^2 + a_4 \left(\frac{h}{3}\right)^4 + a_6 \left(\frac{h}{3}\right)^6 + \cdots \tag{B}
\]
Multiply the equation (B) by 9 and subtract (a) from it, we get
\[ 8J = 9T\left(\frac{h}{3}\right) - T(h) - \frac{8}{9}a_4h^4 - \frac{80}{81}a_6h^6 + \cdots \]
This gives us a new integration’s formulae:
\[ U(h) = T(h/3) + (T(h/3) - T(h))/8, \]
and the new Euler-Maclaurin’s formulae
\[ J = U(h) + c_4 h^4 + c_6 h^6 + \cdots \quad (C) \]
where \( c_4, c_6 \) are new constants. This form is valid also for \( h/3, \)
\[ J = U\left(\frac{h}{3}\right) + c_4 \left(\frac{h}{3}\right)^4 + c_6 \left(\frac{h}{3}\right)^6 + \cdots \quad (D) \]
Again, multiply (d) by 9 and subtract it from (C), then divide both sides by 80, we get
\[ J = U\left(\frac{h}{3}\right) + \frac{1}{80} \left[ U\left(\frac{h}{3}\right) - U(h) \right] + d_6 h^6 + \cdots \]
If we could continue, we would get a extrapolation’s formulae where
\[ U(i, 0) = T\left(\frac{h}{3^i}\right), \]
\[ U(i, j) = U(i, j - 1) + \frac{1}{9^j - 1} \left[ U(i, j - 1) - U(i - 1, j - 1) \right] \]
which would set up a table
\[
\begin{array}{cccc}
U(0,0) & U(1,0) & U(1,1) & U(2,0) \\
U(1,0) & U(1,1) & U(2,1) & U(2,2)
\end{array}
\]
or we put in the numbers:

\[
\begin{array}{cccc}
0.9207354924 & 0.9432914291 & 0.9461109212 & 0.9457731886 \\
0.9461109212 & 0.9460834085 & 0.9460830646 & 0.9460830646
\end{array}
\]