A Runge-Kutta-Fehlberg solver using traits and `Eigen`

Paolo Joseph Baioni

April 11, 2025

Finite differences

Implement a `C++` template class to compute derivatives of any order of a given function (callable object) using recursive backward/forward first-order finite difference formulas.

RKF solver

This example (a modified version of `Examples/src/RKFSolver`) is about a set of tools that implement embedded Runge-Kutta-Fehlberg methods to solve non-linear scalar, vector and matrix Ordinary Differential Equations, based on the `Eigen` library.
Consider an ODE of the form

$$\frac{\mathrm{d}y}{\mathrm{d}y} = f(t, y)$$

and consider the coefficients b_i, b_i^*, a_{ij}, c_i to be given in the form of a Butcher tableau. We recall that, at each time step t_n, an embedded RKF method of order p consists of the following steps.

1. Compute the high-order solution $y_{n+1} = y_n + h\sum_{i=1}^{p} b_i k_i$, where $k_i = f\left(t_n + c_i h_n, y_n + \sum_{j=1}^{i-1} a_{ij} k_j\right)$, and $h_n = t_{n+1} - t_n$.
2. Compute the low-order solution $y_{n+1}^* = y_n + h\sum_{i=1}^{p} b_i^* k_i$.
3. Compute the error $\varepsilon_{n+1} = y_{n+1} - y_{n+1}^* = h\sum_{i=1}^{p}(b_i - b_i^*)k_i$.
4. Adapt the step size $h_{n+1} = \tau_n h_n$, where τ_n is a reduction (< 1)/expansion (> 1) factor depending on whether ε_{n+1} is larger or smaller than a prescribed tolerance.

RKF solver

The code structure is the following:

- `ButcherRKF` contains the definition of the Butcher tableaux for the most common RKF methods.
- `RKFTraits` defines the basic structure that enable to statically select the type of the equation(s) to be solved (*i.e.* scalar, vector, matrix).
- `RKFResult` is a data structure containing the output of the RKF solver.
- `RKF` implements a generic RKF solver interface, filling a proper `RKFResult` object.