1. **Polynomial interpolation.** For any given four points there exists a unique polynomial that passes through each of these points. Evaluate this polynomial for points (1,1), (2,3), (3,6), (4,8) at $x = 2.5$. Do this by implenting the Neville's algorithm by hand and then by using the Numerical Recipes library function `polint`.

2. **Chebyshev polynomial expansion.** A function can be approximated by the Chebyshev polynomial expansion formula

$$f(x) \approx -\frac{1}{2}c_0 + \sum_{k=0}^{N-1} c_k T_k(x), \tag{1}$$

where coefficients c_j are

$$c_j = \frac{2}{N}\sum_{k=1}^{N} f\left(\cos\frac{\pi(k-1/2)}{N}\right)\cos\frac{\pi j(k-1/2)}{N}. \tag{2}$$

Calculate the approximation for function $\tanh(x)$, where $-2 \leq x \leq 2$ for $N = 4, 6, 10$. Plot the approximations and compare them to the original function. Use Numerical Recipes routines `chebft` and `chebev`.

3. **Padé approximant.** Padé approximants are a widely used alternative to Taylor and Chebyshev series. Their convergence is not restricted to the nearest pole, and they can even reflect the analytic properties of the underlying function. Find the Padé approximant for an odd function whose power series is found experimentally to be

$$g(x) \approx 1.000x + 0.333x^3 + 0.133x^5 + \cdots \tag{3}$$

Do this first by hand and then by using Numerical Recipes routines.

4. **Rational and polynomial interpolation.** Approximate a function $f(x)$ in given intervals using polynomial and rational function interpolation.

   (a) $f(x) = \cos(x)$, $x \in [0, 2\pi]$.

   (b) Function with a pole in the interpolation interval, $f(x) = \tan(x)$, $x \in [0, 2]$.

   (c) Runge's function, $f(x) = 1/(1 + 25x^2)$, $x \in [-1, 1]$.

   (d) Function with an essential singularity, $f(x) = \ln(x^2/2)$, $x \in [\epsilon, 1/10]$, where $\epsilon > 0$ is a small number that is representable with single precision variable.

   Tabulate the functions at a few, N, points and use `ratint` and `polint` to do the interpolation. Experiment with different values of N.

`void pade(double coef[], int n, float *resid)`

Generates coefficients for a Padé approximant

Arguments

`double coef[]`	*input*	Array of coefficients of the power series of the fitted function
`int n`	*input*	Number of coef's in array `coef`.
`float *resid`	*output*	Residual, indicates the quality of the fit.

`double ratval(double x, double coef[], int n, int m)`

Evaluates a rational function

Arguments

`double x`	*input*	Point where function is evaluated
`double coef[n+m+1]`	*input*	Coefficients of the rational function, `coef[0:n]` for numerator and `coef(n+1:m+n)` for denominator. Assuming zeroth term in denominator is 1.
`int n`	*input*	Polynomial order of the numerator.
`int m`	*input*	Polynomial order of the denominator.

`void polint(float xa[], float ya[], int n, float x, float *y, float *dy)`

Evaluates the polynomial interpolating given points at a point

Arguments

`float xa[]`	*input*	Array of x–coordinates
`float ya[]`	*input*	Array of y–coordinates
`int n`	*input*	Number of points
`float x`	*input*	Point where to evaluate the polynomial
`float *y`	*output*	Pointer to the value of the interpolating function
`float *dy`	*output*	Pointer to an error estimate

`void ratint(float xa[], float ya[], int n, float x, float *y, float *dy)`

Evaluates the rational function interpolating given points at a point

Arguments

`float xa[]`	*input*	Array of x–coordinates
`float ya[]`	*input*	Array of y–coordinates
`int n`	*input*	Number of points
`float x`	*input*	Point where to evaluate the rational function
`float *y`	*output*	Pointer to the value of the interpolating function
`float *dy`	*output*	Pinter to an error estimate