Fitting polynomials to:

Discrete data

(either computed or empirical, and collected in a $table$ of x and y values).

This may be done for several different reasons. We may want to

1. accurately $interpolate$ (compute y using a value of x not found in the table itself).
2. draw a smooth picture connecting all data points,
3. fit a simple curve (linear, quadratic) to empirical (not so accurate) data. The curve be 'as close as possible' to the individual data points - we will have to agree on some overall criterion.

More complicated mathematical functions

over a specific range of x values. Similarly to the previous case. we cannot do this exactly, but have to minimize (in some well defined sense) the error of the fit.

There are several reasons for doing this:

1. Polynomials are easy to evaluate (we just add/subtract and multiply - and ultimately, all numerical computation has to be reduced to these)
2. they are also easy to integrate and differentiate - we may thus substitute our fitted polynomial for the actual function (which may be very hard or even impossible to integrate).

To facilitate the procedure (of fitting polynomials to a function), several sets of orthogonal polynomials are introduced (e.g. Legendre, Chebyshev, Hermite, etc.).

Trigonometric polynomials

are used for a specific type of data - fast Fourier transform.

Integration

The way how we can numerically evaluate $\int_A^B y(x)\,dx$ is to choose a set of x values (so called NODES) in the $[A,\ B]$ interval, for each (say $x_i,\ i = 0,\ 1,\ 2,\ ...,\ n$) of these compute the corresponding $y_i \equiv y(x_i)$. We then have to develop a formula for combining these y_i values to accurately estimate the integral (the area between the $y(x)$ function and the x axis).

A part of the problem is clearly the choice of the nodes. There are two distinct ways of approaching it:

1. A sensible (but in a sense arbitrary) choice of n equidistant values (effectively subdividing $[A,\ B]$ into n equal-length subintervals, leading to two basic 'rules' of integration, trapezoidal and Simpson, to be studied in detail.
2. A choice of n points which is, in a certain sense, $optimal$ (we can define 'optimal' only when we have a better understanding of the issues).

Differentiation

similarly involves estimating the value of $y'(x)$, $y''(x)$, etc. at $x = x_0$. This can be done by computing $y(x)$ at x_0 and a few extra values of x in the neighborhood of x_0 (this time, we will almost always choose them equidistant), and plugging them into the corresponding formula (which, of course, will be our task to *develop*). The major problem facing us here will the round-off error.

Ordinary differential equations

The formulas for numerical differentiation can also be used (this is in fact their major application) to solve, numerically, various types of ordinary and partial differential equations. We will deal with some examples of the ODE variety only, of two basic types

1. Boundary-value problem. In this context, we will also have to learn solving nonlinear (regular) equations.Solving ordinary differential equations
2. Initial-value problem - Runge-Kutta methods.

Matrix Algebra

The basic problem is to solve n linear equations for n unknowns, i.e. $\mathbb{A}\mathbf{x} = \mathbf{r}$, where $\mathbb{A}$ is an n by n (square) matrix, $\mathbf{x}$ is the (column) vector of the n unknowns, and $\mathbf{r}$ is similarly a vector of the right hand side values. The simplest technique uses the so called Gaussian elimination and backward substitution. One can reduce the round-off error by adding an extra step (row interchange) called pivoting.

We will then learn how to solve, iteratively, n *non*-linear equations for n unknowns, by Newton's method - we will still need matrices!

Eigenvalues and eigenvectors

of square matrices are defined by

$$\mathbb{A}\mathbf{x} = \lambda\mathbf{x}$$

where $\mathbf{x}$ (non-zero) is an eigenvector and λ an eigenvalue.

To simplify the issue, we will assume that $\mathbb{A}$ is symmetric (a fairly important class of matrices), which implies that both eigenvalues and eigenvectors must be *real* (they could be *complex* in general). We will then learn how to find them, one by one (there is n of them in general), by first utilizing Housholder's method to reduce $\mathbb{A}$ to a tridiagonal matrix, and then the applying, repeatedly, the so called $\mathbb{QL}$ algorithm to extract the smallest eigenvalue. The resulting matrix is then deflated and the process repeated till all eigenvalues are found.

Remaining Topics

There is a number of important topics which we will not have time to discuss in this brief course, namely:

1. Solving partial differential equations.
2. Optimizing a function of several variables (finding its largest or smallest value).