values of $J_0(x)$ for range $x = 0.3, 0.9, 1.1, 1.5$ and $2.0$. Look up the tabulated values and compare. [Computer]

16. Modify `interp` so that it can handle any number of data points by using higher-order polynomials. After testing your program, give it the following values of the Bessel function: $J_0(0) = 1.0$; $J_0(0.2) = 0.9900$; $J_0(0.4) = 0.9604$; $J_0(0.6) = 0.9120$; $J_0(0.8) = 0.8463$; $J_0(1.0) = 0.7652$, and repeat the previous exercise. Do your estimates improve? [Computer]

17. Write a function that is similar to `intrpf` but that returns the estimated derivative at the interpolation point. The function will accept three $(x, y)$ pairs, fit a quadratic to the data, then return the value of the derivative of the quadratic at the desired point. [Computer]

18. A Bézier cubic curve is defined by the parametric equations

$$\begin{aligned} x(t) &= a_x t^3 + b_x t^2 + c_x t + x_1 \\ y(t) &= a_y t^3 + b_y t^2 + c_y t + y_1 \end{aligned}$$

where $0 \leq t \leq 1$. The Bézier control points are given by the relations

$$\begin{array}{ll} x_2 = x_1 + c_x/3 & y_2 = y_1 + c_y/3 \\ x_3 = x_2 + (c_x + b_x)/3 & y_3 = y_2 + (c_y + b_y)/3 \\ x_4 = x_1 + c_x + b_x + a_x & y_4 = y_1 + c_y + b_y + a_y \end{array}$$

The curve goes from $(x(0), y(0)) = (x_1, y_1)$ to $(x(1), y(1)) = (x_4, y_4)$ and is tangent to the lines $(x_1, y_1)$–$(x_2, y_2)$ and $(x_3, y_3)$–$(x_4, y_4)$. Write a program to draw a Bézier curve, given the control points $(x_1, y_1)$, ..., $(x_4, y_4)$. Draw the curve with control points (0,0), (2,1), (-1,1), and (1,0). [Computer]

## 1.5 NUMERICAL ERRORS

### Range Error

A computer stores individual floating-point numbers using only a small amount of memory. Typically, single precision (`float` in C++) allocates 4 bytes (32 bits) for the representation of a number, while double precision (`double` in C++; MATLAB's default precision) uses 8 bytes. A floating-point number is represented by its mantissa and exponent (for $1.60 \times 10^{-19}$, the decimal mantissa is 1.60 and the exponent is $-19$). The IEEE format for double precision uses 53 bits to store the mantissa (including one bit for the sign) and the remaining 11 bits for the exponent. Exactly how a computer handles the representation is not as important as knowing the maximum range and the number of significant digits.

The maximum range is the limit on the magnitude of floating-point numbers given the fixed number of bits used for the exponent. For single precision, a typical value is $2^{\pm 127} \approx 10^{\pm 38}$; for double precision it is typically $2^{\pm 1023} \approx 10^{\pm 308}$. Exceeding the single precision range is not difficult. Consider, for example, the evaluation of the Bohr radius in SI units,

$$a_0 = \frac{4\pi\epsilon_0 \hbar^2}{m_e e^2} \approx 5.3 \times 10^{-11} \text{ m} \tag{1.6}$$