MA2501
Deadline April 12, 2019

You can work in groups of maximum 3 persons.

Adaptive quadrature and Romberg quadrature

$\boxed{1}$ In this exercise you will have to implement Romberg integration and test your code computing the integral of different functions.

We denote the exact integral with $\mathcal{I}_{[a,b]}(f)$,

$$\mathcal{I}_{[a,b]}(f) := \int_a^b f(x)\,dx.$$

a) Implement the algorithm for Adaptive Simpson quadrature to compute the integral $\mathcal{I}_{[a,b]}(f)$ at a desired accuracy. Assume f is four times differentiable. Denote with $S(a,b)$ the Simpson quadrature on the interval $[a,b]$:

$$S(a,b) = \frac{b-a}{6}\left[f(a) + 4f\left(\frac{a+b}{2}\right) + f(b)\right] \approx \mathcal{I}_{[a,b]}(f).$$

Input

- The function that we want to integrate.
- The interval of integration, $[a,b]$.
- A tolerance (TOL) to be used in a stopping criterion when the error is sufficiently small.

Core of the recursive algorithm

```
AdQuad($f$,$[a,b]$,$TOL$)

• $I_0 = S(a,b)$
• $c := \frac{a+b}{2}$
• $\tilde{I} := S(a,c) + S(c,b)$.
• Error estimate $\tilde{e} := \frac{1}{15}|\tilde{I} - I_0|$.
• If $\tilde{e} \leq TOL$ then
      $\tilde{I} := \tilde{I} + \frac{1}{15}(\tilde{I} - I_0)$,
  else
      $\tilde{I} := \tilde{I}_{[a,c]} + \tilde{I}_{[c,b]}$,
  where $\tilde{I}_{[a,c]} =$ AdQuad($f$,$[a,c]$,$\frac{TOL}{2}$) and $\tilde{I}_{[c,b]} =$ AdQuad($f$,$[c,b]$,$\frac{TOL}{2}$).
```

- Return $\tilde{I}$.

b) Test your code to make sure that it computes the correct values of the integrals and to the correct accuracy, i.e. with an error below the tolerance. You should test your program on the following test problems:

- $f(x) = \cos(2\pi x)$, $x \in [0, 1]$;
- $f(x) = e^{3x} \sin(2x)$, $x \in [0, \pi/4]$.

Compare the results you obtain against the exact value of the integrals, $\mathcal{I}_{[a,b]}(f)$. Make a plot of the error $|\tilde{I} - \mathcal{I}_{[a,b]}(f)|$ for decreasing values of the tolerance, verify that the error is bounded by the tolerance.

c) Implement a function in Python performing Romberg integration. Let $h_0 = b - a$ and $h_n = \frac{h_0}{2^n}$, the elements of the first column of the Romberg matrix are:

$$R(n, 0) = \frac{1}{2}R(n-1, 0) + h_n \sum_{i=1}^{2^{(n-1)}} f(a+(2i-1)h_n), \quad n = 1, \ldots, m-1, \tag{1}$$

with

$$R(0, 0) = \frac{1}{2}h_0(f(a) + f(b)) \tag{2}$$

Use the formula for computing the columns of the Romberg matrix:

$$R(n, k) = R(n, k-1) + E(n, k), \quad k = 1, \ldots, n. \tag{3}$$

where

$$E(n, k) := \frac{1}{4^k - 1}[R(n, k-1) - R(n-1, k-1)]. \tag{4}$$

Input

- The function that we want to integrate.
- The interval of integration, $[a, b]$.
- Maximum allowed dimension (number of rows and columns) for the Romberg matrix: m.
- A tolerance (TOL) to be used in a stopping criterion when the error is sufficiently small.

Core of the algorithm

Start computing $R(0, 0)$ by the trapezoidal rule on $[a, b]$.
For $n = 1, \ldots, m - 1$

- Compute $R(n, 0)$ using (1).
- Compute the correction in formula (4). This is also an estimate of the error.
- Update the columns of the n-th row of the Romberg matrix using (3) for $k = 1, \ldots, n$.

- Check convergence. Use (4) as an estimate for the error. If $|E(n,n)| < TOL$ give $R(n,n)$ as output approximation to the integral, else continue.

d) You should test your program on the following test problems:

- $f(x) = \cos(2\pi x)$, $x \in [0,1]$, with exact integral $\mathcal{I}_{[0,1]}(f) = 0$;
- $f(x) = x^{\frac{1}{3}}$, $x \in [0,1]$, with exact integral $\mathcal{I}_{[0,1]}(f) = \frac{3}{4}$.

Compare the results you obtain against the exact value of the integrals, $\mathcal{I}_{[a,b]}(f)$ and against the values computed by Adaptive Simpson's quadrature.

You should provide numerical evidence of the convergence of Romberg algorithm as $n \to \infty$. Consider $\mathcal{E}(n,0) := |\mathcal{I}_{[a,b]}(f) - R(n,0)|$, $\mathcal{E}(n,n) := |\mathcal{I}_{[a,b]}(f) - R(n,n)|$, where $R(n,k)$ for $n = 0,\ldots,$ $k = 0,\ldots,n$ are the entries of the Romberg matrix.

Explain the behaviour using the Euler-Mclaurin formula.

Rigid body simulation

$\boxed{2}$ Consider the free rigid body Euler equations

$$\dot{\mathbf{m}} = \mathbf{m} \times (T^{-1}\mathbf{m}), \quad \mathbf{m}(0) = \mathbf{m}_0,$$

where T is the diagonal inertia tensor, $T = \mathrm{diag}(I_1, I_2, I_3)$. We assume that the principal moments of inertia (the diagonal entries of the inertia tensor) are distinct and in increasing order, i.e. $I_1 < I_2 < I_3$.

The 2-norm of the angular momentum γ and the energy function E

$$\gamma = \mathbf{m}(t)^T\mathbf{m}(t), \quad E = \frac{1}{2}\mathbf{m}(t)^T\,(T^{-1}\mathbf{m}(t)),$$

are constant along the solution $\mathbf{m}(t)$. This means that the solutions $\mathbf{m}(t)$ are the curves obtained by the intersection of the sphere $\gamma = \mathbf{m}(t)^T\mathbf{m}(t)$ with the ellipsoid $E = \frac{1}{2}\mathbf{m}(t)^T\,(T^{-1}\mathbf{m}(t))$, see [1, p. 100] for an illustration.

a) Implement the mid-point Runge-Kutta implicit integration method to solve numerically the free rigid body Euler equations.

You are free to choose your favourite approach to solve the nonlinear system of equations to be solved at each time-step. You could for example find an explicit expression for the Jacobian of the nonlinear system and implement a Newton method, or you could use fixed-point iteration.

b) Implement the improved Euler method:

$$\begin{aligned}\mathbf{m}_{n+\frac{1}{2}} &= \mathbf{m}_n + \frac{h}{2}(\mathbf{m}_n \times (T^{-1}\mathbf{m}_n)),\\ \mathbf{m}_{n+1} &= \mathbf{m}_n + h(\mathbf{m}_{n+\frac{1}{2}} \times (T^{-1}\mathbf{m}_{n+\frac{1}{2}})).\end{aligned}$$

This is an explicit method and you do not have to solve any equations.

c) Use now one of the NumPy routines for the numerical solution of ordinary differential equations to solve the free rigid body Euler equations. The goal is to obtain a very accurate reference solution that can be used to test that your implementation of the midpoint and of the improved Euler method is correct.

Provide numerical evidence that the midpoint method and the improved Euler method have order 2: compare the solution given by these numerical methods for different values of h and the reference solution obtained using the NumPy ODE routine. Choose $[0, 1]$ as time interval. Provide a `loglog` plot of the error versus the step-size h showing that you get a line of slope 2.

d) Integrate on relatively large time intervals. Compute and plot the error in γ and E as a function of time for the midpoint method, the improved Euler method, and for the NumPy ODE routine. What do you observe? How is this error affected by the tolerances used in the NumPy method and in the solution of the nonlinear equations for the midpoint?

e) Plot the solutions obtained by the various rigid body integrators as curves on the sphere $\gamma = \mathbf{m}(t)^T\mathbf{m}(t)$. Make one or several plots similar to the ones you can find in [1, p. 100, 127,164,169].

References

[1] E. Hairer, C. Lubich and G. Wanner, *Geometric Numerical Integration*, Springer.