This project has two objects. The first is to give you some experience with elementary Matlab programming. The second is to explore some of the many ways numerical computations can go wrong (or right). There are three subprojects treating 1) serious errors without cancellation, 2) convergence criteria for sequences, and 3) computing the hyperbolic function sinh.

Repeated square roots.

If you enter the number 2 onto a hand calculator and repeatedly take square roots, you will eventually end up with the number 1. If you try to recover the original number 2 by repeated squaring, you will get nowhere.

Write a function

```matlab
function repeated_sqrt(x, n)
    % This function computes the nth root of x and the nth power of the result.
    y = x; z = x;
    for i = 1:n
        y = sqrt(y);
        z = y^2;
        disp([x, y, z])
    end
    disp([abs((x-z)/x), abs(y-1), abs((x-z)/x)*abs(y-1)])
end
```

Run the following script with your function.

```matlab
for i=1:60
    repeated_sqrt(2, i)
end
```

1. What is invariant in the relation between the relative error in z as an approximation to x and in the relative error in y as an approximation to one. In plain words what does this relation mean?

2. At some point near the end of the run, the output changes dramatically. Why?

You should hand in a listing of your function, the output from the script (the Matlab command `diary` will be useful here), and the answers to the above questions.
Testing for convergence

Many algorithms generate a sequence $x_0, x_1, x_2, \ldots$ of increasingly accurate approximations to a desired solution $x^*$. The problem is to determining when to stop the iteration. One procedure is to select a convergence criterion $\epsilon$ and return $x_{i+1}$ when

$$|x_{i+1} - x_i| \leq \epsilon$$  (1)

But if convergence is slow enough, this can result in an approximation to $x^*$ with error greater than $\epsilon$. This part of the project investigates this phenomenon.

Let $0 < \alpha < 1$. The recursion

$$x_{i+1} = 1 + \alpha(x_i - 1)$$  (2)

 generates a sequence that converges to $x^* = 1$. Specifically,

$$x_{i+1} - 1 = \alpha(x_i - 1)$$  (3)

so that each iteration reduces the error $|x_i - x^*|$ by a factor of $\alpha$. In particular if $\alpha$ is near one the convergence is slow.

Write a function

```matlab
function [xconv, index] = conv_test(x0, alpha, epsilon)
```

That computes the recursion (2) until (1) is satisfied and returns the value of $x_{i+1}$ in $\text{xconv}$ and $i + 1$ in $\text{index}$. This function should use no arrays.

Write a script that evaluates

```matlab
[xc(k), idx(k)] = conv_test(2, alpha(k), 1e-4);
err(k) = xc(k) - 1;
disp([xc(k), idx(k), err(k)]);
```

for $k = 1, \ldots, 10$ and $\text{alpha}(k) = 1 - 2^{-k}$. The script should also give a loglog plot of $1-\alpha(k)$ vs $\text{idx}(k)$.

For $1 - \alpha \leq 0.1$ the plot looks like a straight line with negative slope one. Here is a sketch of an argument why this happens. We have the expression

$$x_i = 1 + \alpha^i(x_0 - x^*) = 1 + \alpha^i.$$  (4)

Hence

$$|x_{i+1} - x_i| = \alpha^i(1 - \alpha).$$

Our convergence criterion demands we stop when

$$\alpha^i(1 - \alpha) \cong \epsilon.$$  (5)
Hence the error for stopping with \( x_{i+1} \) is
\[
e_{i+1} = x_{i+1} - 1 = \alpha^{i+1} \approx \frac{\alpha \epsilon}{1 - \alpha} \approx \frac{\epsilon}{1 - \alpha}.
\]
Taking logs gives the answer.

Give a detailed derivation of (4). Justify the approximations in (6). Show precisely how taking logs gives the answer. Also account for the fact that the linearity fails slightly when \( 1 - \alpha \leq 0.1 \).

For this part hand in listings of your mfiles, the plot, and your written argument for the linearity of the plot.

The series approximation of the hyperbolic sign

How do you compute
\[
\sinh x = \frac{e^x - e^{-x}}{2} = \frac{e^x - 1/e_x}{2}.
\]
if you do not have a reliable library routine? Evaluating it from the formula will not work when \( x \) is small, owing to cancellation in forming \( e^x - e^{-x} \). On the other hand when \( |x| \geq 1 \), the formula works well. This suggests that we use the power series
\[
\sinh(x) = x + \frac{x^3}{3!} + \frac{x^5}{5!} + \cdots = x \left( 1 + \frac{x^2}{3!} + \frac{x^4}{5!} + \cdots \right),
\]
which converges quickly for \( |x| \leq 1 \), and use the formula \texttt{sinhform} when \( x > 1 \). We will focus on the power series.

Write a function
\[
\text{function } [y, \text{errest}, \text{termnum}] = \text{series_sinh}(x)
\]
to evaluate \( \sinh \) using the second formula in (8). Denote the terms in the series in parentheses by \( T_1, T_3, T_5, \ldots \) and use the fact that \( T_{i+2} \) can be easily generated from \( T_i \) in your evaluation. Stop summing the series for the first \( k \) for which
\[
|T_i| \leq |T_1 + T_3 + \cdots + T_{i-2}|\epsilon_M.
\]
Here \( \epsilon_M \) is the rounding unit, which in Matlab is the variable \texttt{eps}. The function should return
\[
x(T_1 + T_3 + \cdots + T_{i-2}) \quad \text{for } y
\]
\[
x|T_i| \quad \text{for } \text{errest}
\]
\[
i \quad \text{for } \text{termnum}
\]
The return arguments \texttt{errest} and \texttt{termnum} are optional. Their presence can be detected by interrogating the Matlab variable \texttt{nargout}. 3
For a given integer $n$ let $h = 1/n$ and for $i=1:n$ let $x(i) = ih$. Write a script that computes

$$[y(i), \text{est}(i), \text{tn}(i)] = \text{series_sinh}(x(i)), \quad i=1:n$$

In addition compute the relative error $\text{relerr}(i)$ in $y(i)$ as an approximation to $\sinh(x(i))$ as computed by the Matlab function $\sinh$. Plot on the same plot $x$ vs $\text{relerr}$ (connected by lines) and $x$ vs $\text{est}$ (as the symbol $\ast$). Run your script for $n = 100$.

Answer the following two questions.

1. Why is $\text{est}(i)$ generally smaller than $\text{relerr}(i)$?

2. Why the discontinuities in the plot of $\text{relerr}(i)$? [Hint: Try $\text{hold}, \text{plot}(x, 1e-17*\text{tn})$.]

Submit your programing listings, plot, and answers.