1. Given a function $f(x)$, use Taylor approximations to derive a second order centered approximation to $f''(x_0)$ is given by

$$f''(x_0) = af(x_0 - h) + bf(x_0) + cf(x_0 + h) + O(h^2).$$

(a) What are $a$, $b$ and $c$?

(b) What is the precise form of the error term?

(c) Using the formula approximate $f''(\pi/4)$ where $f(x) = \cos x$ for $h = 1/(2^p)$ for $p = 1:15$. Form a table with columns giving $h$, the approximation, absolute error and absolute error divided by $h^2$. For each indicate to which values they are converging.

(d) Verify that the last column appears to be converging to a value derived using the error term.

**ANS:** We have the following Taylor expansions:

$$f(x_0 - h) = f(x_j) - f'(x_0)h + \frac{1}{2}f''(x_0)h^2 - \frac{1}{6}f'''(x_0)h^3 + \frac{1}{24}f^{(4)}(c^-_x)h^4, \quad c^-_x \in (x_0 - h, x_0)$$

$$f(x_0) = f(x_0)$$

$$f(x_0 + h) = f(x_j) + f'(x_0)h + \frac{1}{2}f''(x_0)h^2 + \frac{1}{6}f'''(x_0)h^3 + \frac{1}{24}f^{(4)}(c^+_x)h^4, \quad c^+_x \in (x_0, x_0 + h)$$

**For (a):** Form the linear combination $af(x_0 - h) + bf(x_0) + cf(x_0 + h) = (a + b + c)f(x_0) + (-ah + ch)f'(x_0) + \frac{1}{2}(ah^2 + ch^2)f''(x_0) + \ldots$

Choose the three unknowns $a$, $b$, and $c$ so that $f''(x_0)$ is multiplied by 1, and $f(x_0)$ and $f'(x_0)$ are multiplied by 0, or

$$a + b + c = 0$$

$$-ah + ch = 0 \quad \Rightarrow \quad a = c = \frac{1}{h^2}, \quad b = -\frac{2}{h^2}.$$ 

So the approximation is

$$f''(x_0) \approx \frac{f(x_0 - h) - 2f(x_0) + f(x_0 + h)}{h^2}.$$ 

**NOTE:** Since $a = c$ the coefficient for the $f'''(x_0)$ term is 0.

**For (b):** The error term is then

$$\frac{1}{24}h^4(af^{(4)}(c^-_x) + cf^{(4)}(c^+_x)) = \frac{1}{24}h^2(f^{(4)}(c^-_x) + f^{(4)}(c^+_x))$$
and assuming that $f^{(4)}(x)$ is continuous, $f^{(4)}(c_x^-) + f^{(4)}(c_x^+) = 2f^{(4)}(c_x^*)$ where $c_x^* \in (c_x^-, c_x^+)$ function. Thus, the error term can be written as

$$
\frac{1}{24} h^2 (2f^{(4)}(c_x^*)) = \frac{h^2}{12} f^{(4)}(c_x^*) = O(h^2).
$$

For (c): As $h \to 0$ the error, the second column, initially decreases, along with the third column, error/h², to the constant predicted by the error term. Then the effect of roundoff emerges and the error is seen to grow.

<table>
<thead>
<tr>
<th>h</th>
<th>err</th>
<th>err/h²</th>
</tr>
</thead>
<tbody>
<tr>
<td>5.0000e-01</td>
<td>1.4609e-02</td>
<td>5.8437e-02</td>
</tr>
<tr>
<td>2.5000e-01</td>
<td>3.6752e-03</td>
<td>5.8803e-02</td>
</tr>
<tr>
<td>1.2500e-01</td>
<td>9.2023e-04</td>
<td>5.8895e-02</td>
</tr>
<tr>
<td>6.2500e-02</td>
<td>2.3015e-04</td>
<td>5.8918e-02</td>
</tr>
<tr>
<td>3.1250e-02</td>
<td>5.7543e-05</td>
<td>5.8924e-02</td>
</tr>
<tr>
<td>1.5625e-02</td>
<td>1.4386e-05</td>
<td>5.8925e-02</td>
</tr>
<tr>
<td>7.8125e-03</td>
<td>3.5965e-06</td>
<td>5.8925e-02</td>
</tr>
<tr>
<td>3.9062e-03</td>
<td>8.9913e-07</td>
<td>5.8925e-02</td>
</tr>
<tr>
<td>1.9531e-03</td>
<td>2.476e-07</td>
<td>5.8920e-02</td>
</tr>
<tr>
<td>9.7656e-04</td>
<td>5.6107e-08</td>
<td>5.8832e-02</td>
</tr>
<tr>
<td>4.8828e-04</td>
<td>1.3499e-08</td>
<td>5.6617e-02</td>
</tr>
<tr>
<td>2.4414e-04</td>
<td>2.7884e-09</td>
<td>4.6781e-02</td>
</tr>
<tr>
<td>1.2207e-04</td>
<td>2.7995e-09</td>
<td>1.8787e-01</td>
</tr>
<tr>
<td>6.1035e-05</td>
<td>1.7701e-08</td>
<td>4.7515e+00</td>
</tr>
<tr>
<td>3.0518e-05</td>
<td>4.7503e-08</td>
<td>5.1006e+01</td>
</tr>
</tbody>
</table>

For (d): As $h \to 0$ so does the error until roundoff begins to creep into the calculation, which can be seen in the last few entries since the error/h² column approaches

$$
|\frac{1}{12} f^{(4)}(\pi/4)| = |\cos (\pi/4)| \approx 5.89256e-02,
$$

but then starts to move away at the last value of $h$. 
2. The floating point representation of a number is \( x = \pm (0.b_1b_2 \ldots b_n) \beta \times \beta^e \), where \( b_1 \neq 0, -M \leq e \leq M \). Suppose \( \beta = 2, \ n = 7, \) and \( M = 4 \).

(a) Find the smallest positive \( (x_{\text{min}}) \) and largest positive \( (x_{\text{max}}) \) floating point numbers that can be represented. Give the answers in decimal form (base 10).

(b) What is the machine epsilon, \( \epsilon_{\text{ps}} \), of this number system?

(c) Find the floating point number in this system that is closest to \( x = 4\pi \).

\textbf{ANS: For (a), we have}

\[ x_{\text{min}} = + (0.1000000)_2 \times 2^{-4} = 2^{-5} = \frac{1}{32} = 0.03125, \]

and

\[ x_{\text{max}} = + (0.1111111)_2 \times 2^4 = (127/128)_{10} \times 2^4 = \frac{127}{8} = 15.875. \]

\textbf{For (b), note that}

\[ 1 + \epsilon_{\text{ps}} = + (0.1000001)_2 \times 2^1, \]

or

\[ \epsilon_{\text{ps}} = + (0.0000001)_2 \times 2^{-6} = 2^{-6} = 0.015625, \]

\textbf{For (c), the closest number to 4\pi in this system is}

\[ x_{4\pi} = + (0.1100101)_2 \times 2^4 = \left( \frac{1}{2} + \frac{1}{4} + \frac{1}{32} + \frac{1}{128} \right)_{10} \times 2^4 = 12.625. \]
3. Recall that the *machine epsilon* of a computer is the smallest positive floating point number \( eps \) such that \( fli(1 + eps) > 1 \). We can determine \( eps \) on a given machine, for a given floating point precision, by evaluating the expression

\[
(1 + x) - 1
\]

for decreasing values of \( x \). The smallest representable positive \( x \) for which (*) is nonzero is \( eps \). On a binary machine it is enough to consider the sequence \( x_n = 2^{-n} \) for \( n = 1, 2, \ldots \) (Why?).

Write a MATLAB code to determine \( eps \) on the machine you are using, and compare it with the value of \( eps \) in MATLAB (type ‘eps’ in MATLAB to see this value). What is the relationship between the two. (Note: you may find it useful to first issue the MATLAB command ‘format long e’ so that you are sure of when an expression computes identically to 0). Include a copy of your code.

**ANS:** Here is the code:

```matlab
function [y,n] = myeps

% MYEPS determines the machine epsilon. It returns the
% value, and set corresponding power of 2

n=0; x=1;
while ((1+x) > 1)
    n=n-1; x=x/2;
end
y=2*x; % while loop exited with x=eps/2, so recover proper
n=n+1; % value, and set corresponding power of 2

% result

Executing this function gives

```MATLAB
>> [y,n] = myeps

y =

2.22044604925031e-16

n =

-52
```

showing the \( y = 2^{-52} \). Further, we see

```MATLAB
>> eps-y

ans =

0
```

and we see that the \( \epsilon_M \) we found (returned as \( y \)) is equal to MATLAB’s \( eps \).
4. Consider evaluating the integrals

\[ y_n = \int_0^1 \frac{x^n}{x + 10} \, dx \]

for \( n = 1, 2, \ldots, 30 \).

(a) Show analytically that \( y_n + 10y_{n-1} = 1/n \).

(b) Show that \( y_0 = \log 11 - \log 10 \) and then using this \( y_0 \) (not \( \log 11/10 \)) compute the recursion

\[ y_n = \frac{1}{n} - 10y_{n-1} \]

and then use this recursion to numerically generate \( y_1 \) through \( y_{30} \).

(c) Show for \( n \geq 0 \) that \( 0 \leq y_n \leq 1 \), and discuss the results in (b) in light of this.

**ANS:** For (a), we have

\[ y_n + 10y_{n-1} = \int_0^1 \frac{x^n}{x + 10} \, dx + 10 \int_0^1 \frac{x^{n-1}}{x + 10} \, dx = \int_0^1 x^{n-1} \, dx = \frac{1}{n}, \]

and for (b),

\[ y_0 = \int_0^1 \frac{x^0}{x + 10} \, dx = \int_0^1 \frac{1}{x + 10} \, dx = \ln 11 - \ln 10. \]

Below is the MATLAB code and \( y_n, n = 0, \ldots, 30 \).

```matlab
y = log(11)-log(10);
disp(' ')
disp('n    y_n')
disp('----------');
for n = 1:30
    y = 1/n - 10*y;
    disp([num2str(n),', ',num2str(y)])
end
```

<table>
<thead>
<tr>
<th>n</th>
<th>y_n</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>0.046898</td>
</tr>
<tr>
<td>2</td>
<td>0.031018</td>
</tr>
<tr>
<td>3</td>
<td>0.023154</td>
</tr>
<tr>
<td>4</td>
<td>0.018465</td>
</tr>
<tr>
<td>5</td>
<td>0.015353</td>
</tr>
<tr>
<td>6</td>
<td>0.013138</td>
</tr>
<tr>
<td>7</td>
<td>0.011481</td>
</tr>
<tr>
<td>8</td>
<td>0.010194</td>
</tr>
<tr>
<td>9</td>
<td>0.0091673</td>
</tr>
</tbody>
</table>
For (c), note that $0 \leq \frac{x^n}{(x+10)} \leq 1$, hence $y_n$ is bounded by $(1 - 0) * 1 = 1$. However the iterates start to diverge/blow up around the 16th iterate and begin to grow rapidly. Why? Note that in the recursion formula $y_n = \frac{1}{n} - 10y_{n-1}$ the previous iterate $y_{n-1}$ is multiplied by 10, so any error in that iterate is amplified by an order of magnitude. The true limit is 0.