Consider having a polynomial

\[ p(x) = a_0 + a_1x + a_2x^2 + \cdots + a_nx^n \]

which you need to evaluate for many values of \( x \). How do you evaluate it? This may seem a strange question, but the answer is not as obvious as you might think.

The standard way, written in a loose algorithmic format:

\[
\begin{align*}
poly &= a_0 \\
\text{for } j = 1 : n \\
\quad poly &= poly + a_jx^j \\
\text{end}
\end{align*}
\]
To compare the costs of different numerical methods, we do an operations count, and then we compare these for the competing methods. Above, the counts are as follows:

*additions*: \( n \)

*multiplications*: \[ 1 + 2 + 3 + \cdots + n = \frac{n(n + 1)}{2} \]

This assumes each term \( a_jx^j \) is computed independently of the remaining terms in the polynomial.
Next, do the terms $x^j$ recursively:

$$x^j = x \cdot x^{j-1}$$

Then to compute $\{x^2, x^3, ..., x^n\}$ will cost $n - 1$ multiplications. Our algorithm becomes

$$\text{poly} = a_0 + a_1 x$$

$$\text{power} = x$$

for $j = 2 : n$

$$\text{power} = x \cdot \text{power}$$

$$\text{poly} = \text{poly} + a_j \cdot \text{power}$$

end

The total operations cost is

**additions**: $n$

**multiplications**: $1 + 2(n - 1) = 2n - 1$

When $n$ is not small, this is much less than for the first method of evaluating $p(x)$. E.g., with $n = 20$, the first method has 210 multiplications, whereas the second has 39 multiplications.
We now considered **nested multiplication**. As examples of particular degrees, write

\[
\begin{align*}
n = 2 : & \quad p(x) = a_0 + a_1 x + a_2 x^2 \\
& \quad = a_0 + x(a_1 + a_2 x) \\
n = 3 : & \quad p(x) = a_0 + a_1 x + a_2 x^2 + a_3 x^3 \\
& \quad = a_0 + x(a_1 + x(a_2 + a_3 x)) \\
n = 4 : & \quad p(x) = a_0 + a_1 x + a_2 x^2 + a_3 x^3 + a_4 x^4 \\
& \quad = a_0 + x(a_1 + x(a_2 + x(a_3 + a_4 x)))
\end{align*}
\]

These contain, respectively, 2, 3, and 4 multiplications. This is less than the preceding method, which would have need 3, 5, and 7 multiplications, respectively.
For the general case, write

\[ p(x) = a_0 + a_1 x + a_2 x^2 + a_3 x^3 + \cdots + a_n x^n \]

as

\[ p(x) = a_0 + x \left( a_1 + x \left( a_2 + \cdots + x \left( a_{n-1} + a_n x \right) \cdots \right) \right) \]

This requires \( n \) multiplications, which is only about half that for the preceding method. For an algorithm, write

\[
\begin{align*}
poly &= a_n \\
for \ j &= n - 1 : -1 : 0 \\
\ & \ \\
& poly = a_j + x \cdot poly \\
end
\end{align*}
\]

With all three methods, the number of additions is \( n \); but the number of multiplications can be dramatically different for large \( n \).
Imagine we are evaluating the polynomial

\[ p(x) = a_0 + a_1 x + a_2 x^2 + \cdots + a_n x^n \]

at a point \( x = z \). Thus with nested multiplication

\[ p(z) = a_0 + z (a_1 + z (a_2 + \cdots + z (a_{n-1} + a_n z) \cdots )) \]

We can write this as the following sequence of operations:

\[ b_n = a_n \]
\[ b_{n-1} = a_{n-1} + zb_n \]
\[ b_{n-2} = a_{n-2} + zb_{n-1} \]
\[ \vdots \]
\[ b_0 = a_0 + zb_1 \]

The quantities \( b_{n-1}, ..., b_0 \) are simply the quantities in parentheses, starting from the inner most and working outward.
Introduce
\[ q(x) = b_1 + b_2x + b_3x^2 + \cdots + b_{n-1}x^{n-1} \]

Claim:
\[ p(x) = b_0 + (x - z)q(x) \] (*)

Proof: Simply expand
\[ b_0 + (x - z) \left( b_1 + b_2x + b_3x^2 + \cdots + b_{n-1}x^{n-1} \right) \]
and use the fact that
\[ zb_j = b_{j-1} - a_{j-1}, \quad j = 1, \ldots, n \]

With this result (*), we have
\[ \frac{p(x)}{x - z} = \frac{b_0}{x - z} + q(x) \]

Thus \( q(x) \) is the quotient when dividing \( p(x) \) by \( x - z \), and \( b_0 \) is the remainder.
If $z$ is a zero of $p(x)$, then $b_0 = 0$; and then

$$p(x) = (x - z)q(x)$$

For the remaining roots of $p(x)$, we can concentrate on finding those of $q(x)$. In rootfinding for polynomials, this process of reducing the size of the problem is called deflation. Another consequence of (*) is the following. Form the derivative of (*) with respect to $x$, obtaining

$$p'(x) = (x - z)q'(x) + q(x)$$

$$p'(z) = q(z)$$

Thus to evaluate $p(x)$ and $p'(x)$ simultaneously at $x = z$, we can use nested multiplication for $p(z)$ and we can use the intermediate steps of this to also evaluate $p'(z)$. This is useful when doing rootfinding problems for polynomials by means of Newton’s method.
Define

\[ SF(x) = \frac{1}{x} \int_{0}^{x} \frac{\sin t}{t} dt, \quad x \neq 0 \]

We use Taylor polynomials to approximate this function, to obtain a way to compute it with accuracy and simplicity.
As an example, begin with the degree 3 Taylor approximation to \( \sin t \), expanded about \( t = 0 \):

\[
\sin t = t - \frac{1}{6} t^3 + \frac{1}{120} t^5 \cos c_t
\]

with \( c_t \) between 0 and \( t \). Then

\[
\frac{\sin t}{t} = 1 - \frac{1}{6} t^2 + \frac{1}{120} t^4 \cos c_t
\]

\[
\int_0^x \frac{\sin t}{t} \, dt = \int_0^x \left[ 1 - \frac{1}{6} t^2 + \frac{1}{120} t^4 \cos c_t \right] \, dt
\]

\[
= x - \frac{1}{18} x^3 + \frac{1}{120} \int_0^x t^4 \cos c_t \, dt
\]

\[
\frac{1}{x} \int_0^x \frac{\sin t}{t} \, dt = 1 - \frac{1}{18} x^2 + R_2(x)
\]

\[
R_2(x) = \frac{1}{120} x \int_0^x t^4 \cos c_t \, dt
\]
How large is the error in the approximation

\[ SF(x) \approx 1 - \frac{1}{18} x^2 \]

on the interval \([-1, 1]\)? Since \(|\cos c_t| \leq 1\), we have for \(x > 0\) that

\[ 0 \leq R_2(x) \leq \frac{1}{120} \frac{1}{x} \int_0^x t^4 \, dt = \frac{1}{600} x^4 \]

and the same result can be shown for \(x < 0\). Then for \(|x| \leq 1\), we have

\[ 0 \leq R_2(x) \leq \frac{1}{600} \]

To obtain a more accurate approximation, we can proceed exactly as above, but simply use a higher degree approximation to \(\sin t\).
In the book we consider finding a Taylor polynomial approximation to $SF(x)$ with its error satisfying

$$|R_8(x)| \leq 5 \times 10^{-9}, \quad |x| \leq 1$$

A MATLAB program, `plot_sint.m`, implementing this approximation is given in the text and is made available to the class.
Begin with a Taylor series for \( \sin t \),

\[
\sin t = t - \frac{t^3}{3!} + \frac{t^5}{5!} - \cdots + (-1)^{n-1} \frac{t^{2n-1}}{(2n-1)!} \\
+ (-1)^n \frac{t^{2n+1}}{(2n+1)!} \cos(c_t)
\]

with \( c_t \) between 0 and \( t \). Then write

\[
SF(x) = \frac{1}{x} \int_0^x \left[ 1 - \frac{t^2}{3!} + \frac{t^4}{5!} - \cdots + (-1)^{n-1} \frac{t^{2n-2}}{(2n-1)!} \right] dt \\
+ R_{2n-2}(x)
= 1 - \frac{x^2}{3!3} + \frac{x^4}{5!5} - \cdots + (-1)^{n-1} \frac{x^{2n-2}}{(2n-1)!(2n-1)} \\
+ R_{2n-2}(x)
\]

\[
R_{2n-2}(x) = \frac{1}{x} \int_0^x (-1)^n \frac{t^{2n}}{(2n+1)!} \cos(c_t) dt
\]
\[ R_{2n-2}(x) = \frac{1}{x} \int_0^x (-1)^n \frac{t^{2n}}{(2n+1)!} \cos(c_t) \, dt \]

To simplify matters, let \( x > 0 \). Since \( |\cos(c_t)| \leq 1 \),

\[ |R_{2n-2}(x)| \leq \frac{1}{x} \int_0^x \frac{t^{2n}}{(2n+1)!} \, dt = \frac{x^{2n}}{(2n+1)!(2n+1)} \]

It is easy to see that this bound is also valid for \( x < 0 \). As required, choose the degree so that

\[ |R_{2n-2}(x)| \leq 5 \times 10^{-9} \]

From the error bound,

\[ \max_{|x| \leq 1} |R_{2n-2}(x)| \leq \frac{1}{(2n+1)!(2n+1)} \]

Choose \( n \) so that this upper bound is itself bounded by \( 5 \times 10^{-9} \). This is true if \( 2n + 1 \geq 11 \), i.e. \( n \geq 5 \).
The polynomial is

\[ p(x) = 1 - \frac{x^2}{3!3} + \frac{x^4}{5!5} - \frac{x^6}{7!7} + \frac{x^8}{9!9}, \quad -1 \leq x \leq 1 \]

and

\[ |SF(x) - p(x)| \leq 5 \times 10^{-9}, \quad |x| \leq 1 \]

To evaluate it efficiently, we set \( u = x^2 \) and evaluate

\[ g(u) = 1 - \frac{u}{18} + \frac{u^2}{600} - \frac{u^3}{35280} + \frac{u^4}{3265920} \]

After the evaluation of the coefficients (done once), the total number of arithmetic evaluations is 4 additions and 5 multiplications to evaluate \( p(x) \) for each value of \( x \).