

## 4. Evaluating Polynomials

We used the infinite series expansion for  $\exp(x)$  to compute  $\exp(5.0)$  accurately. A bound for the *truncation error* of the series was used to determine the number of terms necessary to be summed to compute  $\exp(5.0)$  to a required accuracy. If the same series is used to compute  $\exp(-9.1)$ , for example, although a certain number of terms of the series should be sufficient to obtain a required accuracy theoretically, the computed value may not have enough correct significant digits. For negative values of  $x$ , the alternate terms in the series are of opposite signs and consequently **loss of significant digits** will occur as the cumulative sum is formed. Loss of significant digits takes place when two numbers of similar magnitudes of the same sign are subtracted. One way to avoid this when alternating terms of a series are of opposite signs, is to form the differences before summing them up. For example in summing the series  $a_0 - a_1x + a_2x^2 - a_3x^3 + \dots$  form the successive differences  $a_0 - a_1x$ ,  $a_2x^2 - a_3x^3$  first.

A better way to compute  $\exp(-x)$  is to use the relation  $\exp(-x) = 1/\exp(x)$ . In practice infinite series are hardly used for the purpose of calculating functions. Instead, various function approximation methods are utilized to derive *finite series* approximations.

A polynomial  $f(x)$  of degree  $n$  is a function of the form

$$f(x) = a_0 + a_1x + a_2x^2 + \dots + a_nx^n \quad (1)$$

This form, called the **power form**, is the standard method of representing polynomial functions for mathematical purposes. However, if polynomials are to be evaluated in numerical computations, using other representations of  $f(x)$ , one of which is discussed below, often leads to increased accuracy. The reason for this is that these representations are more **numerically stable** than the power form.

One such representation, often called Horner's rule,  $f(x)$  of (1) is re-expressed in the form

$$f(x) = a_0 + (a_1 + (a_2 \dots (a_{n-1} + a_nx)x \dots)x). \quad (2)$$

Consider the evaluation of

$$f(x) = 7 + 3x + 4x^2 - 9x^3 + 5x^4 + 2x^5.$$

An expression for evaluating  $f(x)$  using Horner's rule is

$$7.0 + (3.0 + (4.0 + (-9.0 + (5.0 + 2.0 * x) * x) * x) * x) * x$$

which requires only 5 multiplications, which is at least 10 multiplications less than evaluating  $f(x)$  using the expression

$$7.0 + 3.0 * x + 4.0 * x^2 - 9.0 * x^3 + \dots$$

where  $x^2$  is counted as one multiplication. This representation can be coded in the following form in R:

```
sum= a[n]
for (i in n-1:1){
sum=sum*x+a[i]
}
cat(" Series sum = '",sum,fill=T)
```

where  $\mathbf{a}$  is a vector containing the coefficients  $a_1, a_2, \dots, a_n$ .

A finite series approximation for  $\cos x$  is given by

$$\cos x = a_0 + a_2x^2 + a_4x^4 + a_6x^6 + a_8x^8 + a_{10}x^{10} + \epsilon(x)$$

where  $0 \leq x \leq \pi/2$  and  $|\epsilon(x)| \leq 2 \times 10^{-9}$ , and the constants are

$$\begin{aligned} a_2 &= -.4999999963, \\ a_4 &= .0416666418, \\ a_6 &= -.0013888397, \\ a_8 &= .0000247609, \\ a_{10} &= -.0000002605. \end{aligned}$$

The following script in R uses this approximation to compute  $\cos(1.234)$

```
x=1.234
a=c(1,-.4999999963,.0416666418,-.0013888397,.0000247609,-.0000002605)
sum=a[6]
for (i in 5:1){
sum=sum*x^2+a[i]
}
cat(" x=", x, " approx cosine = ", round(sum,8), " true value = ", round(cos(x),8)
,fill=T)
```

Executing this code gave the result

```
x= 1.234  approx cosine =  0.3304651  true value =  0.3304651
```