H – The Radau IIA method for ODE integration

The method of lines translates an evolutionary PDE into a large system of ODEs. As we mentioned before this system exhibits stiffness which in consequence requires specialized numerical algorithms. Our simulations proved that the family of the Radau methods is a good choice.

The name “Radau methods” is usually applied to the family of fully implicit Runge-Kutta methods which possess good stability properties. Generally, an RK method has the form

$$u_{n+1} = u_n + \Delta t \sum_{i=1}^{s} b_i K_i, \quad n = 0,1,2,\ldots \tag{1}$$

where

$$K_i = f(t_n + c_i \Delta t, u_n + \Delta t \sum_{j=1}^{s} a_{ij} K_j), \quad i = 1,\ldots,s \tag{2}$$

The coefficients $\{a_{ij}\},\{c_i\}$ and $\{b_i\}$ fully characterize the method. If the coefficient $a_{ss} \neq 0$, the method is implicit (so we must solve in general nonlinear system of algebraic equations in order to accomplish the next time step). There are some additional relations between coefficients which must hold to ensure the theoretical convergence of such method

$$\sum_{j=1}^{s} b_j = 1, \ \sum_{j=1}^{s} a_{ij} = c_i \quad i = 1,\ldots,s \tag{3}$$

For $s \geq 2$ these relations do not determine the coefficients, so we have really a big family of Runge-Kutta methods. The Radau methods is just a subfamily of implicit Runge-Kutta methods. The method we use, implemented by Hairer-Wanner in a FORTRAN subroutine called RADAU5, is defined by

$$\begin{array}{cccc|cccc} c_1 & a_{1,1} & a_{1,2} & a_{1,3} & \frac{4-\sqrt{6}}{10} & \frac{88-7\sqrt{6}}{360} & \frac{296-169\sqrt{6}}{1800} & \frac{-2+3\sqrt{6}}{225} \\ c_2 & a_{2,1} & a_{2,2} & a_{2,3} & \frac{4+\sqrt{6}}{10} & \frac{296+169\sqrt{6}}{1800} & \frac{88+7\sqrt{6}}{360} & \frac{-2-3\sqrt{6}}{225} \\ c_3 & a_{3,1} & a_{3,2} & a_{3,3} & 1 & \frac{16-\sqrt{6}}{36} & \frac{16+\sqrt{6}}{36} & \frac{1}{9} \\ \hline & b_1 & b_2 & b_3 & & \frac{16-\sqrt{6}}{36} & \frac{16+\sqrt{6}}{36} & \frac{1}{9} \end{array} \tag{4}$$

Remarks on the implementation

The main task is to solve the system of algebraic equations

$$g_i = u_0 + \Delta t \sum_{j=1}^{s} a_{ij} f(t_0 + c_j h, g_j) \quad for\ i = 1, \ldots, s. \tag{5}$$

Upon introducing $z_i = g_i - u_0$ we have

$$z_i = \Delta t \sum_{j=1}^{s} a_{ij} f(t_0 + c_j h, u_0 + z_j) \quad for\ i = 1, \ldots, s. \tag{6}$$

To advance to the next time step $t_0 + \Delta t$ we use

$$u_1 = u_0 + \Delta t \sum_{j=1}^{s} a_{ij} f(t_0 + c_j h, u_0 + z_j), \tag{7}$$

and in a such form it would require s additional function evaluations. This can be avoided, if the matrix $A = [a_{ij}]$ is nonsingular. In matrix notation the eq. (6) can be written as $z = \Delta t A F(z)$, where $F(z) = (f(t_0 + c_j \Delta t, u_0 + z_j))_{j=1,\ldots,s}$. Now the formula (7) can be expressed as

$$u_1 = u_0 + \sum_{j=1}^{s} d_j z_j \tag{8}$$

where $d = (d_1, \ldots, d_s)^T = (A^T)^{-1} b.$ In addition there are two more advantages of such reformulation (see [1], p. 119).

Theoretical analysis (e.g. the paper of Linger & Willoughby, 1970) and numerical tests strongly suggest that the Newtown's method, instead of the simple fixed-point iteration, should be applied to solve the system (6).

---

[1]Ernst Hairer, Gerhard Wanner, Solving Ordinary Differential Equations II. Stiff and Differential-Algebraic Problems. Springer Series in Comput. Mathematics, Vol. 14, Springer-Verlag 1991, Second revised edition 1996.

The standard Newton's method needs for each iteration the solution of a linear system with matrix

$$[\delta_{ij} - \Delta t a_{ij} D_y f(t_0 + c_j \Delta t, u_0 + z_j)]_{ij=1,\ldots,s} \tag{9}$$

In order to simplify this, we can use the Jacobian $J = D_y f(t_0, u_0)$ for all Newton's iterations. Thus the simplified Newton iterations become

$$\begin{aligned} (I - \Delta t A \otimes J)\Delta Z^{[k]} &= -Z^{[k]} + \Delta t A \otimes IF(Z^{[k]}) \\ Z^{[k+1]} &= Z^{[k]} + \Delta Z^{[k]} \end{aligned} \tag{10}$$

where $Z^{[k]} = (z_1^{[k]}, \ldots, z_s^{[k]})$, $F(Z^{[k]}) = (f(t_0 + c_1 \Delta t, u_0 + z_1^{[k]}), \ldots, f(t_0 + c_s \Delta t, u_0 + z_s^{[k]}))$.

Remarks

- Each iteration requires s evaluations of f and the solution of a *ns*-dimensional linear system.
- The matrix is the same for all iteration in one step.
- The LU decomposition while solving the linear system (10) is most frequently used.

Stopping Criteria for Iterations

If convergence is linear with the constant rate $\vartheta \in]0,1[$ we have

$$\| \Delta Z^{[k+1]} \| \leq \vartheta \| \Delta Z^{[k]} \|. \tag{11}$$

From this we derive

$$\| Z^{[k+1]} - Z^* \| \leq \frac{\vartheta}{1-\vartheta} \| \Delta Z^{[k]} \| \tag{12}$$

where Z^* is the exact solution.

The convergence rate ϑ can be estimated by the quantities

$$\vartheta_k = \| \Delta Z^{[k]} \| / \| \Delta Z^{[k-1]} \| \quad k \geq 1. \tag{13}$$

Thus because $\vartheta_k \simeq \vartheta$, the error can be estimated as follows

$$\| Z^{[k+1]} - Z^* \| \leq \frac{\vartheta_k}{1-\vartheta_k} \| \Delta Z^{[k]} \| \tag{14}$$

yielding the strategy:

$$\eta_k \| \Delta Z^{[k]} \| \leq \kappa \cdot Tol \qquad (\text{with } \eta_k = \vartheta_k / 1-\vartheta_k). \tag{15}$$

where *Tol* is the local truncation error and κ is an additional parameter. The implementation uses κ around 10^{-1} or 10^{-2}.

The adaptive time step strategy is also applied in case we cannot reach a desired accuracy of (15) in a reasonably number of steps (in the implementation "reasonably" means no more than 10). In such case the substitution $\Delta t = \Delta t/2$ is used and computations are restarted with this smaller step size.

The Linear System

Step Size Selection

To control adaptively the time step size the implementation uses an *embedded* pair of methods. The auxiliary lower order method has the form

$$\hat{u}_1 = u_0 + \Delta t \gamma_0 f(t_0, u_0) + \Delta t \sum_{i=1}^{3} \hat{b}_i f(t_0 + c_i \Delta t, g_i) \tag{16}$$

where g_1, g_2, g_3 are the values obtained from the RADAU IIA method. Then the difference

$$\hat{u}_1 - u_1 = \gamma_0 \Delta t f(t_0, u_0) + \Delta t \sum_{i=1}^{3} (\hat{b}_i - b_i) f(t_0 + c_i \Delta t, g_i) \tag{17}$$

serves as the error estimation. This also can be written as follows (see eq. (8))

$$\hat{u}_1 - u_1 = \gamma_0 \Delta t f(t_0, u_0) + e_1 z_1 + e_2 z_2 + e_3 z_3. \tag{18}$$

But for stiff equations this formula is not sufficient so the expression proposed by Shampine is used

$$err = (I - \Delta t \gamma_0 J)^{-1} (\hat{u}_1 - u_1). \tag{19}$$

Further, to avoid the "hump" phenomenon (see [1], Sect. IV.7), in the first iteration step after every rejected time step for which $\| err \| > 1$ we use, instead of error estimate (19), the following expression for step prediction

$$\widetilde{err} = (I - \Delta t \gamma_0 J)^{-1} (\gamma_0 \Delta t f(t_0, u_0 + err) + e_1 z_1 + e_2 z_2 + e_3 z_3) \tag{20}$$