Sec 4.4: 1a, 2a, 3a, 7abc, 17

1a. Compute by hand or use a program. Matlab code for the Composite Trapezoidal rule:

```matlab
function integral = cmptrap(a,b,n,f)
    h = (b-a)/n;
    x = [a+h:h:b-h];
    integral = h/2*(2*sum(feval(f,x))+feval(f,a)+feval(f,b));
end
```

Run with
```matlab
cmptrap(1,2,4,'f')
```
where 'f' is the name of the function definition file

function y = f(t)
    y = t.*log(t); % pay attention to the dot
end

The result is \( \approx 0.6399004 \).

2a. Matlab code for the Composite Simpson’s rule

```matlab
function integral = cmpsimp(a,b,n,f)
    h = (b-a)/n;
    xi0 = feval('f',a)+feval('f',b);
    xi1 = 0;
    xi2 = 0;
    for i = 1:n-1
        x = a+i*h;
        if mod(i,2) == 0
            xi2 = xi2+feval('f',x);
        else
            xi1 = xi1+feval('f',x);
        end
    end
    xi = h*(xi0+2*xi2+4*xi1)/3;
end
```

Result: 0.6363098.

3a. Approximation: 
\[
2 \left( \frac{1}{6} \left( \frac{7}{6} \ln \frac{7}{6} + \frac{9}{6} \ln \frac{9}{6} + \frac{11}{6} \ln \frac{11}{6} \right) \right) = 0.633096.
\]

7.

\[
f(x) = e^{2x} \sin 3x
\]
\[
f''(x) = -5e^{2x} \sin (3x) + 12e^{2x} \cos (3x)
\]
\[
f(4)(x) = -119e^{2x} \sin (3x) - 120e^{2x} \cos (3x)
\]
For $\xi \in (0, 2)$ we look for an upper bound for
\[ |f''(\xi)| = \left| e^{2\xi}(-5\sin 3\xi + 12\cos 3\xi) \right| \leq e^4(5 + 12). \]

Or for a better upper bound
\[ |f''(\xi)| = \left| e^{2\xi}(-5\sin 3\xi + 12\cos 3\xi) \right| \leq e^4\sqrt{5^2 + 12^2} = 13e^4. \]

We use the latter one (it’s OK if you use the first one; then you will get a larger $n$/smaller $h$). Similarly
\[ |f^{(4)}(\xi)| = \left| -119e^{2\xi}\sin(3\xi x) - 120e^{2\xi}\cos(3\xi) \right| \]
\[ \leq e^4\sqrt{119^2 + 120^2} \leq 120\sqrt{2}e^4. \]

Thus by the error for the Composite Trapezoidal rule
\[ \left| -\frac{b-a}{12}h^2 f''(\xi) \right| \leq \frac{2}{12}h^213e^4 < 10^{-4} \]
\[ \implies h < 9.1942 \times 10^{-4} \]
\[ \implies n > 2175.3 \text{ (choose 2176).} \]

Note: it’s quite possible that you got a different answer because you used a different upper bound. That’s OK.

Similarly for the Composite Simpson’s rule
\[ \left| -\frac{b-a}{180}h^4 f^{(4)}(\xi) \right| \leq \frac{2}{180}h^4120\sqrt{2}e^4 < 10^{-4} \]
\[ \implies h < 0.033557 \]
\[ \implies n > 57.6 \text{ (choose 58).} \]

For the Composite Midpoint rule
\[ \left| \frac{b-a}{6}h^2 f''(\xi) \right| \leq \frac{2}{6}h^213e^4 < 10^{-4} \]
\[ \implies h < 6.5013 \times 10^{-4} \]
\[ \implies n > 3076.3 \text{ (choose 3077).} \]

17. Use the parametric equations for the ellipse
\[ \begin{cases} x = 3 \cos t \\ y = 2 \sin t \end{cases}, \ t \in (-\pi, \pi]. \]

Then by the arclength formula
\[ L = \int_{-\pi}^{\pi} \sqrt{[x'(t)]^2 + [y'(t)]^2} \, dt = \int_{-\pi}^{\pi} \sqrt{9\sin^2 t + 4\cos^2 t} \, dt \]
\[ = \int_{-\pi}^{\pi} \sqrt{4 + 5\sin^2 t} \, dt \text{ (by } \sin^2 t + \cos^2 t = 1). \]
Next we can use certain integration rule, say, the Composite Trapezoidal rule to approximate the integral (Theorem 4.5)

\[
\int_a^b f(t) dt = \frac{h}{2} [f(a) + 2 \sum_{j=1}^{n-1} f(t_j) + f(b)] - \frac{b-a}{12} h^2 f''(\xi),
\]

(1)

where \([a, b] = [-\pi, \pi]\), \(h = (b-a)/n\), \(f(t) = \sqrt{4 + 5 \sin^2 t}\), \(\xi \in (a, b)\). To decide \(h\)(also \(n\)) we look for an upper bound for the absolute value of the error. Now

\[
f'(t) = \frac{5 \sin t \cos t}{\sqrt{4 + 5 \sin^2 t}} = \frac{5}{2} \frac{\sin 2t}{\sqrt{4 + 5 (1 - \cos 2t)/2}} = \frac{5 \sin 2t}{\sqrt{26 - 10 \cos 2t}} \quad \text{(by trig identities)}
\]

\[
f''(t) = \frac{52 \cos 2t - 10(1 + \cos^2 2t)}{(26 - 10 \cos 2t)^{3/2}}
\]

Thus

\[
\left| f''(\xi) \right| = \left| \frac{52 \cos 2\pi t - 10(1 + \cos^2 2t)}{(26 - 10 \cos 2t)^{3/2}} \right| \leq \frac{52 + 10 \cdot 2}{(26 - 10)^{3/2}} = \frac{9}{8}
\]

\[
\text{(max for the numerator, min for the denominator)}
\]

We want

\[
\left| \frac{b-a}{12} h^2 f''(\xi) \right| = \left| \frac{2\pi}{12} \left( \frac{2\pi}{n} \right)^2 f''(\xi) \right| \leq \frac{3\pi^3}{4n^2} < 10^{-6}
\]

i.e.

\[
n > \sqrt{\frac{3}{4} \pi^3 \times 10^6} = 4822.3.
\]

So we can choose \(n = 4823\). Use the Composite Trapezoidal rule (1). Try the matlab code in problem 1a. The result is 15.865439589, which is the approximation to the length of the ellipse.

You can also try the Composite Midpoint rule with code

```matlab
function integral = cmpmid(a,b,n,f)
    h = (b-a)/(n+2);
    x = [a+h:2*h:b-h];
    integral = 2*h*sum(feval(f,x));
end
```

The \(n\) will be slightly different.

Of course in maple you can use

```maple
f := sqrt(4+5*sin(t)^2);
evalf(int(f,t=-Pi..Pi));
```

to get 15.86543959.

It’s possible that you get a larger upper bound and thus use a larger \(n\).
And we can also use some iterative method, say, adaptive method, and use the tolerance $10^{-6}$. This way we don’t need to compute a specific $n$.

**Sec 4.5: 1a, 3a, 5, 14**

1a, 3a. Try by hand or use the Romberg code

```matlab
function R = romberg(f, a, b, max_k)
    % function R = romberg(f, a, b, max_k)
    % Computes the triangular extrapolation table for Romberg integration
    % using the composite trapezoid rule, starting with h=b-a
    % f: function name (either a name in quotes, or an inline function)
    % a, b: lower and upper limits of integration
    % max_k: the number of extrapolation steps
    % (= number of columns in R, plus one.)
    % max_k=0 will do no extrapolation.
    
    % Example: R = romberg('f',1,1.5,8)
    h = b-a;
    R = zeros(max_k+1);
    R(1,1) = h/2*(feval('f',a)+feval('f',b));
    for k = 2:max_k+1
        R(k,1) = 0;
        for i = 1:2^(k-2)
            R(k,1) = R(k,1)+feval('f',a+(2*i-1)*h/2^(k-1));
        end
        R(k,1) = (R(k-1,1)+h/2^(k-2)*R(k,1))/2;
    end
    for k=2:max_k+1
        for j=k:max_k+1
            R(j,k) = (4^(k-1)*R(j,k-1)-R(j-1,k-1))/(4^(k-1)-1);
        end
    end

    Run in matlab
    R = romberg('f',1,1.5,3)
    we can get $R_{i,j}$, $i, j = 1, 2, 3, 4$

<table>
<thead>
<tr>
<th>i \ j</th>
<th>1</th>
<th>2</th>
<th>3</th>
<th>4</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>0.22807412331084</td>
<td>0.19224530741310</td>
<td>0.19225933731444</td>
<td>0.19225935770658</td>
</tr>
<tr>
<td>2</td>
<td>0.20120251138753</td>
<td>0.19225846044561</td>
<td>0.19225933731444</td>
<td>0.19225935770658</td>
</tr>
<tr>
<td>3</td>
<td>0.19449447318109</td>
<td>0.19225846044561</td>
<td>0.19225933731444</td>
<td>0.19225935770658</td>
</tr>
<tr>
<td>4</td>
<td>0.19281809429207</td>
<td>0.19225930132906</td>
<td>0.19225933731444</td>
<td>0.19225935770658</td>
</tr>
</tbody>
</table>
```

We then have $R_{3,3}$.

For problem 3a, to get the accuracy $10^{-6}$, it’s helpful to look at the details of the Richardson extrapolation and get familiar with the increasing of the error
orders. The error $O(h_k^{2k})$ is decreased in certain pattern (page 209). We want the error $|O(h_k^{2k})| < 10^{-6}$, $h_k = \frac{1}{2^{k-1}}$. Thus approximately we want at least

$$\left( \frac{1}{2^k} \right)^{2k} < 10^{-6}.$$ 

When $k = 3, 4$ respectively, the left hand side $\approx 3.8 \times 10^{-6}, 2.3 \times 10^{-10}$. Now as we need to consider the coefficient in the notation $O(h_k^{2k})$, we can choose for safety $k = 4$. $R_{4, 4}$ is as above.

Finally run until $|R_{k-1,k-1} - R_{k,k}| < 10^{-6}$, or $k = 10$, we can get the approximation 0.1922593577.

5. We can use Romberg integration to get high accuracy. The $f$ values can be used in the formula (4.32) for Romberg integration approximation

$$R_{k,1} = \frac{1}{2} [R_{k-1,1} + h_{k-1} \sum_{i=1}^{2^{k-2}} f(a + (2i - 1)h_k)].$$

Here the $f$ values are used for the following 3 values

- $R_{1,1} = \frac{h_1}{2} [f(a) + f(b)], \quad h_1 = b - a$
- $R_{2,1} = \frac{1}{2} [R_{1,1} + h_1 f(a + h_2)], \quad h_2 = \frac{b - a}{2}$
- $R_{3,1} = \frac{1}{2} \{R_{1,1} + h_2 [f(a + h_3) + f(a + 3h_3)]\}, \quad h_3 = \frac{b - a}{4}$

Thus If the given 5 $f$ values are $f_1, \cdots, f_5$, then use them in

- $R_{1,1} = \frac{h_1}{2} [f_1 + f_5], \quad h_1 = b - a$
- $R_{2,1} = \frac{1}{2} [R_{1,1} + h_1 f_3], \quad h_2 = \frac{b - a}{2}$
- $R_{3,1} = \frac{1}{2} \{R_{2,1} + h_2 [f_2 + f_4]\}, \quad h_3 = \frac{b - a}{4}$

It’s basically the Richardson extrapolation. This way $R_{3,3}$ can have the highest accuracy based on the given data. Now you can compute by hand, or by a code like

```matlab
a = 1; b = 5;
h = b-a;
R = zeros(3);
f = [2.4142 2.6734 2.8974 3.0976 3.2804];
R(1,1) = h/2*(f(1)+f(5));
R(2,1) = (R(1,1)+h*f(3))/2;
R(3,1) = (R(2,1)+h/2*(f(2)+f(4)))/2;
for k=2:3
    for j=k:3
        R(j,k) = (4^(k-1)*R(j,k-1)-R(j-1,k-1))/(4^(k-1)-1);
    end
end
```

end
The corresponding $R_{i,j}(i, j = 1, 2, 3)$ values are

<table>
<thead>
<tr>
<th>$i \setminus j$</th>
<th>1</th>
<th>2</th>
<th>3</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>11.3892</td>
<td></td>
<td></td>
</tr>
<tr>
<td>2</td>
<td>11.4894</td>
<td>11.5228</td>
<td></td>
</tr>
<tr>
<td>3</td>
<td>11.5157</td>
<td>11.5244667</td>
<td>11.524577778</td>
</tr>
</tbody>
</table>

14. When we use Romberg integration, we want the error $|O(h^{2k})| < 10^{-7}$, $h_k = \frac{1}{2^k-1}$. Thus approximately we want at least

$$\left(\frac{1}{2^k-1}\right)^{2k} < 10^{-7}.$$  

When $k = 4, 5$ respectively, the left hand side $\approx 6 \times 10^{-8}$ and $9 \times 10^{-13}$. Now as we need to consider the coefficient in the notation $O(h^{2k})$, we can choose for safety $k = 5$. Then use the Romberg code in problem 1a

```
2/sqrt(pi)*romberg('f',0,1,5)
```
we can get the approximation $erf(1) \approx 0.84270079326867$(this has even higher accuracy than $10^{-7}$). Of course in maple you can verify this:

```
evalf(erf(1.));
```

4.6: 1a, 3a, 5 (for sin(1/x)), 9

1a. Use a simple matlab code

```
function y = s(a,b)
    h = (b-a)/2;
    y = h/3*(f(a)+4*f(a+h)+f(b));
end
```

We have

$$S(a, b) \approx 0.192245307$$
$$S(a, (a + b)/2) \approx 0.039372434$$
$$S((a + b)/2, b) \approx 0.152886026.$$  

The real value is

$$\int_1^{1.5} x^2 \ln x dx = \frac{1}{3} x^3 \ln x|_1^{1.5} - \int_1^{1.5} \frac{1}{3} x^2 dx$$
$$= \frac{1.5^3}{3} \ln 1.5 - \frac{1}{9} (1.5^3 - 1)$$
$$\approx 0.1922593577.$$  

3a. Note: the matlab builtin function “quad” used adaptive Simpson’s quadrature method just as our text. The following is one version of adaptive Simpson’s quadrature code. It will be used for all the problems in this section. *If you get slightly different answers from your own code, it’s OK.*
function [y, errest, iflg, nofun] = adpsim(a, b, tol, fun)
% an implementation of adaptive Simpson quadrature
% user must supply the integrand, fun.
% a,b : limits of integration, tol: absolute error tolerance
% errest: estimated error
% iflg: mode of return, gives number of subintervals where
% maximum number (levmax=10) of bisections is required and value is
% accepted by default. The larger iflg, the less confidence one
% should have in the computed value, y.
% nofun: number of fun evaluations
% initialization
y=0; iflg=0; jflg=0; errest=0; levmax=10;
fsave=zeros(levmax,3); xsave=zeros(levmax,3); simpr=zeros(levmax);
% protect against unreasonable tol
tol2=tol+10*eps;
tol1=tol2*15/(b-a);
x=a:(b-a)/4:b;
for j=1:5
    f(j)=feval(fun,x(j));
end
nofun=5;
level=1;
% level=0 means entire interval is covered, hence finished
while level>0
    % save right half subinterval info
    for k=1:3
        fsave(level,k)=f(k+2);
        xsave(level,k)=x(k+2);
    end
    h=(x(5)-x(1))/4;
simpr(level)=(h/3)*(f(3)+4*f(4)+f(5));
    if jflg<0
        simp1=2*(h/3)*(f(1)+4*f(3)+f(5));
    end
    simp1=(h/3)*(f(1)+4*f(2)+f(3));
simp2=simp1+simpr(level);
diff=abs(simp1-simp2);
    if diff<=tol1*4*h
        % accept approx on current subinterval
        level=level-1;
        jflg=0;
    end
    % we use the conservative strategy of accepting the two-panel composite
    % Simpson rule. The method described on the web page uses y=y+simp1.
    y=y+simp2;
errest=errest+diff/15;
if level<=0
    fprintf('predicted error bound= %g. \n',errest)
    fprintf('number of times integrand was evaluated: %g. \n',nofun)
    fprintf('the computed value of the integral is : %g. \n',y)
    return
end
for j=1:3
    jj=2*j-1;
    f(jj)=fsave(level,j);
    x(jj)=xsave(level,j);
end
else
    level=level+1;
    simp1=simpl;
    if level <= levmax
        jflg=1;
        f(5)=f(3);f(3)=f(2);
        x(5)=x(3);x(3)=x(2);
    else%levmax is reached, accept value computed and continue
        jflg=jflg+1;
        level=level+1;
        jflg=0;
        y=y+simpl2;
        errest=errest+diff/15;
    end
end
for k=1:2
    kk=2*k;
    x(kk)=.5*(x(kk+1)+x(kk-1));
    f(kk)=feval(fun,x(kk));
end
nofun=nofun+2;
end

Define the function in f.m. Run in matlab with:
adpsim(1,3,1e-5,'f')
The result is ≈ 108.5552806.

5. Run the previous code for the function here.
adpsim(0.1,2,1e-3,'f')
The result is ≈ 1.1454993. It’s possible that you may get slightly different answers.
The graph is as follows.
9. Define the functions in c.m and s.m. Call the code in problem 1a.
   fprintf(' t c(t) s(t)n');
   for t = 0.1:0.1:1
     fprintf('%16.12g %16.12f %16.12g
', t, ... 
            adpsim(0,t,1e-3,'c'), adpsim(0,t,1e-3,'s'));
   end

Results (you may get slightly different ones):

<table>
<thead>
<tr>
<th>t</th>
<th>c(t)</th>
<th>s(t)</th>
</tr>
</thead>
<tbody>
<tr>
<td>0.1</td>
<td>0.099997526203</td>
<td>0.000523589386911</td>
</tr>
<tr>
<td>0.2</td>
<td>0.199920852789</td>
<td>0.00418758860931</td>
</tr>
<tr>
<td>0.3</td>
<td>0.299399445796</td>
<td>0.0141166481358</td>
</tr>
<tr>
<td>0.4</td>
<td>0.397474592626</td>
<td>0.0333568374407</td>
</tr>
<tr>
<td>0.5</td>
<td>0.492327199200</td>
<td>0.064720316156</td>
</tr>
<tr>
<td>0.6</td>
<td>0.581060998724</td>
<td>0.110498453372</td>
</tr>
<tr>
<td>0.7</td>
<td>0.659604949810</td>
<td>0.172021508022</td>
</tr>
<tr>
<td>0.8</td>
<td>0.722827448864</td>
<td>0.249078463858</td>
</tr>
<tr>
<td>0.9</td>
<td>0.764971727407</td>
<td>0.339270624009</td>
</tr>
<tr>
<td>1</td>
<td>0.780505925697</td>
<td>0.438216495327</td>
</tr>
</tbody>
</table>