GOAL ORIENTED A POSTERIORI ERROR ESTIMATION FOR IMPLICIT-EXPLICIT RUNGE-KUTTA SCHEMES

JEHANZEB H. CHAUDRY, J.B. COLLINS

1. Introduction

We derive goal oriented *a posteriori* error estimates for implicit-explicit(IMEX) Runge-Kutta schemes for solving ordinary differential equations. Due to the variational nature of these error estimates, we must first restate these finite difference methods as finite element methods. This is done using a series of Taylor series projection operators. We then derive error estimation using standard methods, and break the error into various components.

IMEX schemes are used to solve equations that have both a stiff and non-stiff component. Here we solve the autonomous ODE

$$\dot{y} = f(y) + g(y), \tag{1}$$

where $g(y)$ is a stiff term, and $f(y)$ can be easily solved using explicit schemes.

2. IMEX Runge-Kutta schemes and equivalent finite element method

IMEX Runge-Kutta schemes are multi-stage finite difference methods that can be described by two Butcher tables, The two tables describe both the explicit and

$$\begin{array}{c|c} \mathbf{c} & \mathrm{A} \\ \hline & \mathbf{w} \end{array} \qquad \begin{array}{c|c} \mathbf{d} & \mathrm{B} \\ \hline & \mathbf{v} \end{array}.$$

implicit portions of the scheme. The matrices $A, B \in \mathbb{R}^{\nu \times \nu}$ describe the explicit and implicit schemes respectively. We assume that the elements of $\mathbf{d}$ are distinct. The IMEX Runge-Kutta scheme with the above Butcher tables is given as,

$$Y_i = Y_{n-1} + h\sum_{j=1}^{i-1} a_{ij} f(Y_j) + h\sum_{j=1}^{\nu} g(Y_j) \qquad i = 1, \ldots, \nu \tag{2}$$

$$Y_{n+1} = Y_n + h\sum i = 1^{\nu} w_i f(Y_i) + v_i g(Y_i). \tag{3}$$

Generic ν Stage Method

We look at a generic ν stage IMEX method for solving the autonomous problem

$$\dot{y} = f(y).$$

I am guessing at the form these Butcher tables should take, but from the examples I've seen, a general table should look like

$$\begin{array}{c|c} \mathbf{c} & \mathrm{A} \\ \hline & \mathbf{w} \end{array} \qquad \begin{array}{c|c} \mathbf{d} & \mathrm{B} \\ \hline & \mathbf{v} \end{array}$$

The matrix $A = (a_{ij})_{i,j=1}^{\nu}$ is strictly lower triangular, representing an explicit method. The matrix $B = (b_{ij})_{i,j=1}^{\nu}$ is lower triangular, representing a DIRK method. Here we assume the values d_i are all nonnegative distinct real numbers and are ordered in increasing order. If this is not the case, the classic example being RK4, then the work below must be modified, but I believe it is possible.

Notation. The following notation is used throughout this section.

- All calculation are done on the subinterval $I_n := [t_n, t_{n+1}]$.
- $h_n = t_{n+1} - t_n$ is the length of the subinterval under consideration
- $Y_n := Y(t_n)$.
- $t_{n+s} = t_n + h_n s$ for any $s \geq 0$.

Equivalency. The simplified form of the finite element will be:
Find $Y \in W_h$ such that

$$\langle \dot{Y}, v \rangle = \langle f(\mathcal{I}Y), v \rangle_{I_n, Q^f} + \langle g(\mathcal{I}Y), v \rangle_{I_n, Q^g} \quad \forall \ v \in V. \tag{4}$$

The $\mathcal{I}$ operator approximates Y with multiple Taylor approximations $P_i Y(t)$, described below. The operator is defined such that $\mathcal{I}Y(t_{n+d_i}) = P_i Y(t_{n+d_i})$. The simplest way I can think of doing this is using Lagrange interpolation, that is $\mathcal{I}Y(t) = \sum_{i=1}^{\nu} P_i Y(t) \prod_{j=1 j\neq i}^{\nu} \frac{(t - t_j)}{(t_i - t_j)}$. However, we could also define $\mathcal{I}Y(t)$ piecewise to be each $P_i Y(t)$ function on the correct subintervals.

We now define the quadratures used in equation (4).

$$\langle \varphi \rangle_{[a,b],Q^f} = (b-a) \sum_{i=1}^{\nu} w_i \varphi(a + d_i(b-a)) \tag{5}$$

$$\langle \varphi \rangle_{[a,b],Q^g} = (b-a) \sum_{i=1}^{\nu} v_i \varphi(a + d_i(b-a)) \tag{6}$$

Finally, we define the P_i operators. As mentioned, these replace Y with a Taylor approximation of Y. This is done by using the in constant term and the integral remainder form of the Taylor series:

$$P_i y(t) \approx y(t_n) + \int_{t_n}^{t} \dot{y}\, dt = \int_{t_n}^{t} f(y)\, dt + \int_{t_n}^{t} g(y)\, dt. \tag{7}$$

We then approximate the integrals in various ways.

Remark 1. In general these integral approximations, i.e. quadrature rules, are not very accurate. Most of them are only first order accurate. I am still trying to figure out how to identify the cancelation that must occur for the method as a whole to be accurate to higher order. It is possible there is another way of finding this equivalency that elucidates this cancelation better.

To obtain equivalency, we must define the P_i operators so that the following equations hold:

$$P_iY(t_{n+d_i}) = Y_n + h_n \sum_{j=1}^{i-1} a_{ij} f(P_jY(t_{n+d_j})) + h_n \sum_{j=1}^{i} b_{ij} g(P_jY(t_{n+d_j})) \tag{8}$$

Remark 2. Note that these are the equations for the stage variables in the IMEX method. Therefore, $P_iY(t_{n+d_i}) = Y_i$ in terms of the stage variables. Also, it is assumed here that we are considered DIRK implicit methods. If other methods wish to be considered, the upper limit of the second sum must be changed to 4.

To obtain an operator P_i we approximate the integrals in (7) with very particular quadratures,

$$P_iY(t) = Y_n + \langle f(\mathcal{I}Y)\rangle_{[t_n,t],Q_i^f} + \langle g(\mathcal{I}Y)\rangle_{[t_n,t],Q_i^g}. \tag{9}$$

The quadratures Q_i^f and Q_i^g are defined specifically so that when we evaluate $P_iY(t_{n+d_i})$ we satisfy (8).

$$\langle\varphi\rangle_{[a,b],Q_i^f} = (b-a)\sum_{j=1}^{i-1} \frac{a_{ij}}{d_i}\varphi\Big(a + \frac{d_j}{d_i}(b-a)\Big) \tag{10}$$

$$\langle\varphi\rangle_{[a,b],Q_i^g} = (b-a)\sum_{j=1}^{i} \frac{b_{ij}}{d_i}\varphi\Big(a + \frac{d_j}{d_i}(b-a)\Big) \tag{11}$$

With this definition, there should be an equivalence between $P_iY(t_{n+d_i})$ and the stage variables Y_i. That is, they should both satisfy the same equation and so be equal on each subinterval I_n.

Finally, if we evaluate (4) with $v = 1$, we should obtain the update equation for Y_{n+1} for the IMEX method. The only difference being that instead of Y_i we will see $P_iY(t_{n+d_i})$. If we are using a higher order cG method, the remaining nodal values can be obtained by using different function for v and the values of Y_n and Y_{n+1}

Summary. As an equivalency, this works. I encourage you to try it out, probably on a particular IMEX scheme and see that we get the same update equation. There are a few problems that I see immediately. First, this is really quite complicated. I'm not sure how much this matters, but I feel as though I had to contort myself backwards and sideways to get all the math to work out. For instance, the quadratures I derive give equivalency, but they are highly inaccurate. The $\mathcal{I}$ must also be defined carefully. The definition I gave above can turn into a high degree polynomial quickly, which could be a problem. However, defining it piecewise would make integration difficult. I'm wondering if splines might be the way to go. Still thinking about it.

A second problem is that this does not work if f depends on t. If we have a non-autonomous system, then this only works if $\mathbf{c} = \mathbf{d}$. Otherwise, we need a strange operator $\mathcal{I}$ that doesn't really make sense. If you're curious I can explain further. My suggestion is that we first get things working for the autonomous case, and fix the problem with the non-autonomous case later.

The final problem is that to do error estimation, we must evaluate $\mathcal{I}Y(t)$ many times, which means evaluating $P_iY(t)$ many times. This function is defined recursively and so requires a nonlinear solve for each evaluation. Now if g is linear, this is not a problem, otherwise, this could be prohibitively costly. Now $P_iY(t)$ must be evaluated a certain number of times to run the forward scheme, so it may be possible to use these evaluations in a quadrature for the error estimation. I suppose it would depend on the method being used.