Collocation integrator based on Legendre polynomials

V.Sh. Shaidulin

Abstract. This work presents an algorithmic implementation of a collocation integrator based on Legendre polynomials. Issues about effective implementation, application conditions, and numerical stability are considered. The presented integrator already has a software implementation.

Introduction

Collocation integration methods are a different interpretation of the class of Runge–Kutta methods first noted by Hammer and Hollingsworth [1, 2]. The essence of this interpretation is to present the solution in a polynomial form. It is interesting to note that, unlike other integration methods, we obtain a continuous solution at each step. This work presents an algorithmic implementation of a collocation integrator based on Legendre polynomials.

1. Collocation polynomial

Suppose we have a system of ordinary differential equations, represented in the form

$$\dot{\mathbf{y}} = \mathbf{f}(t, \mathbf{y}), \qquad \mathbf{y} \in \mathbb{O} \subseteq \mathbb{R}^n, \quad \mathbf{f} : \mathbb{R} \times \mathbb{O} \to \mathbb{R}^n. \tag{1}$$

Collocation integration methods propose to approximately represent the solution of a system at a step of size h with a beginning at t_0 in the form of a collocation polynomial $\mathbf{u}$ of a given degree s:

$$\mathbf{y}(t_0 + h\tau) \approx \mathbf{u}(\tau),$$

where τ is dimensionless time varying on the interval $[0, 1]$. We can define the polynomial $\mathbf{u}(\tau)$ as a linear combination over some basis $P_k(\tau)$:

$$\mathbf{u}(\tau) = \sum_{k=0}^{s} \boldsymbol{\alpha}_k P_k(\tau), \qquad \boldsymbol{\alpha}_k \in \mathbb{R}^n.$$

The coefficients $\boldsymbol{\alpha}_k$ are implicitly determined by a system of nonlinear equations obtained from (1) for some set of nodes c_j of size s:

$$\sum_{k=1}^{s} \boldsymbol{\alpha}_k P'_k(c_j) = \mathbf{f}(t_0 + hc_j, \mathbf{y}_j)h, \qquad \mathbf{y}_j = \sum_{k=0}^{s} \boldsymbol{\alpha}_k P_k(c_j), \tag{2}$$
$$\boldsymbol{\alpha}_0 + \sum_{k=1}^{s} \boldsymbol{\alpha}_k P_k(0) = \mathbf{y}(t_0).$$

2. Calculation of coefficients $\boldsymbol{\alpha}_k$

For efficient calculations, we rewrite system (2) in the form:

$$\sum_{k=1}^{s} \boldsymbol{\alpha}_k P'_k(c_j) = \mathbf{f}(t_0 + hc_j, \mathbf{y}_j)h, \qquad \mathbf{y}_j = \mathbf{y}(t_0) + \sum_{k=1}^{s} \boldsymbol{\alpha}_k \Big(P_k(c_j) - P_k(0) \Big).$$

This allows us to move on to the matrix notation of this system of equations:

$$\mathcal{A}\mathcal{C}^T = \mathcal{F}.$$

Here $\boldsymbol{\alpha}_k$ for $k = 1, \ldots, s$ are collected into a matrix $\mathcal{A}$ of size $n \times s$, $\mathbf{f}(t_0 + hc_j, \mathbf{y}_j)h$ into a matrix $\mathcal{F}$ of size $n \times s$ and $P'_k(c_j)$ into a matrix $\mathcal{C}$ of size $s \times s$. In the matrices $\mathcal{A}$ and $\mathcal{F}$, the columns are the corresponding vectors in ascending order of the indices k and j. In the matrix $\mathcal{C}$, index k lists the columns and j lists the rows.

The calculation of matrix A is done iteratively and is efficient when using modern linear algebra libraries.

3. Application conditions

Let's write it like this:

$$\mathcal{A} = \mathcal{F}\mathcal{C}^{-T}, \tag{3}$$

The iterative process of calculating the matrix $\mathcal{A}$, given by the equation (3), will converge if the norm of the Jacobian matrix of the right side of the equation (3) is strictly less than one:

$$\left\| \frac{\partial \left(\mathcal{F}\mathcal{C}^{-T} \right)}{\partial \mathcal{A}} \right\| < 1.$$

We use the definitions given earlier and obtain a restriction for the right-hand side function $\mathbf{f}$:

$$\max_p \sum_{q=1}^{n} \sum_{i=1}^{s} \left| \left(\frac{\partial f_p}{\partial y^{(q)}} \right)_{\mathbf{y}=\mathbf{y}_i} \right| < \frac{1}{hsbz}\,. \tag{4}$$

Here:

$$b = \max_{j,k} \Big| P_k(c_j) - P_k(0) \Big|, \qquad z = \max_{j,k} \Big| \left(\mathcal{C}^{-T} \right)_{j,k} \Big|.$$

4. Numerical stability

Along with the convergence of the iterative process of calculating the matrix $\mathcal{A}$, it is important to ensure that there is numerical stability so that rounding errors don't significantly affect the result. As can be seen from equation (3), the numerical stability will be determined by the value of z introduced earlier. The smaller it is, the better.

Conclusion

The presented algorithm for the collocation integrator already has a software implementation (`https://github.com/shvak/collo`). It proved the efficiency of calculating matrix $\mathcal{A}$ using equation (3). This work tries to determine how successful the choice of Legendre polynomials is as a basis for different quadrature grids.

References

[1] Hammer P. C., Hollingsworth J. W., *Trapezoidal methods of approximating solutions of differential equations* // Math. Tables Aids Comput. — 1955. — Vol. 9. — P. 92–96.

[2] Hairer E., Wanner G., Lubich C., *Geometric Numerical Integration: Structure-Preserving Algorithms for Ordinary Differential Equations. Second Edition.* Springer, 2006. — P. 644.

V.Sh. Shaidulin
The Department of Celestial Mechanics
Saint Petersburg State University
Saint Petersburg, Russia
e-mail: `v.shaidulin@spbu.ru`