1 Problem 1

Derive the Richardson extrapolation theme for a series $S_N$ that converges to $L$ that uses $S_{2N}$, $S_{4N}$ and so forth. In other words, if you think of $S_N$ as a partial sum for an infinite series, each step uses twice as many steps as the one before.

1.1 Part a.

Derive a linear scheme for $S_N = L + \frac{A}{N} + ...$

1.1.1 Solution

We have\[ S_N = L + \frac{A}{N} + ... \]
\[ S_{2N} = L + \frac{A}{2N} + ... \]

so\[ L \approx 2S_{2N} - S_N + ... \]

which suggests\[ T_N = 2S_{2N} - S_N \]

1.2 Part b.

Derive a quadratic scheme for $S_N = L + \frac{B}{N^2} + ...$

1.2.1 Solution

We have\[ S_N = L + \frac{B}{N^2} + ... \]
\[ S_{2N} = L + \frac{B}{4N^2} + ... \]

so\[ L \approx \frac{4S_{2N} - S_N}{3} + ... \]
which suggests
\[ T_N = \frac{4S_{2N} - S_N}{3} \]

1.3 Part c.
Derive a combined linear and quadratic scheme for \( S_N = L + \frac{A}{N} + \frac{B}{N^2} + \ldots \)

1.3.1 Solution
We have
\[
\begin{align*}
S_N &= L + \frac{A}{N} + \frac{B}{N^2} + \ldots \\
S_{2N} &= L + \frac{A}{2N} + \frac{B}{4N^2} + \ldots \\
S_{4N} &= L + \frac{A}{4N} + \frac{B}{16N^2} + \ldots
\end{align*}
\]

so
\[
\begin{bmatrix}
1 & \frac{1}{N} & \frac{1}{N} & \ldots \\
1 & \frac{2}{N} & \frac{2}{N} & \ldots \\
1 & \frac{4}{N} & \frac{4}{N} & \ldots
\end{bmatrix}
\begin{bmatrix}
L \\
A \\
B
\end{bmatrix}
= 
\begin{bmatrix}
S_N \\
S_{2N} \\
S_{4N}
\end{bmatrix}
\]

Solving
\[
\begin{bmatrix}
L \\
A \\
B
\end{bmatrix}
= 
\begin{bmatrix}
1 & \frac{1}{N} & \frac{1}{N} & \ldots \\
1 & \frac{2}{N} & \frac{2}{N} & \ldots \\
1 & \frac{4}{N} & \frac{4}{N} & \ldots
\end{bmatrix}^{-1}
\begin{bmatrix}
S_N \\
S_{2N} \\
S_{4N}
\end{bmatrix}
= 
\begin{bmatrix}
\frac{8}{3}S_{4N} - 2S_{2N} + \frac{1}{3}S_N \\
-2NS_N + 10NS_{2N} - 8NS_{4N} \\
\frac{8}{3}N^2S_N - 8N^2S_{2N} + \frac{16}{3}N^2S_{4N}
\end{bmatrix}
\]

We are only interested in \( L \)
\[ L = \frac{8}{3}S_{4N} - 2S_{2N} + \frac{1}{3}S_N \]

which suggests
\[ T_N = \frac{8}{3}S_{4N} - 2S_{2N} + \frac{1}{3}S_N \]

1.4 Part d.
Substitute Part a. into Part b. and show that you get the answer in Part c. In other words, performing first order (linear) scheme followed by a second order (quadratic) scheme is equivalent to doing both simultaneously.

1.4.1 Solution
Let the result of the linear Richardson extrapolation be \( U_N \)
\[ U_N = 2S_{2N} - S_N \]
Then

\[ T_N = \frac{4}{3} U_{2N} - \frac{1}{3} U_N \]

\[ = \frac{4}{3} (2S_{4N} - S_{2N}) - \frac{1}{3} (2S_{2N} - S_N) \]

\[ = \frac{8}{3} S_{4N} - 2S_{2N} + \frac{1}{3} S_N \]

\section{Problem 2}

\subsection{Part a.}

Program the fast series

\[ \pi = \sqrt{\frac{16}{\sum_{n=1}^{\infty} \frac{1}{(2n-1)^2 (2n+1)^2}} + 8} \]

and use a log-log plot to determine the order of convergence of this series. You should get third order convergence. Turn in the plot and indicate by hand how you determined that the order is 3. You can use \( S_{10} \), \( S_{100} \), \( S_{1000} \), etc. or \( S_2 \), \( S_4 \), \( S_8 \), etc. In the second case, you should probably compute more partial sums.

\subsection{Solution}

Should see a straight line on a log-log chart with a slope of \(-3\).

\subsection{Part b.}

Derive the third order Richardson extrapolation for your sequence of partial sums.

\subsection{Solution}

We have

\[ S_N = L + \frac{B}{N^3} + ... \]

\[ S_{2N} = L + \frac{B}{8N^3} + ... \]

so

\[ L \approx \frac{8S_{2N} - S_N}{7} \]

\subsection{Part c.}

Apply the scheme you derived in Part b. to the series you obtained in Part a. Demonstrate that the new series converges with order 4.

Turn in all code.
2.3.1 Solution

Should see a straight line on a log-log chart with a slope of $-4$.

3 Problem 3

Write a semi-vectorized version of piSum:

```matlab
function s = piSumSemiVectorized(N)
    The idea is to break up $N$ into pieces of no more than, say, 1000000 terms.
    Add up all the terms in a single piece with vectorized code and add up the pieces
    together with a loop. Make sure to add the terms and the pieces backwards.
    Your code must work properly with $N$ that is not a clean multiple of 1000000,
    for example, $N = 6000123$.

    CHUNK = 1000000;
    STEPS = floor(N/CHUNK);
    s = 0;
    for ii = 1:STEPS
        s = s + sum(((ii - 1)*CHUNK+1:ii*CHUNK).^(-2));
    end
    s = s + sum((STEPS*CHUNK+1:N).^(-2));
    s = sqrt(6*s);
    The key is: no loop from 1 to N!!!
```

4 Problem 4

What answer does the last command produce and why? Use "help" for function
you don’t know.

4.1 Part a.

```matlab
>> M = eye(500);
    >> sum(sum(M(2:2:end, 2:2:end)))
```

4.1.1 Solution

250 – sum of entries that are in both even rows and even cols.

4.2 Part b.

```matlab
>> M = eye(500);
    >> sum(sum(M(2:2:end, 1:2:end)))
```
4.2.1 Solution
0 – sum of entries that are in even rows but odd cols.

4.3 Part c.

\[
\gg 10.^[1 2 3 4] - [10 100 1000 10000]
\]

4.3.1 Solution

\[
[0 0 0 0]
\]