Stability of the Lagrange-Galerkin method: the performance of quadrature in theory and practice

R.O. Jack

We investigate the stability properties of the Lagrange-Galerkin method using quadrature for both linear and quadratic elements. We find the regions of instability for various types of quadrature for linear advection in one dimension, and then perform numerical experiments on two test problems to assess the implications of these results for practical situations. The first problem is the familiar rotating cone problem and the second is advection in the flow field generated by the driven cavity problem.

The author is grateful to the Central Electricity Generating Board and the United Kingdom Atomic Energy Authority for their financial support.

The work reported here forms part of the research programme of the Oxford—Reading Institute for Computational Fluid Dynamics.

Oxford University Computing Laboratory
Numerical Analysis Group
8–11 Keble Road
Oxford, England OX1 3QD
Contents

1 Introduction 3

2 Stability of quadrature for the Lagrange–Galerkin method 3

2.1 Formulation of the direct Lagrange–Galerkin method for linear advection 4

2.2 Stability of the exactly integrated scheme 4

2.3 Stability of the scheme with quadrature 5

2.3.1 Piecewise linear elements 5

2.3.2 Stability with diffusion 8

2.3.3 Quadratic elements 9

2.4 Results 10

2.4.1 Linear elements 10

2.4.2 Quadratic elements 11

2.5 Fourier analysis for the Lagrange-Galerkin method with quadratic basis functions 12

3 Subdivision: a stable implementation of the method 16

3.1 Flow field definition 19

3.2 Advection problem 19

3.3 Results 20

3.4 Assessment of subdivision 22

4 Conclusions 24
1 Introduction

The Lagrange–Galerkin method has been used on quite an extensive range of problems [1,2,3]. Although the method is unconditionally stable in theory, in practice integrals are computed using quadrature, and then less is known about the stability properties of the method. It was only recently that the method was found to be only conditionally stable for linear advection with piecewise linear elements, when quadrature was used to evaluate inner products [4]. It was also realised that diffusion can make the method unconditionally stable providing that the diffusion coefficient is large enough, for advection–diffusion problems in one dimension with linear elements [5]. The analysis up until now was primarily concerned with demonstrating instability. No detailed analysis was performed for quadratic elements. In many ways it seems quite remarkable that so little should be known about this clearly critical topic for such a widely used method.

In this report we investigate the stability of the Lagrange-Galerkin method for linear advection–diffusion in one dimension with quadratic and linear elements. We also investigate stability questions for biquadratic elements for two two–dimensional test problems. In the first section we give our stability analysis. In the next section we present the results for the first test problem, the by now familiar rotating cone problem, with quadratic elements. The purpose of the experiment is to compare the stability properties of various quadratures in practice. In the last section we investigate another advection problem, in which a sinusoidal pulse is advected in the flow field generated by the double glazing problem. We compare the results obtained using quadrature with a generalised version of area–weighting which would be suitable for use on non–structured grids. We present our conclusions in the final section.

2 Stability of quadrature for the Lagrange–Galerkin method

We investigate the effects of quadrature on the stability of the Lagrange–Galerkin method. Without loss of generality we look at the direct formulation [4]. For completeness and motivation we first recall the proof of the stability of the exactly integrated scheme for the case of pure advection.
2.1 Formulation of the direct Lagrange–Galerkin method for linear advection.

Consider the Cauchy problem for the scalar, linear advection equation for $u(x, t)$:

$$ u_t + a \cdot \nabla u = 0, \quad x \in \mathbb{R}^d, \quad t > 0, $$

$$ u(x, t) = u_0, $$

where $u_0$ belongs to $L^2(\mathbb{R}^d)$ and the velocity field is incompressible, i.e.

$$ \nabla \cdot a = 0. $$

We can define characteristic paths or trajectories, $X(x, s; t)$, by

$$ X(x, s; t) = x, $$

$$ \frac{d}{dt} X(x, s; t) = a(X(x, s; t), t). $$

For $t^{n+1} = t^n + \Delta t$ we will denote $X(y, t^{n+1}; t^n)$ and $X(x, t^n; t^{n+1})$ by $x$ and $y$ respectively. The relation

$$ u(X(\cdot, t, t + \tau), t + \tau) = u(\cdot, t) $$

gives the solution to the original advection equation.

The so called direct Lagrange–Galerkin method uses this relation and an approximation at time $t^n$ given in terms of the finite element basis functions $\phi_j$,

$$ U^n = \sum U^n_j \phi_j, $$

to obtain $U^{n+1}$ satisfying

$$ (U^{n+1}, \phi_i) = \int U^n(x) \phi_i(y) dy $$

where $(\cdot, \cdot)$ denotes the $L^2$ inner product over $\mathbb{R}^d$.

2.2 Stability of the exactly integrated scheme

Suppose that we denote by $E_{\Delta t}(t)$ the solution operator $u(\cdot, t + \Delta t) = E_{\Delta t}(t)u(\cdot, t)$. Then we can write the direct formulation of the Lagrange–Galerkin method as follows

$$ (U^{n+1}, \phi^i) = (EU^n, \phi^i). $$
Multiplying both sides by $U^{n+1}_i$, summing over $i$ and using the Cauchy–Schwarz inequality together with the fact that $\|E\| = 1$ yields
\[(U^{n+1}, U^{n+1}) = (EU^n, U^{n+1}) \leq (U^n, U^n)^{1/2}(U^{n+1}, U^{n+1})^{1/2}.
\]
Hence
\[(U^{n+1}, U^{n+1}) \leq (U^n, U^n),
\]
demonstrating stability in the $L_2$ norm.

2.3 Stability of the scheme with quadrature

We denote the inner products evaluated by quadrature by $< \cdot, \cdot >$. If the weights of the quadrature rule are all positive then the inner product will give a norm. The quadrature based scheme is
\[< U^{n+1}, \phi^i > = < EU^n, \phi^i >.
\]
As for the exactly integrated scheme we multiply both sides with $U^{n+1}_i$ and sum over $i$ to give
\[< U^{n+1}, U^{n+1} > \leq < EU^n, EU^n >.
\]
For stability it suffices to demonstrate that
\[< EU^n, EU^n > \leq < U^n, U^n >.
\]
This amounts to requiring that the mass matrix evaluated with its basis functions evolved through one time step is 'no greater' than the mass matrix. Denoting the 'shifted' mass matrix by $M(\nu)$, $\nu$ being a CFL number and the 'unshifted' mass matrix by $M(0)$, we need to show that
\[M = M(0) - M(\nu) \geq 0,
\]
that is, $M$ is positive semi-definite.

2.3.1 Piecewise linear elements

For simplicity, we take a uniform mesh in one dimension of spacing $h$. In what follows we will denote the abscissae of the nodes of the quadrature rule in local coordinates (with reference to the interval $[-1,1]$) by $x'$ and the corresponding weights by $w'$. Using such a rule an integral
\[\int_0^h f(x)dx
\]
is approximated by the sum
\[ \frac{h}{2} \sum_{i=1}^{m+1} f(x^i)w^i. \]
We will frequently use the abbreviations \( \alpha = x + \nu \) and \( \alpha^i = x^i + \nu \), where \( \nu \) is the CFL number. In our notation a CFL number of 2 will correspond to a translation through one element. We will also encounter sums of the form \( \sum_{i=1}^{m+1} \) in what follows. Here \( q \) is defined to be such that \( \nu \in [1 - x_{q-1}, 1 - x_q] \).

For each element there are two basis functions,
\[ \phi_1(s) = \frac{1}{2}(1 - s), \]
\[ \phi_2(s) = \frac{1}{2}(1 + s). \]

The unshifted mass matrix is, ignoring boundary conditions,
\[ M(0) = \frac{h}{2}[a, b, a] \]
where
\[ a = \frac{1}{4} \sum_{i=0}^{m+1} (1 - x_i^3)w^i \]
and
\[ b = \frac{2}{4} \sum_{i=0}^{m+1} (1 + x_i^3)w^i. \]

Imposing periodic boundary conditions deletes the last row and column of the matrix and modifies the first row and column, to give entries \( M_{11} = a \), \( M_{1N-1} = b \) and \( M_{N-1} = b \).

The shifted mass matrix has a similar form:
\[ M(\nu) = \frac{h}{2}[a^*, b^*, a^*], \]
with the boundary conditions modifying this as for the standard mass matrix. Here
\[ a^* = \frac{1}{4} \sum_{i=0}^{m+1} (1 - \alpha_i^3)w^i - \sum_{i=q}^{m+1} (1 - \alpha_i)w_i \]
and
\[ b^* = \frac{1}{2} \sum_{i=0}^{m+1} (1 + \alpha_i^3)w^i - 2 \sum_{i=0}^{m+1} (1 - \alpha_i)w^i. \]

We find that the matrix
\[ M = M(0) - M(\nu) \]
has the form

\[ \frac{h}{2} [\epsilon_1, 2\epsilon_1, \epsilon_1] \]

where

\[ \epsilon_1 = \frac{1}{4} \sum_{i=0}^{m+1} (\alpha_i^2 - x_i^2)w^i + \sum_{i=q}^{m+1} (1 - \alpha_i)w^i. \]

Rewriting the first sum as an integral (it is at most linear in \( z^i \)) and rearranging gives

\[ \epsilon_1 = \int_{1-\nu}^{1} (1 - \alpha)dx - \sum_{i=q}^{m+1} (1 - \alpha^i)w^i. \]

We see therefore that the quantity \( \epsilon_1 \) indicates how well the quadrature rule approximates

\[ \int_{1-\nu}^{1} (1 - \alpha)dx. \]

The matrix \( M \) will be diagonally dominant and hence positive definite where \( \epsilon_1 \geq 0 \) and so if the quadrature rule under estimates this quantity the resulting numerical scheme will be stable. The modifications to the first and last rows and columns of the mass matrix and the shifted mass matrix do not alter the condition on \( \epsilon_1 \) for the matrix \( M \) to be diagonally dominant. We have determined the regions of instability for a variety of the commonly used quadrature schemes.

One corollary of this analysis is that we can find the regions of instability of a compound scheme provided we know those of the primitive rule, assuming the primitive rule integrates linears exactly. To be more specific, suppose we use a rule with which the scheme is unstable for \( \nu \in [x_1, x_2] \) and we use \( N \) subdivisions. Then with the compound rule the scheme will be unstable when

\[ \nu \in [x_1/N + 2(N - k - 1)/N, x_2/N + 2(N - k - 1)/N), \]

\[ k = 1, \ldots N. \]

This is because with the compound quadrature rule the scheme will have error only for the subdivisions that fall across the nonsmooth part of the integrand. The error will be of the same form as for the primitive rule, as we shall now show.

Suppose the integrand is non–smooth in the \( k \)-th subinterval of the compound rule. Then we have to evaluate

\[ \int_{1-\nu}^{2k/N} (1 - \alpha)dx - \sum_{i=q}^{m+1} f_i w^i / N, \]
where
\[ f_i = 1 - \nu - \frac{1}{2}(-1 + 2(k - 1)/N)(1 - x^i) - \frac{1}{2}(-1 + 2k/N)(1 + x^i). \]
Changing variables leads to the condition that
\[ \int_{1 - \nu_2}^{1} (1 - \nu_2 - x)dx - \sum_{i=q}^{m+1} (1 - \nu_2 - x^i)w^i \]
is non-negative, where \( \nu_2 = N\nu - 2(N - k) \), which is of the same form as the stability condition for the primitive rule. Because with the primitive trapezium rule the scheme is stable, we deduce immediately that with the compound rule it is also stable.

2.3.2 Stability with diffusion

It is straightforward to extend the analysis to include diffusion. The equation for advection-diffusion in one dimension is,
\[ u_t + cu_x - \theta u_{xx} = 0 \]
which, with fully implicit Euler time-stepping leads to the discrete equations,
\[ < U^{n+1}, \phi^i > + \theta \Delta t < U^{n+1}_x, \phi_x^i > = < EU^n, \phi^i >. \]
In analogy with the pure advection case we deduce that a sufficient condition for stability is
\[ M = M(0) + \theta \Delta t S - M(v) \geq 0, \]
where
\[ S = \frac{h}{2}(\frac{2}{h})^2[-1/4, 2/4, -1/4] \]
is a stiffness matrix. The presence of the matrix \( S \) will make the method more stable because \( S \) itself is positive semi-definite. This gives a matrix \( M \) of the form (ignoring boundary conditions)
\[ \frac{h}{2} \left[ \varepsilon_1 - 2\rho, 2(2\rho + \varepsilon_1), \varepsilon_1 - 2\rho \right] \]
where \( \rho = \theta \Delta t/h^2 \). For this problem we can either impose periodic boundary conditions or have zero boundary conditions. If we adopt the former then the
modifications to the matrix $M$ will be of a similar form to made in the case of pure advection which turned out not to affect the positive definiteness of the matrix under consideration. Of course what we really want to know is the minimum value of $\rho$ for which the matrix $M$ is positive semidefinite. The simplest way to do this is to find, for a given quadrature rule, the value of $\nu$ that gives $\epsilon_1$ a negative value of the largest magnitude. One then chooses a value of $\rho$ accordingly to keep the matrix diagonally dominant for that value of $\nu$.

2.3.3 Quadratic elements

The analysis readily extends to higher order elements, although one must do more work to evaluate the entries of the matrix $M(\nu)$. For quadratic elements there are three basis functions with support on each element.

\[
\phi_1(s) = -\frac{1}{2}s(1-s) \\
\phi_2(s) = 1 - s^2 \\
\phi_3(s) = +\frac{1}{2}s(1+s).
\]

Globally there are two types of basis functions. Those which have support on two elements, being made up of a $\phi_1$ and a $\phi_2$ on adjoining elements, will be referred to as type I, while those denoted by $\phi_2$ will be referred to as type II.

First we display the structure of the full matrix $M$:

\[
\begin{array}{cccccc}
  b & d & b & 0 & 0 \\
  a & b & c & b^* & a \\
  0 & 0 & b^* & d & b \\
  0 & 0 & a & b & c & \ldots
\end{array}
\]

Imposing periodic boundary conditions will delete the last row and column of the matrix and make modifications to the first row and column of the matrix. The entries $c, b, b^*, d$ and $a$ are linear combinations of the errors in evaluating the ramp function and higher powers of it using a particular quadrature which is symmetric and evaluates quadratics exactly. We define the following errors:

\[
\epsilon_1 = \int_{1-\nu}^{1} (1 - \alpha) - \sum_{i=q}^{m+1} (1 - \alpha_i)w^i,
\]

\[
\epsilon_2 = \int_{1-\nu}^{1} (1 - \alpha)^2 - \sum_{i=q}^{m+1} (1 - \alpha_i)^2w^i.
\]
\[ \epsilon_3 = \int_{1-\nu}^{1} (1 - \alpha)^3 - \sum_{i=q}^{m+1} (1 - \alpha_i)^3 \omega', \]

and we find that

\[
\begin{align*}
    a &= 2\epsilon_3 + \epsilon_1, \\
b &= -4\epsilon_3 - 3\epsilon_2 - 2\epsilon_1, \\
b' &= -4\epsilon_3 + 3\epsilon_2 - 2\epsilon_1, \\
d &= 8\epsilon_3, \\
c &= 4\epsilon_3 + 6\epsilon_1,
\end{align*}
\]

If the quadrature rule does not integrate quadratics exactly then one must evaluate the integrals using the relevant quadrature rule. The result that the stability of a primitive rule implies the stability of the compound rule does not necessarily follow because the errors do not scale in the same way as for linear elements.

To demonstrate stability, one has to show that the matrix is positive semidefinite. One could use Gershgorin's theorem to see if there are any negative eigenvalues or see if the Rayleigh quotient took on negative values using analysis similar to that used by Wathen [6]. By searching through values of \( \nu \) one can identify the regions of instability. One can then extend the analysis to include diffusion in a manner similar to that carried out with linear elements. We simply increase the value of \( \rho \) until the matrix \( M \) is positive semidefinite for all values of \( \nu \).

\subsection*{2.4 Results}

\subsubsection*{2.4.1 Linear elements}

Below we give the results for linear elements. We present the regions of stability in terms of \( \nu \) and indicate the minimum value of \( \rho \) needed to guarantee stability for all \( \nu \). We only give the regions for \( \nu \in [0,1] \) as the regions for \( \nu \in [1,2] \) can be obtained from them by symmetry. \( N \) is the number of quadrature points.
<table>
<thead>
<tr>
<th>Quadrature rule</th>
<th>N</th>
<th>region of stability</th>
<th>minimum value of $2\rho$</th>
</tr>
</thead>
<tbody>
<tr>
<td>Trapezoidal</td>
<td>2</td>
<td>stable</td>
<td>$3.93 \times 10^{-2}$</td>
</tr>
<tr>
<td>Lobatto</td>
<td>3</td>
<td>[0.0,0.66]</td>
<td>$5.75 \times 10^{-2}$</td>
</tr>
<tr>
<td></td>
<td>4</td>
<td>[0.0,0.32]</td>
<td>$4.35 \times 10^{-2}$</td>
</tr>
<tr>
<td></td>
<td></td>
<td>[0.76,1.0]</td>
<td></td>
</tr>
<tr>
<td></td>
<td>5</td>
<td>[0,0.2]</td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td>[0.46,0.84]</td>
<td></td>
</tr>
<tr>
<td>Gauss</td>
<td>2</td>
<td>[0.62,1.0]</td>
<td>$8.82 \times 10^{-2}$</td>
</tr>
<tr>
<td></td>
<td>3</td>
<td>[0.32,0.78]</td>
<td>$6.96 \times 10^{-2}$</td>
</tr>
<tr>
<td></td>
<td>4</td>
<td>[0.2,0.5]</td>
<td>$3.65 \times 10^{-2}$</td>
</tr>
<tr>
<td></td>
<td></td>
<td>[0.8,1.0]</td>
<td></td>
</tr>
<tr>
<td></td>
<td>5</td>
<td>[0.14,0.34]</td>
<td>$2.76 \times 10^{-2}$</td>
</tr>
<tr>
<td></td>
<td></td>
<td>[0.56,0.86]</td>
<td></td>
</tr>
<tr>
<td></td>
<td>6</td>
<td>[0.1,0.24]</td>
<td>$1.82 \times 10^{-2}$</td>
</tr>
<tr>
<td></td>
<td></td>
<td>[0.42,0.64]</td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td>[0.86,1.0]</td>
<td></td>
</tr>
</tbody>
</table>

The results agree with those of Morton et al. [4] for the special case of $\nu < 1 - \frac{1}{m}$. Notice that the Lobatto rules all have a region of stability for small $\nu$, whilst Gauss rules have a region of instability corresponding to that region. Lobatto rules could therefore be stabilised by choosing the timestep small enough. However this is a rather Draconian measure as by doing so one essentially abandons one of the main advantages of the Lagrange–Galerkin method. Fortunately diffusion also stabilises the scheme. The parameter $\rho$ translates into a mesh Peclet number: in fact $\rho = \max(2, \nu) / \text{Pe}$. Thus the scheme is stable for Peclet numbers of up to 100 for the 6x6 Gauss rule. A glance at the table shows that the values of $\rho$ needed for stability are smaller for Gauss rules than for the corresponding Lobatto rules.

2.4.2 Quadratic elements

We find that for symmetric quadrature rules which evaluate quadratics exactly the method does not appear to identify any regions of stability and, worse still, it would appear that diffusion would never stabilise the scheme because the matrix $M$ is never diagonally dominant. Before drawing conclusions however it is worth remarking that firstly our method of examining the positive definiteness of the error matrix does not give sharp bounds on the eigenvalues, and secondly because the whole analysis relies on using the Cauchy–Schwarz inequality, it may not yield fine enough results. We therefore resort to a Fourier analysis.
2.5 Fourier analysis for the Lagrange-Galerkin method with quadratic basis functions

It is convenient to use the weak formulation to perform the analysis. In the weak formulation we construct test functions on the \( n \)th time level by tracking them back in time from the \( n + 1 \)th time level. These will in general have support on three elements which we label \(-1, 0, 1\) and the solution \( U^n \) defined on these elements we denote \( U_{-1}, U_0 \) and \( U_1 \). We look for Fourier modes of the form

\[
\begin{pmatrix}
\alpha \\
\beta
\end{pmatrix}
\, e^{i\kappa y}
\]

Thus we define

\[
\begin{align*}
U_{-1} &= \alpha(1 - x^2)e^{-2i\nu} + \beta e^{-2i\nu}(x^2 \cos y + ix \sin y) \\
U_0 &= \alpha(1 - x^2)e^{-i\nu} + \beta e^{-i\nu}(x^2 \cos y + ix \sin y) \\
U_1 &= \alpha(1 - x^2)e^{i\nu} + \beta e^{i\nu}(x^2 \cos y + ix \sin y)
\end{align*}
\]

For brevity we denote \( a = x + \nu, \ b = x + \nu - 2, \ a_i = x_i + \nu \) and \( b_i = x_i + \nu - 2 \). Fourier analysis of the finite element equations leads to two independent equations which we call \( I \) and \( II \) and refer to their right hand sides as \( R_I \) and \( R_{II} \). Thus

\[
R_I = \sum_{k=0}^{q-1} w^k(a(1 + a)U_0 - a(1 - a)U_1)/2 + \sum_{k=q}^{m+1} w^k(-b(1 - b)U_0 + b(1 + b)U_{-1})/2,
\]

\[
R_{II} = \sum_{k=0}^{q-1} w^k((1 - a^2)U_0) + \sum_{k=q}^{m+1} w^k((1 - b^2)U_{-1}),
\]

which on rearranging gives

\[
R_I = \sum_{k=0}^{m+1} w^k(a(1 + a)U_0 - a(1 - a)U_1)/2 + \sum_{k=q}^{m+1} w^k(a(1 - a)U_1 + b(1 + b)U_{-1} + U_0(-a - b - a + b^2))/2,
\]

\[
R_{II} = \sum_{k=0}^{m+1} w^k((1 - a^2)U_1) + \sum_{k=q}^{m+1} w^k((1 - b^2)U_{-1} - (1 - a^2)U_1).
\]

To carry out the algebra we used the symbolic manipulator REDUCE. It did not prove possible to obtain any simple closed form expression that gave the
regions of stability or gave necessary and sufficient conditions for stability on
the diffusion, so we had to resort to evaluating the expressions produced by
REDUCE numerically. We finally arrive at an eigenvalue problem for a 2x2
complex matrix. By searching through the values of $\nu$ and $y$ we can identify
the regions of stability. We can also find the minimum value, $\rho_{\text{min}}$, of $\rho$, necessary
to ensure unconditional stability. As for the case of linear elements, we can
express this stability condition in terms of a mesh Peclet number giving $\rho_{\text{min}} =
\max(2, \nu)/Pe$. It is perhaps worth remarking that whilst the value of $\rho_{\text{min}}$ is
sufficient to give stability, in practise, for values of $\rho$ slightly less than $\rho_{\text{min}}$ the
regions of instability might be so small as to make instability unobservable.
Moreover, in a problem where the velocity field is not constant, the Peclet
number $Pe_{\text{max}}$ that the quadrature based Lagrange–Galerkin method requires
for stability is not necessarily the maximum mesh Peclet number in the domain,
as in parts of the domain where the Peclet number is smaller the unstable modes
will be damped.

<table>
<thead>
<tr>
<th>Quadrature rule</th>
<th>$N$</th>
<th>region of stability</th>
<th>minimum value of $\rho$</th>
</tr>
</thead>
<tbody>
<tr>
<td>Trapezoidal</td>
<td>2</td>
<td>stable</td>
<td>$1.83 \times 10^{-2}$</td>
</tr>
<tr>
<td>Lobatto</td>
<td>3</td>
<td>stable</td>
<td>$1.76 \times 10^{-2}$</td>
</tr>
<tr>
<td></td>
<td>4</td>
<td>$[0,0.36]$</td>
<td>$[0.73,1.0]$</td>
</tr>
<tr>
<td></td>
<td>5</td>
<td>$[0,0.2]$</td>
<td>$[0.46,0.85]$</td>
</tr>
<tr>
<td>Gauss</td>
<td>2</td>
<td>unstable</td>
<td>$5.40 \times 10^{-2}$</td>
</tr>
<tr>
<td></td>
<td>3</td>
<td>$[0.33,0.82]$</td>
<td>$3.89 \times 10^{-2}$</td>
</tr>
<tr>
<td></td>
<td>4</td>
<td>$[0.2,0.51]$</td>
<td>$1.07 \times 10^{-2}$</td>
</tr>
<tr>
<td></td>
<td>5</td>
<td>$[0.13,0.35]$</td>
<td>$[0.56,0.88]$</td>
</tr>
<tr>
<td></td>
<td>6</td>
<td>$[0.1,0.25]$</td>
<td>$[0.41,0.66]$</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>$[0.86,0.1]$</td>
</tr>
</tbody>
</table>

The results show an improvement over linear in that the amount of diffusion
needed to stabilise a quadrature rule is less than for the corresponding linear
case. However from this latter viewpoint it would appear that Gaussian rules
possess advantages over Lobatto rules. Indeed, for this simple model case it is
possible to guarantee stability for a Peclet number of at least up to 500 and
working with a CFL number of 5 a Peclet number of 1250.

Returning to the issue of why the energy analysis did not appear to yield
useful results, we note that the examination of the numerical values of $a, b, b^*, c$ and $d$ indicates that the regions of $v$ for which the diagonal elements $c$ and $d$ are greater than zero coincide with the regions of stability as calculated from the Fourier analysis. This seems to indicate that the problem lay not with the actual energy analysis, but with the interpretation of the results it gave, namely that the tests we used to determine whether the matrix $M$ was positive definite were not sharp enough.

For linear advection the analysis extends into two dimensions. However if one includes diffusion it does not because the stiffness matrix is not a tensor product of the one-dimensional stiffness matrices. If one considers the energy method to demonstrate stability it suffices to show that

$$< U^n, U^n > + \rho < \partial_x U^n, \partial_x U^n > + \rho < \partial_y U^n, \partial_y U^n > \leq < EU^n, EU^n >.$$  

If the case were analogous to pure advection, the diffusion from the $x$ partial derivatives would stabilise Fourier modes in the $x$ direction and similarly for the $y$ direction. However one has something slightly better because, except for certain worst case situations, for example when $U^n$ is constant in the $y$ direction or the flow is diagonal to the mesh, diffusion in, say the $x$ direction may stabilise modes propagating in the $y$ direction.

In order to gain some indication of the performance of quadrature in practice, we compared various quadratures on the rotating cone problem. We used a 10x10 grid of biquadratic elements and a time step of 0.02. The runs of 250 time steps correspond to five revolutions. In addition, for two of the quadrature rules runs of 1000 time steps were carried out corresponding to twenty revolutions. We present the results in table (1). The maximum and minimum values of the field $U$ give indications of the stability of the scheme. The sum over the mesh points of the field $U$ gives information concerning the conservation properties of the scheme. Notice however that whilst a cone height of unity after five revolutions would appear at first sight to indicate a scheme with little damping, in actual fact it indicates that the scheme is unstable.

The hybrid rules (a) and (b) corresponded to using a Lobatto rule on odd elements say and a Gauss rule of the same order on even elements. For type (a), a 3x3 Gauss rule was used with a 4x4 Lobatto rule, whilst for type (b), a 4x4 Gauss rule was used with a 5x5 Lobatto rule. 2x2 Gaussian quadrature was found to be unstable. Results for a 5x5 Gauss rules showed that although the maximum height of the cone was acceptable (0.8) the undershoot (-1.7) indicated instability. After 1000 time steps the method had indeed gone unstable. The next table, table (2), tests the stability of the schemes hybrid(a) and
Table 1: Comparison of different quadratures for the rotating cone problem

<table>
<thead>
<tr>
<th>Quadrature rule</th>
<th>$U_{max}$</th>
<th>$U_{min}$ (10$^{-2}$)</th>
<th>$l^2$ (10$^{-2}$)</th>
<th>$\sum U_{ij}$</th>
<th>CPU time</th>
</tr>
</thead>
<tbody>
<tr>
<td>hybrid (a)</td>
<td>0.833</td>
<td>-6.3</td>
<td>6.9</td>
<td>8.62</td>
<td>85.4</td>
</tr>
<tr>
<td>hybrid (b)</td>
<td>0.808</td>
<td>-5.3</td>
<td>7.1</td>
<td>9.25</td>
<td>131.0</td>
</tr>
</tbody>
</table>

3x3 Lobatto  
(Simpson's rule)
| 1x1 subdivisions | 0.236     | -2.1                   | 19.3             | 9.9            | 58.3     |
| 2x2 subdivisions | 1.05      | -19.0                  | 7.64             | 10.4           | 150.0    |
| 3x3 subdivisions | 0.83      | -11.1                  | 7.46             | 9.19           | 307.4    |

4x4 Lobatto
| 1x1 subdivisions | 2.81      | -245                   | 76.1             | 10.20          | 99.6     |
| 2x2 subdivisions | 0.817     | -5.8                   | 6.9              | 9.27           | 284.5    |
| 3x3 subdivisions | 0.811     | -5.4                   | 7.0              | 9.24           | 627.0    |

5x5 Lobatto
| 1x1 subdivisions | 0.827     | -25.6                  | 10.1             | 10.2           | 160.0    |
| 2x2 subdivisions | 0.811     | -5.35                  | 7.0              | 9.24           | 599.6    |

4x4 Gauss
| 1x1 subdivisions | 0.992     | -26.0                  | 10.3             | 8.20           | 103.8    |
| 4x4 subdivisions | 0.817     | -5.92                  | 7.0              |                |          |

Area Weighting
| 1x1 subdivisions | 0.492     | -3.47                  | 12.0             |                |          |
| 2x2 subdivisions | 0.712     | -3.61                  | 8.35             |                |          |
| 4x4 subdivisions | 0.789     | -4.91                  | 7.3              |                |          |

Trapezium rule
| 10x10 subdivisions | 0.682     | -3.5                   | 9.11             | 9.03           | 763.0    |
hybrid(b). The mesh used was as described above and the runs were for 1000 time steps corresponding to 20 revolutions.

Table 2: Further tests of the hybrid quadrature rules on the rotating cone problem

<table>
<thead>
<tr>
<th>Quadrature rule</th>
<th>$U_{max}$</th>
<th>$U_{min}(10^{-2})$</th>
<th>$l^2(10^{-2})$</th>
<th>$\sum U_{ij}$</th>
<th>CPU time</th>
</tr>
</thead>
<tbody>
<tr>
<td>hybrid (a)</td>
<td>0.583</td>
<td>-9.1</td>
<td>11.6</td>
<td>7.04</td>
<td>311.7</td>
</tr>
<tr>
<td>hybrid (b)</td>
<td>0.583</td>
<td>-6.0</td>
<td>11.2</td>
<td>8.72</td>
<td>502.6</td>
</tr>
</tbody>
</table>

In conclusion, the 5x5 Lobatto appears to be the most accurate of the standard quadrature rules tested for the time step chosen. Although the hybrid schemes demonstrate favourable properties on this problem, the more accurate of the two gave disappointing results on another test problem tried at Harwell (see below). For the problem tried here the Lobatto rule appeared to be accurate and stable, even without any diffusion. This is because the Lobatto rule possesses a region of stability for small CFL numbers. For this problem it would seem that the CFL number is on average within the stability limit. Gaussian quadrature appears to be quite promising for problems where there is diffusion present because for a given number of quadrature points stability for a large range of mesh Peclet numbers is guaranteed.

3 Subdivision: a stable implementation of the method

Because of the potential instability of the Lagrange-Galerkin method when quadrature is used, Morton et al [4] devised an alternative method known as area-weighting. Instead of tracking quadrature points the strategy was to track the centroids of elements and approximate the flow map by assuming that the element was transported unrotated and undistorted so that the inner products of functions defined on the element with functions on the underlying mesh could be calculated exactly. For this to be done in a simple manner the mesh had to be a tensor product of one-dimensional meshes so that the regions of intersection between the transported element and the underlying mesh were rectangular. The stability of this scheme was established for linear advection and its accuracy demonstrated on the "rotating cone" problem. Unfortunately if one desires to work with completely unstructured meshes it is difficult to generalise area-weighting whilst retaining all of its desirable properties.
The weak formulation of the Lagrange-Galerkin method is by definition,

\[(U^{n+1}, \phi_i) = \int U^n(x)\phi_i(y(x))dx\]

\[y(x) = X(x, t; t^{n+1}),\]

where \(X(x, t; t^{n+1})\) was defined in section 2.

There are a number of ways to approximate the right hand-side of this. The usual one is, of course, to track the quadrature points forward in time and interpolate for the basis functions on the element where the point lands. Another option is to track basis functions backward in time and integrate over the support of the tracked-back basis functions. In this latter approach the test function \(\phi_i(y)\) is interpreted as a test function at the \((n+1)th\) time level tracked back in time to give a new test function \(\chi(x) = \phi(y(x))\). The nodes of the grid are tracked backwards in time to create a grid on the previous time level. The tracked-back grid defines these new test functions up to an isoparametric transformation and therefore to construct the right hand-side we integrate these test functions against the advected field defined on the original grid at the previous time level.

If one simply applied a quadrature rule to these tracked back elements then one would be up against the same problem that Lagrange-Galerkin method with standard quadrature faces - the scheme would, without diffusion, be unstable. If however one followed the analogy with area-weighting and attempted to integrate the tracked back basis functions against the field defined on the underlying grid, then one would have a stable scheme which would also be more accurate than area-weighting as one is making a better approximation to the flow map. The practical problem is that the intersections are often of very complicated shapes, and it is a nontrivial task to find them. If one has biquadratic isoparametric elements then the sides of the elements are in general curved, and the regions of intersection, or "subdivisions", are in general quite distorted, and as they may sometimes have over four sides with a centroid which may not lie inside the subdivision, the task of integrating over these subdivisions, once they are found, is also nontrivial. In what follows we shall refer to the tracked back grid as the "red" grid and the fixed grid as the "black" grid. Similarly we refer to "red" elements, nodes, and so on.

The basic algorithm is as follows:

Track back nodes of the black grid to form the red grid.
For each element of the red grid
create subdivisions
For each subdivision
  integrate the basis functions defined on
  red element against the
  field defined on the black grid

  Add the contributions to the element RHS
Endfor
Add element RHS to global RHS
Endfor

We now describe our method of finding the subdivisions. Given a red element, we find the black element into which the first node on the first side of the red element falls. This black element becomes our "current" element, and the side of the red element our "current" side. We find the first intersection along the current side of the current side with a side of the black element (the "target" side), and store coordinates defining the curve segment that we have found. The target face becomes the current face and the black element our new current element. If there is no intersection then the side must end in the current element and the end of the side defines the end of the curve segment, and we proceed using the next side of the red element until an intersection is found. We also store the number of the black element into which the first red current side goes into on first leaving the current element. We continue in this way until we reach the point at which we started, in which case the first subdivision has been processed. We then continue processing the element stored previously until we have processed all of the elements which have intersections with the sides of the red element. Black elements that fall completely inside the red element are then integrated over. We illustrate the method in the figure below. Once the subdivisions had been constructed, the integrals over each of them could be performed by subdividing the subdivision into triangles or quadrilaterals and after an isoparametric transformation using a standard quadrature rule. As the algorithm is quite expensive we tested it against quadrature in order to see whether there was any considerable benefit in using it. Quadrature will be most inaccurate on a pure advection problem so we chose to compare subdivision and quadrature on a linear advection problem with a velocity field generated by the double glazing problem. This should give a worst case test for quadrature as against subdivision.
3.1 Flow field definition

The equations for free convection in the Boussinesq approximation were solved on a 7x7 grid using Q9/3 elements (piecewise continuous biquadratic elements for velocity and temperature with discontinuous piecewise linear elements for pressure). A Raleigh number of 1000 was used with a Prandtl number of 0.72.

3.2 Advection problem

The equations for linear advection were solved using the flow field, \( a(x) \), calculated by the previous problem:

\[
\phi_t + a \nabla \phi = 0.
\]

A 7x7 grid of biquadratic elements was used. The initial datum was

\[
\phi(x, 0) = \begin{cases} 
\cos 4\pi r & \text{if } r < 0.125 \\
0 & \text{otherwise,}
\end{cases}
\]

where \( r^2 = (x - 0.5)^2 + (y - 0.125)^2 \). The aim of the exercise was to compare the stability properties of subdivision and quadrature. We performed one hundred time steps with a time step of 0.01. We compared 6x6 Lobatto quadrature, subdivision and a 6x6 Lobatto / 5x5 Gauss hybrid rule.
Figure 2: Streamline plot of velocity field

We plotted contour plots of $\phi$ every 10 time steps, and also plots of the grid with the subdivisions superimposed. For the purposes of the plotting the sizes of the subdivisions were scaled in order to enable them to be distinguished clearly from the grid.

3.3 Results

We show the contours of the streamlines in Fig. 2, and plots of the subdivisions on a 3x3 grid and a 7x7 grid in Fig 3 and 4 to illustrate the complexity of the procedure and the degree to which elements are distorted when tracked back in the flow field.

One of the clearest indications of the instability of the Lagrange-Galerkin method is the magnitude of the under and overshoots. We present the maximum and minimum of the scalar field in the table below.

<table>
<thead>
<tr>
<th>subdivision</th>
<th>min $\phi$</th>
<th>max $\phi$</th>
</tr>
</thead>
<tbody>
<tr>
<td>subdivision</td>
<td>-1.73E-1</td>
<td>3.31E-1</td>
</tr>
<tr>
<td>6x6 Lobatto</td>
<td>-2.20E-1</td>
<td>3.64E-1</td>
</tr>
<tr>
<td>hybrid</td>
<td>-3.07E-1</td>
<td>2.50E-1</td>
</tr>
</tbody>
</table>

The present grid is really too coarse to give an accurate solution to the advection problem. If we take the subdivision results as a bench mark (since we
Figure 3: Plot of tracked-back grid for 3x3 mesh

Figure 4: Plot of tracked-back grid for 7x7 mesh
know that subdivision will not introduce any instability) we see that whilst the hybrid quadrature results are disappointing, the results for the Lobatto quadrature compare very well with the subdivision results. The quadrature based scheme was much cheaper, by a factor of about ten. The tracking of the grid for the subdivision based scheme took about 0.45s while the subdivision and integration took approximately 4.6s and 2.2s respectively. The tracking time for the quadrature scheme is the same, and the integration cost about 0.5s. No optimization has been carried out so that these figures can only be really used to give a rough comparison between the two methods and not benchmark figures as to their absolute performance. Our actual method of using quadrature was to integrate over the tracked back elements rather than tracking back quadrature points. This has the advantage that only four points per element need be tracked rather than say 36 for a 6x6 Gauss rule.

3.4 Assessment of subdivision

Although subdivision was accurate and stable, in practise it suffered from a number of disadvantages.

- Subdivision is much slower than quadrature. The actual time to construct the subdivisions is a costly overhead and the integration over the subdivisions is also costly as more points per element must in fact be used. Whereas in using quadrature one uses a single rule, with subdivision, each subdivision must be integrated over with the same degree of accuracy.

- The current subdivision algorithm is not very robust.

- Subdivision is actually not exact in that the flow map is approximated by an isoparametric transformation. This introduces some element of artificial diffusion. If one tracks the quadrature points themselves one avoids this problem.

- Because subdivision relies on tracking the grid, severe distortions of the grid caused by for example a timestep being too large will destroy the accuracy of the method.

- The algorithm is difficult to extend to 3D.

It is perhaps worthwhile discussing why the algorithm used is not robust. The present version of the algorithm runs into problems if
• intersections occur near the ends of element sides on either the red or black grids.

• the distance between intersections is very small.

• the intersections occur very obliquely.

One practical detail was that because the sides of the elements were curved, intersections could be known only up to a finite precision and so in border line cases it was often unknown whether an intersection had occurred or not, or whether the intersection found was actually the required one. This is inevitable with this sort of problem, and one must use tolerances. Because of the sequential nature of the algorithm, however, decisions about intersections were crucial, as these determined the future course of the algorithm. The problem seemed to come from using analytic information to directly determine the next step of the algorithm. Decisions involving floating point arithmetic determined for example the next current element or the next current face. If one of the tests gave an invalid result then the subdivision of an element would fail: the algorithm may continue but there would be little chance of knowing if an error had occurred and then recovering. Besides all this the algorithm had not been specified precisely enough to cope with such things as intersections upon nodes.

The moral that can be learned from this is one which is not new at all, but one that has been emphasised by computer scientists for quite some time. It is that for any piece of software the details of the algorithm must be fully specified in the design stage, and that includes even the most pathological special cases.

To summarise, although subdivision appears at first sight to have the advantages such as stability and accuracy, in actual practice its cost is so great that even without diffusion one could simply use a quadrature rule with a sufficiently large number of points and obtain a scheme that could for all practical purposes be considered stable. If one then uses the option of tracking quadrature points rather than nodes one would have a more robust and accurate scheme than subdivision. Subdivision does not define the Lagrange–Galerkin method, but rather quadrature in the limit of a large number of quadrature points. With diffusion, quadrature is unconditionally stable anyway providing Peclet numbers are not too large, and the limiting Peclet number increases with the number of quadrature points.
4 Conclusions

Previous analysis of the Lagrange–Galerkin method for pure advection [4] showed that the use of quadrature produced, in general, only conditionally stable schemes. We have shown that quadrature, provided that sufficient care is taken, is still a viable method for computing the inner products peculiar to the Lagrange–Galerkin method for advection–diffusion problems. We have shown that if diffusion is present in a problem then stability is restored. Quadratic elements had better stability properties in this respect than linear elements. Our investigation of an alternative method of computing the Lagrange–Galerkin RHS inner products showed that quadrature, despite its limitations, still is an effective method.

References


