Journal of Computational Physics **172,** 356–386 (2001)
doi:10.1006/jcph.2001.6829, available online at http://www.idealibrary.com on IDEAL®

Nonlinear Particle Tracking for High-Order Elements

G. Coppola,[1] S. J. Sherwin,[2] and J. Peiró

Department of Aeronautics, Imperial College of Science, Technology and Medicine, London SW7 2BY, United Kingdom
E-mail: gianfilippo.coppola@yale.edu; s.sherwin@ic.ac.uk; j.peiro@ic.ac.uk

Received December 19, 2000; revised April 30, 2001

The problem of calculating particle trajectories on unstructured meshes using a high-order polynomial approximation of the velocity field is addressed. The calculation of the particle trajectory is based on a Runge–Kutta integration in time. A convenient way of implementing high-order approximations is to employ an auxiliary mapping that transforms a finite element into a topologically equivalent parent element within a normalized parametric space. This presents two possible choices of space in which to perform the time integration of the particle position: the physical space or the parametric space. We present algorithms for implementing both particle tracking strategies using high-order elements and discuss their merits. The main drawback of both methods is their reliance on nonlinear procedures to calculate the particle trajectory. A novel alternative hybrid approach that advances a particle in both the physical and the parametric space without requiring nonlinear iterations is proposed. The error introduced by the alternative linearized procedures and their effect in the rate of convergence of the time integration is discussed. Finally, the performance of the different algorithms is compared using a set of analytical and computational, linear and high-order, velocity fields. © 2001 Academic Press

Key Words: particle tracking; high-order elements; flow visualization.

1. INTRODUCTION

Computer visualization is a powerful tool for the processing and interpretation of the large amounts of geometrical and flow field data currently produced in computational fluid dynamics (CFD) simulations. It permits comparisons between computational and experimental data and eases the identification of important flow features such as vortical flows,

[1] Presently at Department of Engineering and Applied Science, Yale University, New Haven, CT 06520.
[2] Corresponding author.

stagnation regions, and characteristic flow lines. A review of scientific visualization applied to CFD until 1990 is presented in [4].

The emphasis of this paper is on the integration of particle paths using CFD data from unstructured high-order finite element/spectral solvers. The calculation of particle paths is a useful visualization technique for understanding flow behavior. It is also as an integral part of Lagrangian and semi-Lagrangian CFD flow simulations of non-Newtonian fluids where the modelling of history effects such as exposure to shear stresses is important. Examples of such fluids can be found in [10].

Particle tracking for both structured and unstructured meshes is available in the majority of packages for scientific visualization, e.g., Visual3 [6], PLOT3D [15], AVS [14], Tecplot [1], and OpenDX [7] to name but a few. These methods typically assume a piecewise linear representation of the velocity field. A good review of the work on particle tracking for such representations is given by Darmofal and Haimes in [3] where they compare multistep and multistage methods for time integration and discuss their accuracy, stability, and performance.

Although high-order polynomial approximations of the velocity field are common in finite element simulation of incompressible flows, the problem of calculating flow lines for these high-order representations has received little attention to date. Two early examples of particle tracking for high-order elements can be found in the literature of non-Newtonian flows [5, 13]. These references deal with the numerical simulation of two-dimensional non-Newtonian flows using constitutive equations for the stress tensor based on the strain history calculated along streamlines. The integration in time is accomplished using the standard fourth-order Runge–Kutta method. This time integration is applied in physical space with a quadratic approximation of the velocity on a triangular mesh in [13]. Reference [5] performs the time integration on the parametric space of the parent element using unstructured quadrilateral meshes with a quadratic approximation of velocity. A Newton–Raphson iteration is applied to find the intersection with boundaries. Neither of these articles include detailed information on the implementation of the particle tracking method.

The approximation of a high-order velocity field by piecewise linear polynomials and the use of a standard particle tracking algorithm might not result in an accurate calculation of the trajectories. Streamline integration is very sensitive to small changes in kinematics. Inconsistencies between the high-order velocity field and a linear representation might result in kinematic changes that could have a significant effect on the pathlines in complex regions which are very sensitive to the spatial resolution, such as recirculation cells. This is highlighted in Fig. 1 which compares the particle traces for a high-order velocity field calculated by the commercial package Tecplot [1] and the algorithm proposed in this paper. The streamline is computed by Tecplot using a linear interpolation between points. The inconsistency between the two approximations leads to a linearly interpolated streamline that differs form the high-order one in the recirculation cell. Figure 1 shows the whole domain on the left and two views of the junction to highlight the diverging streamlines.

The layout of this paper is arranged as follows. The basic notation for the description of finite element representations of velocity fields and multistage time integration is introduced in Section 2. Section 3 presents two strategies for nonlinear particle tracking using high-order elements and their implementation. Section 4 presents an alternative hybrid approach that advances a particle across elemental boundaries without resorting to nonlinear iterations: the *guided search*. It discusses its accuracy and implementation, and proposes various viable schemes for particle tracking which employ the guided search. Finally,

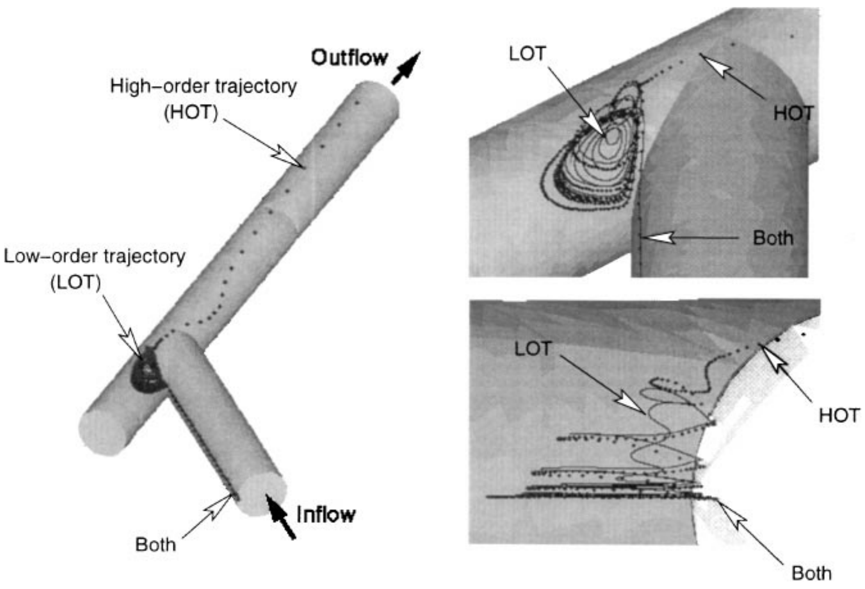


FIG. 1. Streamlines in a junction between two straight pipes. The high-order finite element mapping is based on a seventh-order polynomial expansion and the corresponding streamline is represented by the dotted line. The solid line is the streamline starting at the same point as calculated by the commercial package Tecplot using a subdivision of the mesh of high-order elements into linear elements. (Remark: One high-order element was divided into 84 linear elements).

Section 6 presents some examples of application of the proposed algorithms to both analytical and computational velocity flow fields and within a complex curvilinear geometric model.

2. PARTICLE TRACKING: PROBLEM DEFINITION

The problem of finding the trajectory $\mathbf{x}(t)$ of a particle is formulated as a set of ordinary differential equations

$$\frac{d\mathbf{x}}{dt} = \mathbf{u}(\mathbf{x}, t), \tag{1}$$

where $\mathbf{x}$ is the position in space and $\mathbf{u}$ is the velocity field. This simply states that the tangent to the trajectory curve at a point of coordinates $\mathbf{x}$ is parallel to the velocity $\mathbf{u}$ at that point. The initial condition for this problem amounts to specifying the position $\mathbf{x}_0$ of the particle at a specific time, $t = t_0$, i.e.,

$$\mathbf{x}(t_0) = \mathbf{x}_0. \tag{2}$$

2.1. *Numerical Representation of the Velocity Field*

The velocity field $\mathbf{u}$ is generally not available analytically but through a discretized form. In the finite element method, this field is defined as a piecewise continuous expansion over elemental regions. In this paper we shall consider flow fields generated by both linear and high-order spectral/hp element methods.

Before dealing with the problem of integrating Eq. (1) in time using a finite element representation of the velocity field, we will introduce some basic nomenclature relating to the finite element method.

2.1.1. Isoparametric finite elements. In a general finite element formulation the computational domain Ω is divided into N_e nonoverlapping subdomains Ω^e, or finite elements, such that

$$\Omega = \cup_e \Omega^e \quad e = 1, \ldots, N_e,$$

and which are topologically identified with triangular and quadrilateral subdomains in two dimensions and tetrahedral, prismatic, pyramidal, and hexahedral subdomains in three dimensions.

It is standard practice to map a subdomain or element Ω^e in physical space $\mathbf{x} = (x_1, \ldots, x_n)$, where n denotes the number of dimensions, onto a straight-sided subdomain in $\boldsymbol{\zeta} = (\zeta_1, \ldots, \zeta_n)$ with the same topology but fixed dimensions, called the *parent element*. This operation is illustrated in Fig. 2 where a generic high-order element (solid line) and a linear element (dashed line) are drawn for comparison. In both cases, the mapping is an homeomorphism between the element in the physical space and the parent element, and can be expressed by the analytical mapping in $\boldsymbol{\zeta}$

$$\mathbf{x} = \chi^e(\boldsymbol{\zeta}), \tag{3}$$

and the corresponding inverse, typically nonanalytical, mapping will be denoted by

$$\boldsymbol{\zeta} = \boldsymbol{\Psi}^e(\mathbf{x}). \tag{4}$$

A significant difference between linear and high-order (nonlinear) elements is that the mapping for linear elements has an analytical inverse while for the high-order elements, the inverse of the nonlinear mapping has to be determined iteratively, in general.

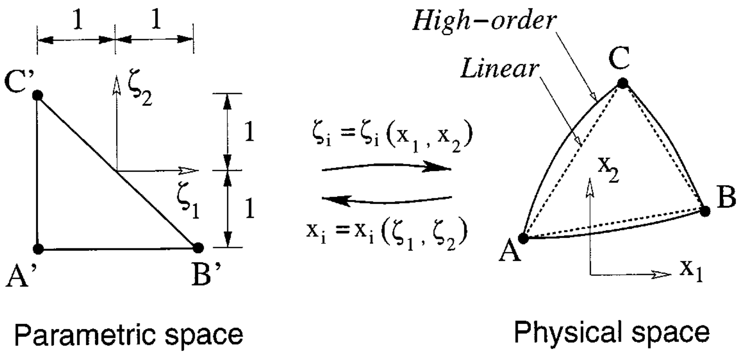


FIG. 2. Two-dimensional isoparametric mapping between the physical and the parametric spaces. The triangle $\widehat{ABC}$ on the right is in the physical space and the triangle $\widehat{A'B'C'}$ is its projection in the parametric space known as the parent element. The dashed lines represent triangle $\widehat{ABC}$ with straight sides.

In the *isoparametric* approach the coordinate system is expressed as a mapping (3) symbolically written as

$$\mathbf{x} = \sum_I \hat{\mathbf{x}}_I \phi_I(\boldsymbol{\zeta}), \tag{5}$$

where ϕ_I represents a member of a complete family of polynomial expansion bases, $\mathbf{x}$ is the vector of physical space coordinates, $\boldsymbol{\zeta}$ is the vector of local Cartesian coordinates in the parametric space, and the quantities $\hat{\mathbf{x}}_I$ represent the expansion coefficients. If a nodal expansion is used where the basis ϕ_I has the Kronecker delta property

$$\phi_I(\hat{\mathbf{x}}_J) = \begin{cases} 1 & \text{if } I = J \\ 0 & \text{if } I \neq J \end{cases}, \tag{6}$$

then the expansion coefficients $\hat{\mathbf{x}}_I$ coincide with the physical coordinates of the mesh nodes $\mathbf{x}_I$. The vector field is approximated in a similar fashion as

$$\mathbf{u}(\mathbf{x}, t) \approx \tilde{\mathbf{u}}(\mathbf{x}(\boldsymbol{\zeta}), t) = \sum_I \hat{\mathbf{u}}_I(t)\phi_I(\boldsymbol{\zeta}), \tag{7}$$

where the quantities $\hat{\mathbf{u}}_I$ denote the expansion coefficients representing the velocity.

Over the parent element, the shape functions can be represented as a sum of a finite number of polynomials (the expansion basis), whose number and shape depend on the topology of the subdomain and on the required polynomial order of the expansion. For instance, the linear polynomial expansion basis on a triangular region consists of three linear polynomials. Each polynomial is associated to a different vertex of the triangle with a value of 1 at that vertex and a value of 0 at the other two vertices.

2.1.2. Spectral/hp element method. An important feature of a high-order unstructured formulation is computational efficiency. The efficient construction of an expansion basis in an unstructured subdomain can be achieved through the introduction of a new coordinate system, referred to as *auxiliary space* or *collapsed coordinate system* [11]. The mapping between the parameteric space identified by the local Cartesian coordinate, $\boldsymbol{\zeta}$, and the auxiliary space represented by, $\boldsymbol{\eta}$, is denoted by

$$\boldsymbol{\zeta} = \boldsymbol{\zeta}(\boldsymbol{\eta}). \tag{8}$$

A two-dimensional example of this mapping is depicted in Fig. 3.

The advantage of such auxiliary mapping is that the local coordinates $\boldsymbol{\eta}$ are now defined in a square region $-1 \leq \eta_i \leq 1$. This permits the definition of an expansion basis as a generalized tensor-product of one-dimensional expansions which can be evaluated in a computationally efficient manner. The approximation of the velocity within a two-dimensional element is given by

$$\mathbf{u}^e[\chi^e(\boldsymbol{\zeta})] = \sum_{p,q} \hat{\mathbf{u}}_{pq}\tilde{\phi}_{pq}(\zeta_1, \zeta_2) = \sum_{p,q} \hat{\mathbf{u}}_{pq}\phi_p(\eta_1)\phi_{pq}(\eta_2), \tag{9}$$

where $\phi_p(\eta_1)$ and $\phi_{pq}(\eta_2)$ represent appropriate one-dimensional polynomials in η_1 and η_2, respectively. Alternatively, the approximation can also be written in terms of the Lagrange

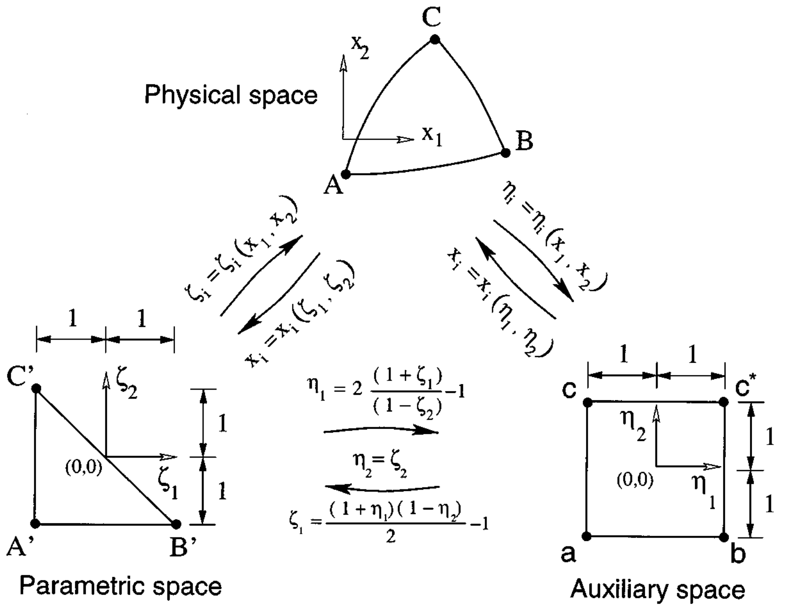

C
x_2
Physical space
x_1
B
A
$\zeta_i = \zeta_i(x_1, x_2)$
$x_i = x_i(\zeta_1, \zeta_2)$
$\eta_i = \eta_i(x_1, x_2)$
$x_i = x_i(\eta_1, \eta_2)$
1 1
C'
ζ_2
(0,0)
ζ_1
1
1
A'
B'
Parametric space
$\eta_1 = 2\frac{(1+\zeta_1)}{(1-\zeta_2)} - 1$
$\eta_2 = \zeta_2$
$\zeta_1 = \frac{(1+\eta_1)(1-\eta_2)}{2} - 1$
1 1
c
c*
η_2
(0,0)
η_1
1
1
a
b
Auxiliary space

FIG. 3. Two-dimensional example of the various mappings involved in the interpolation of functions using unstructured high-order spectral/*hp* elements. The mapping between the physical and parametric spaces corresponds to the standard isoparametric finite element representation. The mapping between the auxiliary and parametric spaces is characteristic of the spectral/*hp* element approach and is introduced for computational efficiency.

polynomial as

$$\mathbf{u}^e[\chi^e(\boldsymbol{\eta})] = \sum_{p,q} \mathbf{u}_{pq} h_p(\eta_1) h_q(\eta_2), \tag{10}$$

where $h_p(\eta_1)$ and $h_q(\eta_2)$ are the Lagrange polynomials through a set of points typically chosen to be the quadrature points. In this representation the coefficients $\mathbf{u}_{pq}$ are the solution values at the quadrature points resulting from the Kronecker Delta nature of the Lagrange polynomial basis. This form is particularly convenient for interpolation where the Kronecker Delta leads to implementation efficiency.

The new coordinate system introduces a singularity where the Jacobian of the inverse transformation,

$$\boldsymbol{\eta} = \boldsymbol{\eta}(\boldsymbol{\zeta}), \tag{11}$$

can be multivalued but the coordinates are well-defined and bounded over the singular region [11]. The main advantage of this generalized tensor-product formulation is that, by using a sum factorization technique [11], the evaluation of expression (9) at a series of P^2 quadrature points requires only $O(P^3)$ operations in 2D, and $O(P^4)$ in 3D, if $\phi_p(\eta_1)$ and $\phi_{pq}(\eta_2)$ are polynomials of order P. A straightforward evaluation would have resulted in an operation count of $O(P^4)$ in 2D and $O(P^6)$ in 3D.

2.2. *Time Integration Schemes*

There are two main types of integration schemes for ordinary differential equations, multistage methods (mainly Runge–Kutta type), and multistep methods. The literature on the subject suggests, e.g., [9], that multistep methods are more efficient and accurate in problems where the velocity field is smooth. However, they require a start-up procedure and their time step cannot be changed easily. On the other hand, a multistage algorithm does not require a start-up procedure, the time step can be changed easily and is very robust. However, it is computationally more expensive, since it requires more function evaluations. Given that the velocity fields in our work are defined in a piecewise continuous fashion and might represent sharp velocity gradients, even flow discontinuities, we have chosen to adopt the Runge–Kutta schemes because of their flexibility and robustness.

The application of an s-stage explicit Runge–Kutta method to the general system of ordinary differential equations

$$\frac{d\mathbf{y}}{dt} = \mathbf{f}(\mathbf{y}, t), \tag{12}$$

with the initial conditions

$$\mathbf{y}(t_0) = \mathbf{y}_0, \tag{13}$$

results in the iteration

$$\mathbf{y}^{n+1} = \mathbf{y}^n + \Delta t \sum_{i=1}^{s} b_i \mathbf{f}_i, \tag{14}$$

where $\mathbf{y}^n$ denotes the value $\mathbf{y}(t^n)$,

$$\mathbf{f}_i = \mathbf{f}\left(\mathbf{y}^n + \Delta t \sum_{j=1}^{i-1} a_{ij}\mathbf{f}_j, t^n + c_i \Delta t\right), \tag{15}$$

s is the number of stages and Δt is the time step. The values b_i, c_i, and a_{ij} are the entries of the corresponding *Butcher array* [9]:

$$\begin{array}{c|ccc} c_1 & a_{11} & \cdots & a_{1s} \\ \vdots & \vdots & & \vdots \\ c_s & a_{s1} & \ldots & a_{ss} \\ \hline & b_1 & \cdots & b_s \end{array}. \tag{16}$$

The coefficients of this array define the particular scheme employed and are given, for the schemes employed in this paper, at the end of this section.

The second term on the right-hand side of Eq. (14) is a weighted average of the values $\mathbf{f}_i$ taken at each stage. Hence, if we set

$$\bar{\mathbf{f}} = \sum_{i=1}^{s} b_i \mathbf{f}_i, \tag{17}$$

then a generic Runge–Kutta scheme, as represented by Eq. (14), can be considered as a Euler scheme that marches in time using an averaged value of $\mathbf{f}$. The same consideration can be applied to each stage, hence the second term on the right-hand side of Eq. (15) can be considered as an Euler step taken with average velocity

$$\hat{\mathbf{f}}_i = \sum_{j=1}^{i-1} a_{ij}\mathbf{f}_j. \tag{18}$$

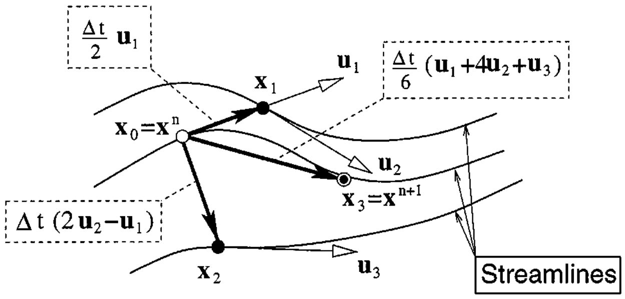

$\frac{\Delta t}{2}\mathbf{u}_1$
$\mathbf{u}_1$
$\frac{\Delta t}{6}(\mathbf{u}_1+4\mathbf{u}_2+\mathbf{u}_3)$
$\mathbf{x}_1$
$\mathbf{x}_0=\mathbf{x}^n$
$\mathbf{u}_2$
$\mathbf{x}_3=\mathbf{x}^{n+1}$
$\Delta t\,(2\mathbf{u}_2-\mathbf{u}_1)$
$\mathbf{x}_2$
$\mathbf{u}_3$
Streamlines

FIG. 4. Interpretation of the Runge–Kutta integration as a series of Euler steps using a suitably averaged velocity. The figure shows the steps for a three-stage scheme.

This interpretation of the Runge–Kutta scheme is depicted in Fig. 4 for the three-stage scheme which shows how each stage of the Runge–Kutta integration can be envisaged as a Euler step in the direction of an suitably averaged velocity. We will use this interpretation as the basis for the *guided search* algorithm which will be developed in the next sections.

Four Runge–Kutta integration methods have been investigated in this paper which provide a wide range of orders of accuracy. These are the following:

1. RK1: A single-stage scheme, the Euler method. The entries of its Butcher array in Eq. (16) are

$$\begin{array}{c|c} 0 & 0 \\ \hline & 1 \end{array}. \tag{19}$$

2. RK2: A two-stage scheme, the improved Euler method, with Butcher array entries:

$$\begin{array}{c|cc} 0 & & \\ 1 & 1 & \\ \hline & \frac{1}{2} & \frac{1}{2} \end{array}. \tag{20}$$

3. RK3: A three-stage scheme, Kutta's third-order formula, with a Butcher array given by

$$\begin{array}{c|ccc} 0 & & & \\ \frac{1}{2} & \frac{1}{2} & & \\ 1 & -1 & 2 & \\ \hline & \frac{1}{6} & \frac{2}{3} & \frac{1}{6} \end{array}. \tag{21}$$

4. RK4: A four-stage scheme, the classical Runge–Kutta method, with Butcher array entries:

$$\begin{array}{c|cccc} 0 & & & & \\ \frac{1}{2} & \frac{1}{2} & & & \\ \frac{1}{2} & 0 & \frac{1}{2} & & \\ 1 & 0 & 0 & 1 & \\ \hline & \frac{1}{6} & \frac{1}{3} & \frac{1}{3} & \frac{1}{6} \end{array}. \tag{22}$$

3. PARTICLE TRACKING FOR HIGH-ORDER ELEMENTS

There are two possible strategies for tracking particles depending on the space in which the time integration is performed. The first strategy is to perform the time integration in the physical space. Given the piecewise continuous definition of the flow field, this process involves searching for the element containing the point where the velocity is to be evaluated followed by the interpolation of the velocity using the expansion basis within the element. The two computationally intensive operations to be performed are:

(i) a nonlinear iterative procedure to find the local coordinates $\boldsymbol{\zeta}$ in the parametric space from the Cartesian coordinates $\mathbf{x}$ in physical space, and
(ii) the interpolation of the velocity $\mathbf{u}$ at a point of parametric coordinates $\boldsymbol{\zeta}$.

The second strategy performs the time integration in the parametric space. The process involves advancing the particle within an element using a transformed velocity field in the parametric space until the particle reaches the element boundary. The process is then continued in the neighbor element sharing the boundary where the particle exits. The two main operations to be performed are:

(i) the interpolation of the velocity $\mathbf{u}_\zeta$ at a point of coordinates $\boldsymbol{\zeta}$ in the parametric space, and
(ii) a nonlinear iterative procedure to find the intersection of a pathline with an elemental boundary.

The computational cost of either integration scheme can be very high when applied to the calculation of particle trajectories on a high-order flow field. We therefore wish to devise efficient algorithms which take into consideration the computational cost of spatial interpolation and nonlinear mappings.

3.1. *Particle Tracking in the Physical Space*

We recall that the explicit Runge–Kutta scheme (14) applied to the particle trajectory equation (1) results in

$$\mathbf{x}^{n+1} = \mathbf{x}^n + \Delta t \sum_{i=1}^{s} b_i \mathbf{u}_i \tag{23}$$

where

$$\mathbf{u}_i = \mathbf{u}\left(\mathbf{x}^n + \Delta t \sum_{j=1}^{i-1} a_{ij} \mathbf{u}_j, t^n + c_i \Delta t\right). \tag{24}$$

For a fixed time step Δt, the iterative algorithm at a certain stage n, can be summarized as follows:

1. Find the local coordinates $\boldsymbol{\zeta}_n$ (point P' in Fig. 5) corresponding to the coordinates $\mathbf{x}_n$ of the current position (point P in Fig. 5) using the inverse iteration to be discussed in Section 3.1.1.
2. Interpolate the velocity field $\mathbf{u}$ at point $\boldsymbol{\zeta}_n$ using Eq. (10).
3. Apply the time integration scheme represented by Eqs. (23) and (24), to find the next position coordinates $\mathbf{x}_{n+1}$ (point Q in Fig. 5).
4. If the stopping criterion is satisfied then exit, or else let $n = n + 1$ and return to step 1.

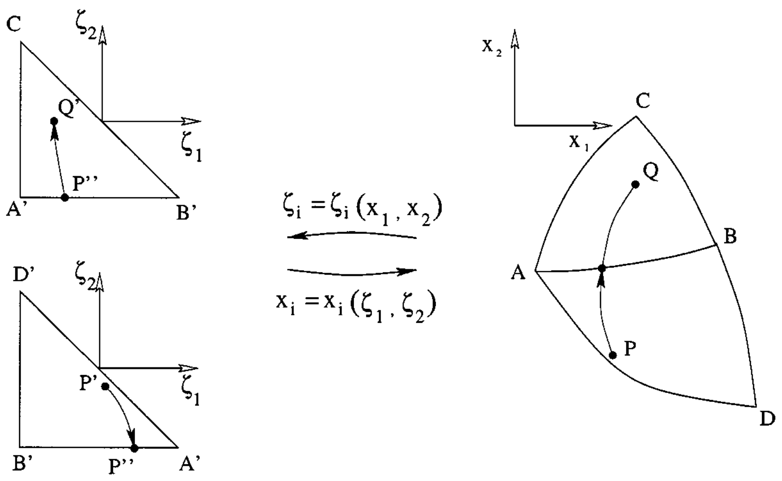


FIG. 5. Particle paths in the parametric and physical space. P and Q are the particle starting and next positions in the physical space, respectively. P' and Q' are their images in the parametric space. P'' is the intersection point with the elemental boundary. Triangles $\widehat{A'B'C'}$ and $\widehat{B'A'D'}$ are the images in the parametric space of triangles $\widehat{ABC}$ and $\widehat{BAD}$.

In the case of linear elements, this procedure is quite efficient since the inverse mapping $\boldsymbol{\zeta} = \boldsymbol{\Psi}(\mathbf{x})$, used in step 1 and every substep of step 3, is analytic. However, for curvilinear high-order elements the computational expense is significantly higher because of the cost of the nonlinear iteration required to calculate the inverse mapping. This is described in Section 3.1.1.

3.1.1. Inverse mapping. The finite element representation of the velocity field is piecewise continuous. Therefore, the evaluation of the velocity at a point in physical space involves two steps: finding the element that contains the point, and interpolating the velocity using the elemental expansion coefficients and polynomial basis in Eq. (10).

The first step is a range searching problem which is solved by a trial and error procedure because the element to which the point $\mathbf{x}$ belongs is not explicitly known. Each element in the mesh is assigned a bounding box that encloses the element. An initial guess of the element containing the point is obtained by searching through the element bounding boxes to find one that contains the desired point. This is followed by the calculation of the local coordinates $\boldsymbol{\zeta}$ in the parametric space of the element associated to the current bounding box. If the calculated local coordinates are outside the limits of definition of the parent element, the rest of the bounding boxes are investigated, until the element containing the point is found.

The efficiency of the searching procedure can be improved by using appropriate data structures. If the number of elements in the mesh is large, tree structures [2] could be used to find the element containing the starting point of a trajectory. For other points along the trajectory, it is generally more efficient to start in the element containing the position at the previous time step and then perform the search in the neighborhood of that element using an element-to-element connectivity data structure.

The computation of the local coordinates for *linear* elements is relatively inexpensive since an analytical expression for the inverse mapping (4) is available. Unfortunately, an analytical expression of the inverse mapping (4) is not available, in the general case, for *high-order* elements and, because of the nonlinearity of the problem to be solved, an iterative procedure such as a Newton–Raphson iteration [8] has to be applied.

For instance, if we would like to determine the coordinate $\boldsymbol{\zeta}$ such that $\chi^e(\boldsymbol{\zeta}) = \mathbf{x}$, we could formulate this as the problem of finding a zero of a function $\mathbf{F}(\boldsymbol{\zeta})$, where

$$\mathbf{F}(\boldsymbol{\zeta}) = \chi^e(\boldsymbol{\zeta}) - \mathbf{x}. \tag{25}$$

The Newton–Raphson iteration applied to Eq. (25) can be written as

$$\mathbf{J}_e \cdot [\boldsymbol{\zeta}_{i+1} - \boldsymbol{\zeta}_i] = -\mathbf{F}(\boldsymbol{\zeta}_i), \tag{26}$$

where i represents an iteration counter and $\mathbf{J}_e$ denotes the Jacobian of the mapping

$$\mathbf{J}_e = \frac{\partial \mathbf{F}}{\partial \boldsymbol{\zeta}} = \frac{\partial \chi^e}{\partial \boldsymbol{\zeta}}. \tag{27}$$

A good strategy to identify the nearest elements is important because of the computational expense of this iterative procedure that requires interpolation of the matrix $\frac{\partial \mathbf{F}}{\partial \boldsymbol{\zeta}}$, which has nine entries in 3D. Since the interpolation involves polynomial evaluation, we also note that the computation time will increase with the polynomial order of the mapping.

3.2. *Particle Tracking in the Parametric Space*

This section describes an alternative method for advancing a particle along a trajectory that avoids the expensive calculation of the local coordinates through the iterative solution of Eq. (25). Rather than considering the rate of change of the position of a particle in physical space as described by Eq. (1), we can map the velocity field onto the parent element to obtain an equation representing the corresponding rate of change of the particle position in the parametric space within element e as

$$\frac{d\boldsymbol{\zeta}^e}{dt} = \mathbf{u}^e_\zeta(\boldsymbol{\zeta}, t). \tag{28}$$

The transformed velocity $\mathbf{u}^e_\zeta$ in the parent element can be evaluated in terms of the physical velocity $\mathbf{u}$ by applying the chain rule to each of its scalar components, i.e.,

$$u_i = \frac{dx_i}{dt} = \frac{\partial x_i}{\partial \zeta_1}\frac{d\zeta_1}{dt} + \frac{\partial x_i}{\partial \zeta_2}\frac{d\zeta_2}{dt} + \frac{\partial x_i}{\partial \zeta_3}\frac{d\zeta_3}{dt} \quad i = 1, 2, 3, \tag{29}$$

which can be written in matrix form as

$$\mathbf{u} = \frac{\partial \chi^e}{\partial \boldsymbol{\zeta}}\mathbf{u}^e_\zeta = \mathbf{J}_e \mathbf{u}^e_\zeta, \tag{30}$$

where $\mathbf{J}_e$ is the Jacobian of the mapping between element e and its parent element. The final expression for the transformed velocity in the parent element is

$$\mathbf{u}^e_\zeta = \frac{d\boldsymbol{\zeta}^e}{dt} = \mathbf{J}_e^{-1}\mathbf{u}. \tag{31}$$

This equation can only be applied within the corresponding elemental region since the representation of the velocity field, and consequently the Jacobian $\mathbf{J}^e$, is piecewise continuous.

However, if we can determine the intersection of the particle trajectory with the boundary of the parent element, then the continuity of the local coordinates along a boundary can be used to advance the particle to the next adjacent element. This strategy leads to the parametric space particle tracking which is described in the following.

Omitting the index e for simplicity, the use of an s-stage Runge–Kutta for the integration in time of Eq. (28) leads to

$$\zeta^{n+1} = \zeta^n + \Delta t \sum_{i=1}^{s} b_i \mathbf{u}_{\zeta i}, \tag{32}$$

where

$$\mathbf{u}_{\zeta_i} = \mathbf{u}_\zeta \left(\zeta^n + \Delta t \sum_{j=1}^{i-1} a_{ij} \mathbf{u}_{\zeta_j}, t^n + c_i \Delta t \right). \tag{33}$$

The starting point of the algorithm for this strategy requires the calculation of the local coordinates ζ_0 of the initial position of the particle $\mathbf{x}_0$ using the inverse iteration described in Section 3.1.1. At a later stage n of the iterative procedure, and using a fixed time step Δt, the algorithm can be summarized as follows:

1. interpolate the velocity field $\mathbf{u}$ and the Jacobian $\mathbf{J}$ at the point ζ_n, within element e^n, to evaluate $\mathbf{u}_\zeta(\zeta_n)$ using Eq. (31).
2. Apply the time integration scheme, given by Eqs. (32) and (33), with time step Δt to determine a new position ζ_{n+1}.
3. Consider the point of local coordinates ζ_{n+1}:
   (i) if the point is interior to element e^n, take $t^{n+1} = t^n + \Delta t$, let $n = n + 1$, and go to step 1, otherwise
   (ii) if the point is outside the element e^n, find the intersection of the trajectory with the boundary ζ_{n+1} (point P'' in Fig. 5) using the algorithm described in Section 3.2.1. The corresponding time increment required to reach the boundary is now $\Delta\tau < \Delta t$ and the following steps are:
      a. find the adjacent element e^{n+1} that shares the boundary point with e^n using the element-to-element connectivity information,
      b. determine the matching local coordinates ζ_{n+1} in the adjacent element e^{n+1},
      c. take $t^{n+1} = t^n + \Delta\tau$, let $n = n + 1$ and go to step 1.

From an initial inspection of this algorithm, it is apparent that advancing the particle in the parametric space has eliminated the nonlinear inverse mapping evaluation discussed in Section 3.1.1. We are however now faced with the problem of determining the intersection of the trajectory with the boundary of the parent element. If we use an Euler scheme ($s = 1$), the intersection with the boundary is easily determined since it is linearly dependent on Δt. However, for a multistage scheme ($s > 1$) if a particle crosses the boundary we note, by inspection of Eqs. (32) and (33), that the point ζ^{n+1} is also a nonlinear function of the timestep. This means that the calculation of the time step $\Delta\tau$ required to move the particle exactly to the boundary is a nonlinear problem that has to be solved iteratively. The need to determine the boundary intersection arises from the discontinuous behavior of the elemental Jacobian which results in changes of direction of the local velocity $\mathbf{u}_\zeta$ across elements.

In the following section, we will discuss two techniques to solve the nonlinear problem of finding the time step $\Delta\tau$ and the intersection between the trajectory and the boundary of the parent element. For a given problem, the effectiveness of this approach decreases as the size of the mesh, and consequently the number of intersections with elemental boundaries, increases. This approach is therefore more suitable to be used in conjunction with p-refinement.

3.2.1. Boundary intersection. The integration in the parametric space must take into account particles leaving an element through an elemental boundary as shown in Fig. 5 for a two-dimensional case. In two dimensions, the boundaries are edges and vertices. We will not consider vertices as separate boundaries, since they correspond to the intersection of two edges. In three dimensions the boundaries are faces, edges, and vertices. Similarly, we will consider only faces, since edges and vertices are the intersection of two and three faces, respectively.

The boundary through which a particle leaves is easily identified since the coordinates of the intersection will lie within a predefined distance from a line or plane in the elemental boundary. This is discussed in more detail in Section 4.3. The problem of determining the time step $\Delta\tau$ required for a particle to hit a boundary is more difficult because the Runge–Kutta coefficients are nonlinear functions of the time step as shown in Eqs. (14) and (15). In a high-order method, the interpolation of the velocity field within an element is an expensive operation, and it is therefore important to use a method with a good rate of convergence to reduce the number of velocity evaluations. We have implemented the quadratically convergent Newton–Raphson and Steffensen's methods [8] to solve this problem.

Let us consider the simple case of triangular elements as an example. If the particle crosses the lower boundary of the parent element, i.e., the edge $B'A'$ as shown in Fig. 5, then we know that $\zeta_2 = -1$. Substituting this value in the second component of Eq. (32), the time step $\Delta\tau$ required to exactly reach the boundary is obtained as the solution of the equation

$$G_1(\boldsymbol{\zeta}^n, \Delta\tau) = 1 + \zeta_2^n + \Delta\tau \sum_{i=1}^{s} b_i u_{\zeta_2 i} = 0. \tag{34}$$

Notice that since $u_{\zeta_2 i} = u_{\zeta_2 i}(\Delta\tau)$ through Eq. (33), the function G_1 is nonlinear in $\Delta\tau$. If a particle crosses the diagonal edge of the parent element, i.e., the edge $A'D'$ in Fig. 5, then $\zeta_1 + \zeta_2 = 0$ and Eq. (32) will now read

$$G_2(\boldsymbol{\zeta}^n, \Delta\tau) = \sum_{k=1}^{2} \left(\zeta_k^n + \Delta\tau \sum_{i=1}^{s} b_i u_{\zeta_k i} \right) = 0. \tag{35}$$

Finally, the functional equation for a particle crossing the vertical boundary of the parent element, $\zeta_1 = -1$, is

$$G_3(\boldsymbol{\zeta}^n, \Delta\tau) = 1 + \zeta_1^n + \Delta\tau \sum_{i=1}^{s} b_i u_{\zeta_1 i} = 0. \tag{36}$$

Therefore, the calculation of the time step $\Delta\tau$ required for a particle to hit a boundary requires the solution of the equation $G_i(\boldsymbol{\zeta}^n, \Delta\tau) = 0$ where the index i depends on the boundary which the particle will intersect.

Newton–Raphson method. The application of this method to the solution of Eqs. (34), (35), or (36) results in the iterative procedure

$$\Delta\tau^{j+1} = \Delta\tau^{j} - \frac{G_i(\zeta^n, \Delta\tau^j)}{G_i'(\zeta^n, \Delta\tau^j)}, \tag{37}$$

where G_i' denotes the derivative of the function G_i with respect to $\Delta\tau$, and j represents the iteration index.

For triangular regions the derivatives of Eqs. (34), (35), and (36), are

$$G_1'(\zeta^n, \Delta\tau^j) = \sum_{i=1}^{s} b_i u_{\zeta_2 i} + \Delta\tau \sum_{i=1}^{s} b_i \frac{du_{\zeta_2 i}}{d\Delta\tau} \tag{38}$$

$$G_2'(\zeta^n, \Delta\tau^j) = \sum_{k=1}^{2} \left(\sum_{i=1}^{s} b_i u_{\zeta_k i} + \Delta\tau \sum_{i=1}^{s} b_i \frac{du_{\zeta_k i}}{d\Delta\tau} \right) \tag{39}$$

$$G_3'(\zeta^n, \Delta\tau^j) = \sum_{i=1}^{s} b_i u_{\zeta_1 i} + \Delta\tau \sum_{i=1}^{s} b_i \frac{du_{\zeta_1 i}}{d\Delta\tau}. \tag{40}$$

Following the notation of Eq. (33), we have

$$u_{\zeta_k i}(\Delta\tau) = u_{\zeta_k}\left(\zeta_{ki}^n, t_i^n\right), \tag{41}$$

with

$$\zeta_{ki}^n(\Delta\tau) = \zeta_k^n + \Delta\tau \sum_{j=1}^{i-1} a_{ij} u_{\zeta_k j}; \quad t_i^n(\Delta\tau) = t^n + c_i \Delta\tau \tag{42}$$

and the derivative of $u_{\zeta_k i}$ with respect to $\Delta\tau$ is calculated as

$$\begin{aligned} \frac{du_{\zeta_k i}}{d\Delta\tau} &= \frac{\partial u_{\zeta_k i}}{\partial t_i^n} \frac{\partial t_i^n}{\partial \Delta\tau} + \frac{\partial u_{\zeta_k i}}{\partial \zeta_{ki}^n} \frac{\partial \zeta_{ki}^n}{\partial \Delta\tau} \\ &= \frac{\partial u_{\zeta_k i}}{\partial t_i^n} c_i + \frac{\partial u_{\zeta_k i}}{\partial \zeta_{ki}^n} \left(\sum_{j=1}^{i-1} a_{ij} u_{\zeta_k j} + \Delta\tau \sum_{j=1}^{i-1} a_{ij} \frac{\partial u_{\zeta_k j}}{\partial \Delta\tau} \right). \end{aligned} \tag{43}$$

Each Newton–Raphson step requires the evaluation of both the velocity and its gradient three times. This involves the calculation of the gradient and then its interpolation.

Steffensen's method. This scheme requires only function evaluations, is quadratically convergent as the Newton–Raphson method and, applied to the system of Eqs. (34)–(36), leads to the iterative sequence

$$\Delta\tau^{j+1} = \Delta\tau^{j} - \frac{[G_i(\zeta^n, \Delta\tau^j)]^2}{G_i(\zeta^n, \Delta\tau^j + G_i(\zeta^n, \Delta\tau^j)) - G_i(\zeta^n, \Delta\tau^j)}. \tag{44}$$

This method is computationally less expensive than the Newton–Raphson iteration since it requires only function evaluations.

Treatment of singularities. Both schemes will quadratically converge to the nearest root if the initial guess is sufficiently close and if the behavior of the function near the root is smooth. This might however present problems if the elemental region is very distorted and sampling of the velocity is required outside the element. The presence of large distortions could result in the Jacobian matrix of the mapping, and therefore the local velocities, being ill-behaved outside the parent element. Further the use of the auxiliary mapping with its collapsed coordinate system in the spectral approach introduces a singularity in the interpolation coordinates $\boldsymbol{\eta}$ (see Fig. 3). Although the Cartesian parametric coordinates are analytic within the element, the numerical evaluation of the local velocity becomes very ill-defined in the vicinity of the singular point of the mapping $\boldsymbol{\zeta} \rightarrow \boldsymbol{\eta}$. Alternative polynomial representations could be used that do not introduce such geometrical singularity, but it is not clear that they could guarantee a well-behaved approximation for curvilinear elements outside their domain of definition. The nonlinearity of the iterative process and the potential misbehavior of the root-finding procedure make this approach unsatisfactory and lead us to consider an alternative treatment: a hybrid approach based on a guided search.

4. GUIDED SEARCH APPROACH TO PARTICLE TRACKING

The previous sections have highlighted several problems in the implementation of the particle tracking algorithm in the physical and parametric spaces. Both strategies show weak points when dealing with high-order elements. The main weakness of the physical space approach is the need to solve the nonlinear inverse mapping problem at each step and substep of a multistage scheme. This deficiency was overcome by using a time integration scheme in the parametric space but at the expense of requiring the solution of the nonlinear problem of finding the intersection of the trajectory with the elemental boundary.

To overcome these problems and improve efficiency we propose a hybrid approach where the velocity is predominantly evaluated in physical space but the substeps utilize the parametric space. Such an approach eliminates the inverse mapping iteration at each substep although, as we shall demonstrate, it does introduce an error associated with the variation of the Jacobian of the mapping.

As a starting point we note that the Runge–Kutta scheme for the physical space particle tracking given by Eqs. (23) and (24) can be equivalently written as

$$\mathbf{x}^{n+1} = \mathbf{x}^n + \Delta t \bar{\mathbf{U}} \tag{45}$$

$$\bar{\mathbf{U}} = \sum_{i=1}^{s} b_i \mathbf{u}_i \tag{46}$$

$$\mathbf{u}_i = \mathbf{u}(\mathbf{x}_i, t^n + c_i \Delta t) \tag{47}$$

$$\mathbf{x}_i = \mathbf{x}^n + \Delta t \bar{\mathbf{U}}_i \tag{48}$$

$$\bar{\mathbf{U}}_i = \sum_{j=1}^{s} a_{ij} \mathbf{u}_j. \tag{49}$$

Let us return to the point discussed in Section 2.2, i.e., that each step of the Runge–Kutta scheme can be considered as a linear step based on an average velocity (see Fig. 4), in the context of the physical space particle tracking. As discussed in Section 3.1, the computational difficulty of this scheme is due to the nonlinear iteration needed to evaluate the local parametric coordinates $\boldsymbol{\zeta}^{n+1}$ and $\boldsymbol{\zeta}_i$, the images in the parametric space of the physical coordinates $\mathbf{x}^{n+1}$ and $\mathbf{x}_i$, respectively, which are necessary to interpolate the velocities. In the hybrid algorithm, we replace Eq. (45) by the substeps:

$$\langle\zeta\rangle_{e(1)} = \zeta^n \tag{50a}$$

$$\langle\zeta\rangle_{e(k+1)} = \mathcal{T}_k\big\{\langle\zeta\rangle_{e(k)} + \langle\Delta\tau\mathbf{J}^{-1}\rangle_{e(k)}\bar{\mathbf{U}}\big\} \quad k=1,\ldots,N_e-1 \tag{50b}$$

$$\zeta^{n+1} = \langle\zeta\rangle_{e(N_e)} + \langle\Delta\tau\mathbf{J}^{-1}\rangle_{e(N_e)}\bar{\mathbf{U}} \tag{50c}$$

$$\mathbf{x}^{n+1} = \chi(\zeta^{n+1}). \tag{50d}$$

Here $e(k);\ k=1,\ldots,N_e$ denotes the list of elements crossed by a particle during the guided search; the symbol $\langle f\rangle_{e(k)}$ indicates that the quantity f is evaluated within element $e(k)$; N_e is the number of such elements; and

$$\mathcal{T}_k : \langle\zeta\rangle_{e(k)} \rightarrow \langle\zeta\rangle_{e(k+1)} \tag{51}$$

represents a mapping of local parametric coordinates across elemental boundaries. The continuity of position across boundary faces can be expressed as

$$\chi^{e(k+1)}(\mathcal{T}_k\{\langle\zeta\rangle_{e(k)}\}) = \chi^{e(k)}(\langle\zeta\rangle_{e(k)}). \tag{52}$$

The elemental time steps $\langle\Delta\tau\rangle_{e(k)}$ are such that

$$\sum_{k=1}^{N_e}\langle\Delta\tau\rangle_{e(k)} = \Delta t. \tag{53}$$

In a similar fashion, Eq. (48) is replaced by the substeps:

$$\langle\zeta\rangle_{e(1)} = \zeta^n \tag{54a}$$

$$\langle\zeta\rangle_{e(k+1)} = \mathcal{T}_k\big\{\langle\zeta\rangle_{e(k)} + \langle\Delta\tau\mathbf{J}^{-1}\rangle_{e(k)}\bar{\mathbf{U}}_i\big\} \quad k=1,\ldots,N_e-1 \tag{54b}$$

$$\zeta_i = \langle\zeta\rangle_{e(N_e)} + \langle\Delta\tau\mathbf{J}^{-1}\rangle_{e(N_e)}\bar{\mathbf{U}}_i \tag{54c}$$

$$\mathbf{x}_i = \chi(\zeta_i). \tag{54d}$$

In this manner we advance the parametric coordinates rather than the physical coordinates thus circumventing the need for the inverse mapping. Interpreting the trajectory given by Eqs. (45) and (48) as a straight line in the direction of an average velocity simplifies the problem of calculating the parametric coordinate of the intersection with the parent element boundary since it is now linear and can therefore be evaluated easily. We shall refer to this process as a *guided search*. A complete description of this step is given in Section 4.1. An outline of this algorithm follows.

The starting point of this procedure is, in common with all the procedures, the calculation of the local coordinates ζ_0 of the starting position $\mathbf{x}_0$. At a later stage n of the iteration, this strategy can be summarized as follows:

1. Interpolate the velocity field and Jacobian matrix at point ζ_n within element $e(n)$ using Eq. (10).
2. Apply the time integration scheme based on the physical space integration as follows.

   (i) For each substep of the Runge–Kutta scheme:

a. evaluate the average velocity in physical space based upon Eq. (49),

b. advance the parametric and physical coordinates $\chi(\zeta_i)$ using the guided search Eqs. (54a)–(54d), and

c. evaluate the intermediate physical velocities and Jacobians using Eq. (10).

(ii) Compute the final average physical velocity using (46).

(iii) Compute the position of the particle in the physical and parametric space $\chi(\zeta_n)$ using the guided search Eqs. (50a)–(50d).

It is also possible to advance Eqs. (45) and (48) concurrently with the guided search Eqs. (50a)–(50d) and (54a)–(54d) to act as an error check. This point will be discussed further in Section 4.2.

4.1. *Guided Search Algorithm*

The *guided search* allows a particle leaving an element to be traced without resorting to an iterative procedure. The idea behind this approach is based upon the observation that each stage of the Runge–Kutta scheme can be considered as a linear substep. In this approach we take a series of linear substeps in the parametric space instead of a linear substep in physical space. This point is illustrated in Fig. 6 where we consider a step starting at point P in the physical space. A linear step in physical space $\Delta\mathbf{x} = \mathbf{v}\Delta t$ would take the particle to point Q. We then require the local parametric coordinate of point Q in order to proceed. In the guided search, the parametric point P' is advanced by a linear substep $\Delta\boldsymbol{\zeta} = \mathbf{v}_\zeta\,\Delta\tau_{e_1}$ based on the local parametric velocity $\mathbf{v}_\zeta = \mathbf{J}_{e_1}^{-1}\mathbf{v}$. In general, the point will not remain within an element. The time taken for the point to meet a boundary of the parent element (point R' in Fig. 6) is then $\Delta\tau_{e_1} \leq \Delta t$. Since the step is linear and the boundary is planar, the intersection can be evaluated analytically. To complete the guided search, the Jacobian matrix is then evaluated at point R' in element e_2 and a new parametric velocity $\mathbf{v}_\zeta = \mathbf{J}_{e_2}^{-1}\mathbf{v}$

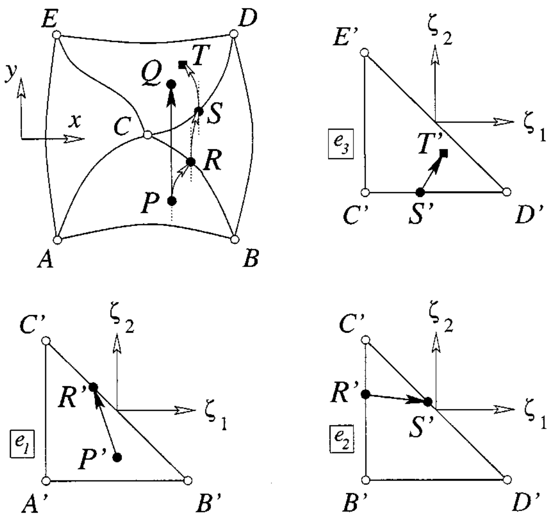


FIG. 6. Illustration of the guided search algorithm. The vector PQ represents the step in the physical space parallel to the global averaged velocity $\mathbf{v}_P$ evaluated at P. The straight segments $P'R'$, $R'S'$, and $S'T'$ are steps in the parametric space parallel to the transformed velocity $\mathbf{v}_\zeta = \mathbf{J}^e\mathbf{v}_P$. The local velocity $\mathbf{v}_\zeta$ is evaluated at the points P', R', and S' for elements e_1, e_2, and e_3, respectively. The path $PRST$ is the image in physical space of the piecewise linear steps taken in the parametric space.

is determined. The point is then linearly advanced through element e_2 over a time $\Delta\tau_{e_2}$, which is evaluated as the time for the particle to reach S'. Since the total timestep has not been completed yet, the Jacobian matrix is evaluated at point S' in element e_3 and the new parametric velocity $\mathbf{v}_\zeta = \mathbf{J}_{e_3}^{-1}\mathbf{v}$ is determined. The particle is then linearly advanced a time $\Delta\tau_{e_3}$, such that $\Delta t = \Delta\tau_{e_1} + \Delta\tau_{e_2} + \Delta\tau_{e_3}$.

This procedure drastically reduces the computation time required to trace a particle and overcomes the problems posed by the iterative solution of the nonlinear problems associated with the two previous particle tracking strategies. The guided search is exact when applied to elements with a constant Jacobian but an error arises when the trajectory crosses elements with varying Jacobian. This will be explained in the next section. Typically, high-order schemes use linear element mappings when dealing with straight-sided elements and so varying Jacobian are usually associated with curvilinear elements.

4.2. *Accuracy of the Guided Search Scheme*

To assess the errors introduced by the guided search, the substeps of the hybrid Runge–Kutta scheme should be interpreted in the physical space. For the hybrid scheme to be exact, we require Eq. (45) to be equivalent to Eqs. (50a)–(50d). In this section we will show that this is satisfied when the mapping between the parametric and physical spaces is linear.

The proof that Eqs. (50a)–(50d) are equivalent to expression (45) for a linear mapping proceeds as follows. For a linear mapping, the Jacobian within an element e is constant and the coordinate transformation can be written in incremental form as

$$\chi^e(\Delta\boldsymbol{\zeta}) = \mathbf{J}_e\Delta\boldsymbol{\zeta}. \tag{55}$$

Substituting Eq. (50c) into (50d) we have

$$\mathbf{x}^{n+1} = \chi(\boldsymbol{\zeta}^{n+1}) = \chi^{e(N_e)}\big(\langle\boldsymbol{\zeta}\rangle_{e(N_e)} + \langle\Delta\tau\mathbf{J}^{-1}\rangle_{e(N_e)}\bar{\mathbf{U}}\big).$$

Using the linearity of the map and expression (55), this equation can be written as

$$\mathbf{x}^{n+1} = \chi^{e(N_e)}(\langle\boldsymbol{\zeta}\rangle_{e(N_e)}) + \langle\Delta\tau\rangle_{e(N_e)}\bar{\mathbf{U}}.$$

Formula (50b) can be used to move across adjacent elements $e(N_e)$ and $e(N_e-1)$ to get

$$\mathbf{x}^{n+1} = \chi^{e(N_e)}\big(\mathcal{T}_{N_e-1}\big\{\langle\boldsymbol{\zeta}\rangle_{e(N_{e-1})} + \langle\Delta\tau\mathbf{J}^{-1}\rangle_{e(N_{e-1})}\bar{\mathbf{U}}\big\}\big) + \langle\Delta\tau\rangle_{e(N_e)}\bar{\mathbf{U}},$$

which, applying continuity across element boundaries through Eq. (52), gives

$$\mathbf{x}^{n+1} = \chi^{e(N_e-1)}\big(\langle\boldsymbol{\zeta}\rangle_{e(N_e-1)} + \langle\Delta\tau\mathbf{J}^{-1}\rangle_{e(N_e-1)}\bar{\mathbf{U}}\big) + \langle\Delta\tau\rangle_{e(N_e)}\bar{\mathbf{U}}.$$

Using the linearity of the mapping and applying the previous operation to all the elements in the particle path in succession leads to

$$\mathbf{x}^{n+1} = \chi^{e(1)}\big(\langle\boldsymbol{\zeta}\rangle_{e(1)}\big) + \left(\sum_{k=1}^{N_e}\langle\Delta\tau\rangle_{e(k)}\right)\bar{\mathbf{U}},$$

where the application of formulas (50a) and (53) finally results in

$$\mathbf{x}^{n+1} = \mathbf{x}^n + \left(\sum_{k=1}^{N_e} \langle \Delta\tau \rangle_{e(k)}\right) \bar{\mathbf{U}} = \mathbf{x}^n + \Delta t \bar{\mathbf{U}},$$

which coincides with expression (45) and therefore concludes the proof.

When the mapping χ is not linear the equivalence clearly does not exist. The error associated with this strategy when applied to high-order elements will be assessed in Section 6 using numerical examples. It is also possible to just use the guided search as an initial guess to the full physical space substeps given by Eqs. (45) and (48). Introducing a user-defined tolerance it is possible to compare the difference between the output of the guided search from Eqs. (50a)–(50b) and (54a)–(54b) with Eqs. (45) and (48) and so limit the error associated with the varying Jacobian.

4.3. *Guided Search Implementation*

We recall that the essential steps of the guided search are the following. At the initial stage of the iteration, $k = 0$, given a parametric coordinate $\boldsymbol{\zeta}_n$ and a velocity $\boldsymbol{V}$ in element $e(0)$, and a time step Δt, we set $\langle \boldsymbol{\zeta} \rangle^n_{e(0)} = \boldsymbol{\zeta}^n$, $\langle \boldsymbol{V}_\zeta \rangle_{e(0)} = \langle \boldsymbol{J}^{-1} \rangle_{e(0)} \boldsymbol{V}$ and $\Delta\tau = \Delta t$.

1. Evaluate $\langle \boldsymbol{\zeta} \rangle^{n+1}_{e(k)} = \langle \boldsymbol{\zeta} \rangle^n_{e(k)} + \Delta\tau \langle \boldsymbol{V}_\zeta \rangle_{e(k)}$.
2. If $\langle \boldsymbol{\zeta} \rangle^{n+1}_{e(k)}$ does not lie within element $e(k)$, then:

(i) determine the parametric coordinates, $\langle \boldsymbol{\zeta}_{in} \rangle^n_{e(k)}$, of the intersection point with the face and the time step $\Delta\tau_{in}$ to reach the face,

(ii) find the adjacent element $e(k+1)$ and calculate the parametric coordinates on the adjacent element as $\langle \boldsymbol{\zeta}_{in} \rangle^n_{e(k+1)} = \mathcal{T}_k(\langle \boldsymbol{\zeta}_{in} \rangle^n_{e(k)})$,

(iii) set $\langle \boldsymbol{\zeta} \rangle^n_{e(k+1)} = \langle \boldsymbol{\zeta}_{in} \rangle^n_{e(k+1)}$, $\langle \boldsymbol{V}_\zeta \rangle_{e(k+1)} = \langle \boldsymbol{J}^{-1} \rangle_{e(k+1)} \boldsymbol{V}$, $\Delta\tau = \Delta\tau - \Delta\tau_{in}$, $k = k + 1$ and return to step 1.

Or else return $\boldsymbol{\zeta}^{n+1} = \langle \boldsymbol{\zeta} \rangle^{n+1}_{e(k)}$.

Two operations in stage 2 of the previous procedure are significant:

1. determining whether a point crosses the planar face of the standard region and identifying the timestep to intersection and the parametric coordinates at the face, and
2. evaluating the local parametric coordinates and element number of the adjacent element.

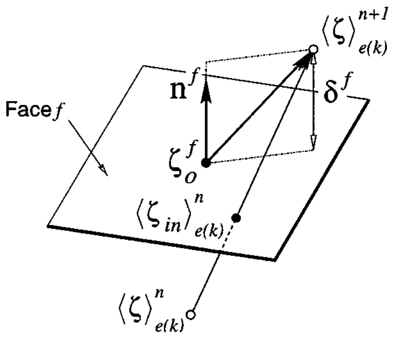
A further operation required is to interpolate the Jacobian matrix $\langle \boldsymbol{J}^{-1} \rangle_{e(k+1)}$ to update the velocity $\langle \boldsymbol{V}_\zeta \rangle_{e(k+1)}$. If the parametric coordinates $\langle \boldsymbol{\zeta}_{in} \rangle^n_{e(k+1)}$ are known, this simply requires the application of Eq. (10).

4.4. *Intersection Criterion*

Using the superscript 'f' to refer to values on a face of the standard element, the distance δ^f of a point $\langle \boldsymbol{\zeta} \rangle^{n+1}_{e(k)}$ to the planar face is evaluated as

$$\delta^f = \left(\langle \boldsymbol{\zeta} \rangle^{n+1}_{e(k)} - \boldsymbol{\zeta}^f_0\right) \cdot \boldsymbol{n}^f, \tag{56}$$

where $\boldsymbol{n}^f$ denotes the unit outwards normal to the face and $\boldsymbol{\zeta}^f_0$ are the coordinates of the centroid of the face. This is illustrated in Fig. 7.



$\langle\zeta\rangle^{n+1}_{e(k)}$
n^f
δ^f
Face f
ζ^f_o
$\langle\zeta_{in}\rangle^n_{e(k)}$
$\langle\zeta\rangle^n_{e(k)}$

FIG. 7. Distance to a face of the standard element.

According to the expression (56), the point will be inside the element if $\delta^f \leq 0$ for all the faces f of the element. Given the time step $\Delta\tau$ and the velocity $\langle \boldsymbol{V}_\zeta\rangle_{e(k)}$, we can evaluate the time step required to reach the face $\Delta\tau^f_{in}$ as

$$\Delta\tau^f_{in} = \Delta\tau - \Delta\tau^f_{over} \quad \text{where} \quad \Delta\tau^f_{over} = \frac{\delta_f}{\langle \boldsymbol{V}_\zeta\rangle_{e(k)} \cdot \boldsymbol{n}^f}. \tag{57}$$

Clearly it is only possible to evaluate $\Delta\tau^f_{over}$ if $\langle \boldsymbol{V}_\zeta\rangle_{e(k)} \cdot \boldsymbol{n}^f$ is different from zero. The velocity could be parallel to a face and, because of numerical round-off error, have a negative normal velocity within an element and a positive normal velocity in the adjacent element. This might result in an infinite loop of substeps where the particle leaves and enters adjacent elements through the same point on the common boundary. This situation can be avoided by introducing a tolerance ϵ representing the distance that a particle can move normal to the face before being interpreted as having left the element. Under such criterion, the point is only considered to have left the element if $|\langle \boldsymbol{V}_\zeta\rangle_{e(k)} \cdot \boldsymbol{n}^f| > \epsilon$, and if this is not true then $\Delta\tau^f_{over} = 0$.

Given the overshoot time of the particle on the linear trajectory after intersecting with the face, $\Delta\tau^f_{over}$, the coordinates of the intersection point are

$$\langle\zeta_{in}\rangle_{e(k)} = \langle\zeta\rangle^{n+1}_{e(k)} - \Delta\tau^f_{over}\langle \boldsymbol{V}_\zeta\rangle_{e(k)}. \tag{58}$$

The last computational issue it to determine which face a given trajectory intersects. A brute force approach would try all faces and find the face with the minimum $\Delta\tau_{in}$ or, equivalently, the maximum $\Delta\tau_{over}$. However, since the orientation of the standard elemental regions is known a priori, an inspection of the linear parametric velocity eliminates certain faces from the intersection problem. For example, consider the standard quadrilateral region shown in Fig. 8(a), if both components of the velocity $\langle \boldsymbol{V}_\zeta\rangle_{e(k)}$ are positive then the particle must intersect either face 1 or face 2. Clearly identifying the relevant faces based upon the sign of each component of the velocity is straightforward and reduces the number of possible boundaries a point can intersect to two sides in two dimensions and three faces in three dimensions. We note that the problem can be further simplified in the case of a simplex such as the standard triangular region shown in Fig. 8(b) where if both components of $\langle \boldsymbol{V}_\zeta\rangle_{e(k)}$ are positive, then the particle must cross face 1.

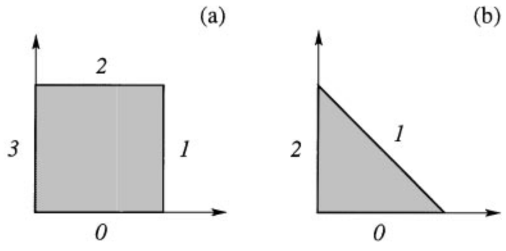


FIG. 8. Standard region definition for (a) a quadrilateral element and (b) a triangular element.

4.5. *Evaluation of Adjacent Coordinates*

Having evaluated the parametric coordinates within the first element $\langle\zeta_{in}\rangle^n_{e(k)}$ we then require the matching parametric coordinates in the adjacent element $\langle\zeta_{in}\rangle^n_{e(k+1)}$. The definition of a *collapsed coordinate* system [11] is advantageous since it provides a consistent local coordinates system (η_1, η_2) within each face of the element (a single coordinate is only required in two dimensions). To permit the hybrid mix of elemental regions, such as tetrahedral and prismatic elemental domains, the evaluation of the adjacent coordinates can be considered in three steps:

1. evaluate face coordinates $\langle\boldsymbol{\eta}\rangle^n_{e(k)}$ from $\langle\zeta_{in}\rangle^n_{e(k)}$,
2. apply rotation/reflections of face connection to obtain new face coordinate $\langle\boldsymbol{\eta}\rangle^n_{e(k+1)}$ from $\langle\boldsymbol{\eta}\rangle^n_{e(k)}$, and
3. determine the new elemental coordinates $(\zeta_{in}\rangle^n_{e(k+1)}$ from $\langle\boldsymbol{\eta}\rangle^n_{e(k+1)}$.

Steps 1 and 3 simply uses the definition of the appropriate components of the collapsed coordinate system [11]. Step 2 however needs to take account of the different rotations in which two elemental regions can connect. An example of this rotation is shown in Fig. 9 where we illustrate the three steps in determining the adjacent coordinate between two tetrahedral elements. In this example, the face must be rotated by 180° to align the

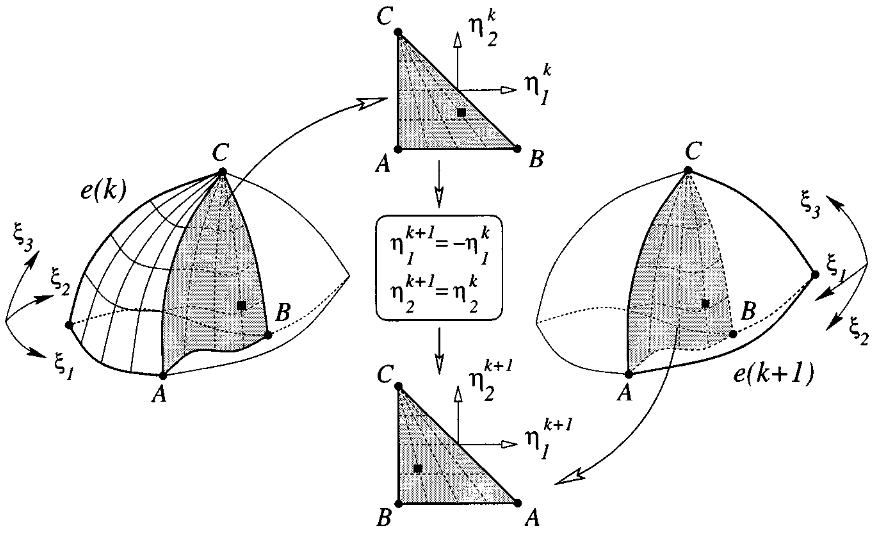


FIG. 9. Tetrahedral connectivity.

face coordinates. This is equivalent to changing the sign of both local face coordinates. Information about elemental orientation is typically stored as part of the finite element mesh and consequently access to this information will depend upon the local implementation. In two dimensions, this requires storing the orientation of the edge coordinate that might need to be transformed.

As a final point, we note that for tetrahedral elemental regions the collapsed coordinate system for a face, as shown in Fig. 9, will contain a singular point at one of the vertices. The consistency of the p-type expansion requires that the singular points of the collapsed coordinate systems at adjacent faces must coincide. It can be shown that this conformity requirement can be achieved for any mesh [11, 16, 17]. It also has the added advantage that only the nondegenerate face coordinate η_1 may need transforming.

4.6. *Viable Schemes for Particle Tracking*

As discussed in Section 3, we have a variety of possible strategies to handle particle tracking within high-order spatial representations. In Section 6 we will compare the following four approaches:

1. Particle tracking in the physical space evaluating the inverse mapping using a Newton–Raphson iteration as discussed in Section 3.1.1. We will denote this scheme as the *physical space* algorithm.
2. Particle tracking in the physical space using the guided search algorithm discussed in Sections 4.1 and 4.3. We will denote this scheme as the *guided search* algorithm.
3. Particle tracking in the physical space using the guided search algorithm (see Sections 4.1 and 4.3) and checking the error between the physical space advancement and the guided search. This allows the error introduced by curved elements to be monitored and requires an error tolerance ϵ above which the iterative technique to evaluate the inverse mapping is applied. We will refer to this scheme as the *guided search* (ϵ) algorithm.
4. Finally, we will use a hybrid scheme where the particles are advanced in the parametric space, as discussed in Section 3.2, provided they remain within the element during all substeps of the Runge–Kutta algorithm. If during a substep the particle leaves the elemental region, then physical space scheme using the error-checked guided search 3 is applied. We will refer to this scheme as the *hybrid* algorithm.

We note that scheme 3 can be considered as an amalgamation of schemes 1 and 2 since if the tolerance ϵ is very small, then the scheme will resort to using the Newton iteration to evaluate the inverse mapping at every substep. Conversely, if ϵ is large, the guided search will be used at every substep. However, there is a cost associated with performing the error check that will be discussed in the next sections.

5. TIME INTERPOLATION OF UNSTEADY DATA

Unsteady data fields, for example, velocity and sometimes meshes, are generally available at a finite number of discrete time levels. The integration schemes often requires data fields at intermediate time levels, which are not available, so interpolation in time is also required. As discussed in [3], this interpolation must be consistent with the order of the integration method used to maintain the required accuracy. In the present work, we have adopted an equispaced Lagrange interpolation centered around the required time level. The approximation of the vector field (7) is calculated as

$$\mathbf{u}(\mathbf{x}, t) \approx \tilde{\mathbf{u}}(\mathbf{x}(\zeta), t) = \sum_I \sum_j a_j \hat{\mathbf{u}}_I(t^j)\phi_I(\zeta), \tag{59}$$

where a_j are the coefficients of the Lagrange interpolation in time.

A question about numerical efficiency arises when considering whether the time interpolation is to be performed before or after the spatial interpolation. These operations may not commute and therefore could result in different computational load and numerical approximation.

Using the convention that the term within brackets is evaluated first, a spatial interpolation on each available time level followed by time integration can be written as

$$\mathbf{u}(\mathbf{x}, t) \approx \sum_j a_j \left[\sum_I \hat{\mathbf{u}}_I(t^j)\phi_I(\zeta) \right]. \tag{60}$$

Following the notation of Fig. 10, for a given time t^j, the value at point P^j is interpolated from the known modal/nodal values A^j, B^j, C^j, D^j, E^j, and F^j. Then the interpolation in time is performed over points P^j, for $j = n - 1, n, n + 1, n + 2$, to compute the field over point $P_{n+\alpha}$.

Alternatively, we could interpolate first in time over all the available points and then in space, i.e.,

$$\mathbf{u}(\mathbf{x}, t) \approx \sum_I \phi_I(\zeta) \left[a_j \sum_j \hat{\mathbf{u}}_I(t^j) \right]. \tag{61}$$

Here a time interpolation is employed to obtain the values at points $A^{n+\alpha}$, $B^{n+\alpha}$, $C^{n+\alpha}$, $D^{n+\alpha}$, $E^{n+\alpha}$, and $F^{n+\alpha}$. For instance, the value at $A^{n+\alpha}$ is obtained using a Lagrange

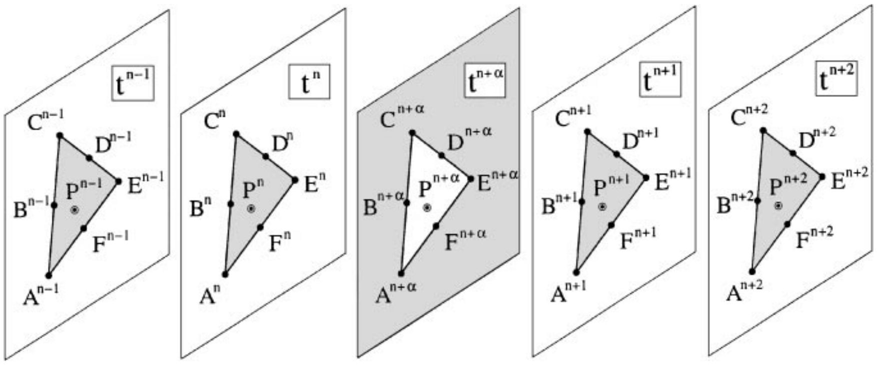


FIG. 10. Third-order time interpolation in two dimensions. The domain has been discretized using quadratic triangular elements. The discrete data is available at time levels t^{n-1}, t^n, t^{n+1}, and t^{n+2} and the data is required at time $t^{n+\alpha}$, $0 < \alpha < 1$. Points A^j, B^j, C^j, D^j, E^j, and F^j; $j = n - 1, n, n + 1, n + 2$, are the quadrature or nodal points over which the solution is available. Points P^j; $j = n - 1, n, n + 1, n + 2$ are the points resulting from spatial interpolation over the available data fields. Points $A^{n+\alpha}$, $B^{n+\alpha}$, $C^{n+\alpha}$, $D^{n+\alpha}$, $E^{n+\alpha}$, and $F^{n+\alpha}$ are the points resulting from time interpolation of the quadrature or nodal points. Point $P^{n+\alpha}$ is the point at which the field is required.

interpolation through the values at points A^{n-1}, A^n, A^{n+1}, and A^{n+2}. Using the values at $t^{n+\alpha}$, an interpolation in space is then performed to obtain the value at $P_{n+\alpha}$.

For high-order elements the polynomial order plays a dominant role. The operation represented by Eq. (60) is computationally more expensive because it requires the evaluation of the interpolating polynomial basis at each time level while that given by Eq. (61) requires a single evaluation of the basis functions only and is, therefore, more efficient for high-order methods.

6. VALIDATION AND PERFORMANCE ANALYSIS

The performance of the algorithms will be tested first in Section 6.1 using an analytic solution in a simple geometry and then in Section 6.2 using a geometrically more complex configuration.

6.1. *Analytic Domain*

In the first series of tests we have considered a range of schemes including Euler/RK1, RK2, RK3, and RK4 using the meshes shown in Fig. 11 which contain 37 prismatic elements adjacent to the boundary and 46 tetrahedral elements in the rest of the domain. The curvature of the surface is represented by positioning one of the triangular faces of the prismatic elements on the cylindrical surface as shown in Fig. 11(a). The procedure employed to generate such meshes is described in detail in [12]. The elemental boundary curvature can be removed to obtain a linear surface representation as shown in Fig. 11(b). In that case, all the prismatic elements have a local to global mapping which is nonconstant. We should point out that the Jacobian of the mapping for prisms is likely to be nonconstant even for linear elements. Nevertheless, it is possible to obtain a constant elemental Jacobian when using straight-sided tetrahedral elements, and this condition is enforced in the high-order mesh generation procedure.

To validate the particle tracking procedure, we first consider an analytic unsteady solution, previously used in [3], within these meshes of the form

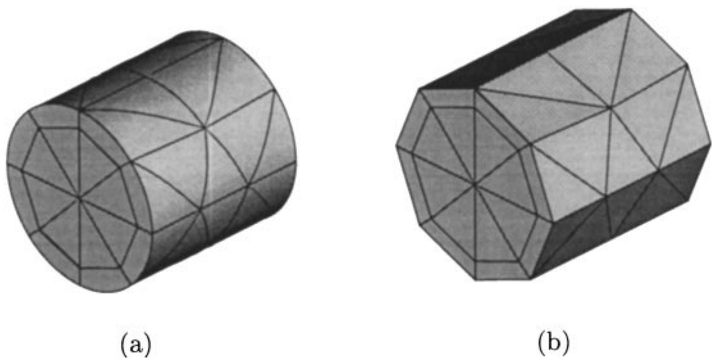


FIG. 11. Mixed prismatic and tetrahedral meshes using 83 elements within a cylindrical pipe: (a) curved elements, (b) straight-sided elements.

$$u = -x$$
$$v = -0.1y$$
$$w = -20ze^{-0.1t},$$

which corresponds to a particle location at time t of

$$x(t) = x_0 e^{-t}$$
$$y(t) = y_0 e^{-0.1t}$$
$$z(t) = z_0 e^{200(e^{-0.1t}-1)},$$

where x_0, y_0, z_0 are the initial coordinates of the particle. The starting point is taken to be $x_0 = 0.5$, $y_0 = 0.25$, $z_0 = 0.35$, which corresponds to an initial velocity of $u(0) = -0.5$, $v(0) = -0.25$ and $w(0) = -7$. The solution of this system is relatively stiff because of the rapid decay of the z coordinate in time. Therefore, we have considered a relatively short final time $T = 0.2$. The integration was carried out using a set of time steps ranging from $\Delta t = 0.01$ to $\Delta t = 0.001$. Figure 12(a) shows a comparison of the convergence rate of the guided search algorithm with error checking, using a tolerance $\epsilon = 10^{-12}$, and the physical space scheme for all the Runge–Kutta schemes. The error in these tests is measured as the distance between the final location of the particle and the analytic solution relative to the exact value.

Figure 12(b) shows the converge rate of the three schemes using the RK4 time integration where we observe that the guided search algorithm with no error checking produces a linear convergence rate only. Since the trajectory determined by the guided search is influenced by the nonlinear elemental mapping the deterioration of convergence is to be expected. We note however that the hybrid algorithm maintains a fourth-order convergence rate until a level of 10^{-9} where the error of the elemental mapping saturates the results. Since the analytic solution is only available in the physical space, the parametric velocity has to be calculated using the numerically determined Jacobian matrix. At a polynomial order $P = 10$, the error

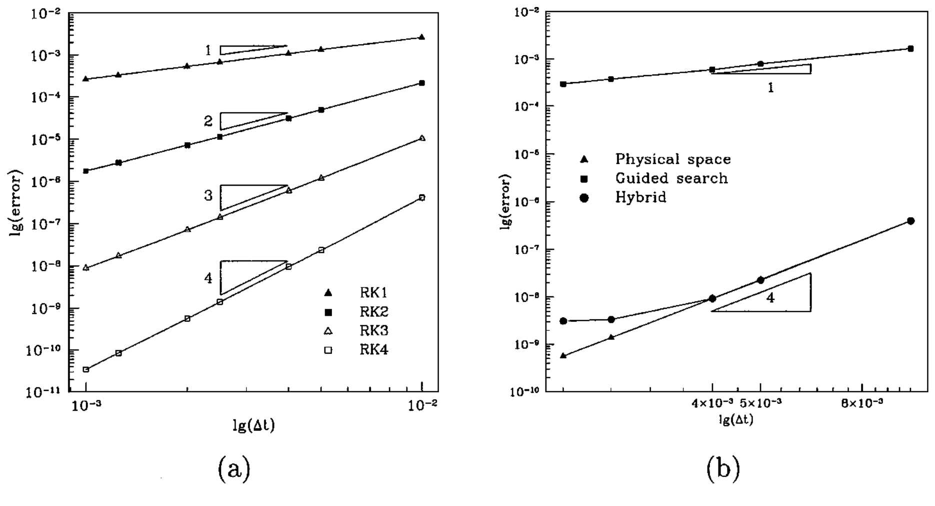
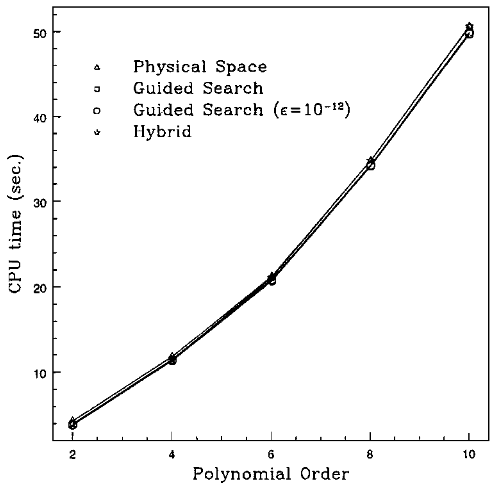


FIG. 12. (a) Temporal convergence for different Runge–Kutta schemes using an analytic solution with the physical space and guided search ($\epsilon = 10^{-12}$) algorithms. (b) Temporal convergence for the RK4 scheme using the physical space, guided search, and hybrid algorithms.



Poly. Order	Physical space	Guided search ($\epsilon = 10^{-12}$)	Guided Search	Hybrid
2	4.34	3.99	3.84	4
4	11.82	11.47	11.32	11.47
6	21.25	20.85	20.68	21.08
8	34.75	34.23	34.1	34.77
10	50.57	49.84	49.63	50.52

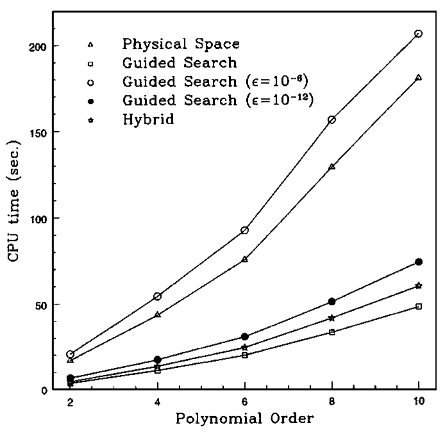
FIG. 13. CPU time to march 100 particles on a circle of radius $0.45D$ over 100 time steps through a mesh of tetrahedral elements with linear mappings as a function of polynomial order using a RK4 scheme.

introduced by this operation appears to be $O(10^{-9})$. This is supported by the observation of earlier saturation if the polynomial order of the approximation is reduced.

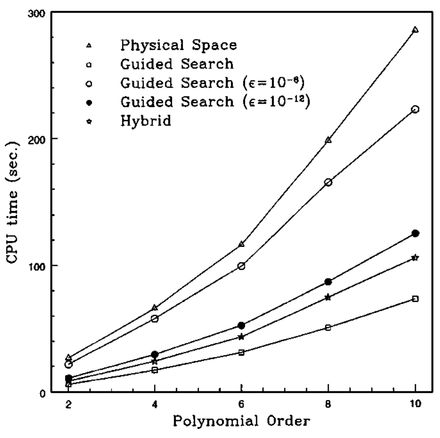
The solution of this problem using the straight-sided tetrahedral mesh, shown in Fig. 11(b), with constant Jacobians leads to identical results to those shown in Fig. 12 for all schemes.

To compare the relative merit of each scheme we also require to assess their computational cost. Figures 13 and 14 show timings for two numerical experiments. In both cases, a circular ring of particles is released within the computational domains, depicted in Fig. 11, where the velocity was set to be the numerical solution to the Poiseuille flow. All the tests were performed using either the RK2 or RK4 schemes over 100 time steps with a time step of 0.0125 on a dedicated SGI R10000 195MHz computer. In the first test, shown in Fig. 13, we consider a ring of diameter $0.45D$ chosen to guarantee that all particles remain within the tetrahedral mesh. Since all these elements have linear mappings, the results indicate that there is practically no difference between the computational cost of the different algorithms for a fixed polynomial order. The scaling within this region is approximately $O(P^{1.6})$ and is well below the asymptotic scaling value of $O(P^3)$ which is expected when the interpolation of the velocity field dominates.

Releasing a ring of particles of a larger diameter, $0.9D$, produces a dramatic difference in the timings included in Fig. 14. These particles now travel within the curved prismatic region of the computational domain and are therefore more sensitive to the nonlinear mapping introduced by the deformation of the elements. In this example we have considered both the RK2 and RK4 schemes. In the RK2 case the guided search with the small tolerance $\epsilon = 10^{-6}$ is the most costly while in the RK4 case the physical space particle tracking is



(a)



(b)

RK2 Data

Poly. Order	Physical	Guided Search ($\epsilon = 10^{-12}$)	Guided Search ($\epsilon = 10^{-6}$)	Guided Search	Hybrid
2	16.91	20.6	6.81	3.85	4.74
4	43.17	54.06	17.06	10.93	13.32
6	75.19	92.35	30.49	19.61	24.11
8	128.77	155.94	50.69	32.88	41.28
10	180.15	205.7	73.57	47.83	59.67

RK4 Data

Poly. Order	Physical	Guided Search ($\epsilon = 10^{-12}$)	Guided Search ($\epsilon = 10^{-6}$)	Guided Search	Hybrid
2	26.61	21.59	10.84	6	8.52
4	66.09	57.93	29.28	17	23.97
6	116.09	98.98	52.29	30.58	43.25
8	197.7	164.99	86.41	50.51	74.26
10	284.45	221.89	124.67	72.92	105.24

FIG. 14. CPU time to march 100 particles at a radius of $0.45D$ over 100 time steps through a region discretized by prismatic elements with nonlinear mappings as a function of polynomial order using (a) RK2 scheme, (b) RK4 scheme.

the most costly. However, they are both approximately four times more expensive than the guided search algorithm without error checking. If we introduce error checking in the guided search algorithm, the cost depends on the error tolerance ϵ. Reducing the error tolerance will force the algorithm to perform the inverse mapping at each substep to correct the error introduced by the guided search. As the value of ϵ is reduced, this scheme becomes more similar to the physical space algorithm and, as shown in the RK2 case, it can even be more expensive than the physical space algorithm because of the extra checking being performed. However, setting a tolerance of 10^{-6} is sufficient to recover most of the speed-up of the hybrid scheme without error checking. Nevertheless, the best is still achieved by the hybrid algorithm which, from our previous tests, also showed better temporal convergence characteristics.

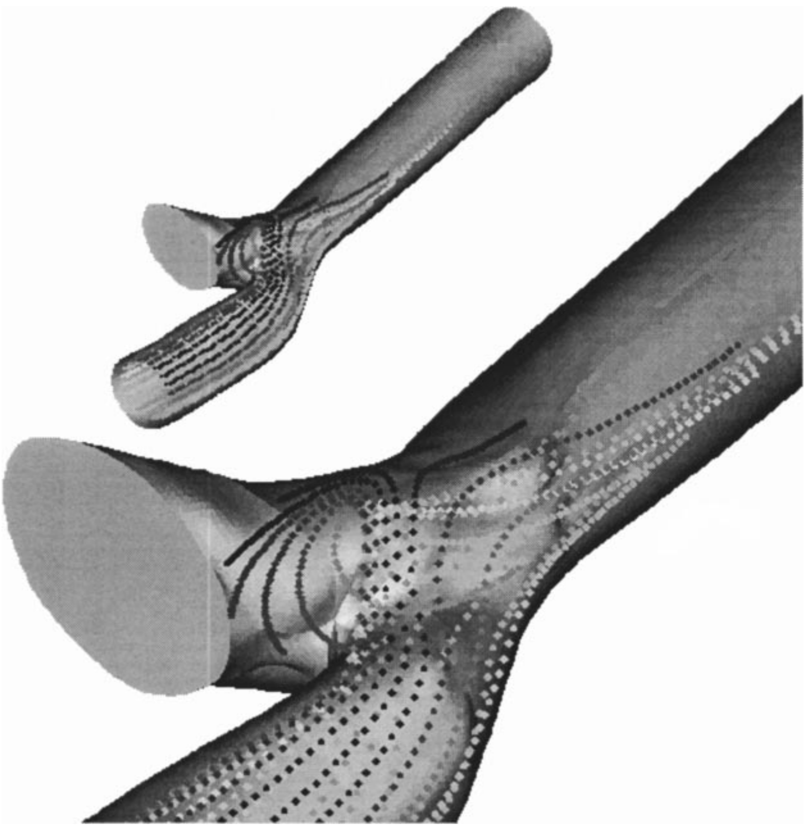


FIG. 15. Streamlines in a reconstruction of a porcine, coronary bypass junction using a prismatic boundary layer mesh with interior tetrahedral elements. Twenty equispaced particles were released at the inflow in a circular ring of diameter of $0.8D$ and time marched for a time $T = 8.0$.

6.2. *Complex Domain*

To compare the results of the previous section with a more realistic configuration we have considered the computational domain shown in Fig. 15 of a reconstruction of a porcine, coronary bypass junction. The mesh was generated using the procedure described in [12] and consisted of 749 prismatic elements creating a boundary layer mesh surrounding 1720 tetrahedral elements with constant elemental Jacobian. The geometry and steady-state solution were represented by a polynomial of order $P = 6$ and, for this test, we released 20 equispaced particles on a ring of diameter $0.8D$. The particles were time marched using different time steps and both second- and fourth-order Runge–Kutta schemes to a final time $T = 8$. This time period was chosen so that all particles remained within the computational domain. The particle were then marched backward in time to assess the error which was calculated as the distance between the initial and final position of the particles. Figure 15 shows that the released particles follow a range of trajectories incorporating a recirculation cell at the junction as well as a stagnation point region.

Figure 16 shows the average error over the 20 particles which have been marched forward and backward over a total time period $T = 0.8$. As before, the physical space and hybrid algorithm both demonstrate the correct order convergence rate. The guided search algorithm, with error checking using a tolerance $\epsilon = 10^{-6}$, also converges at the correct rate until the

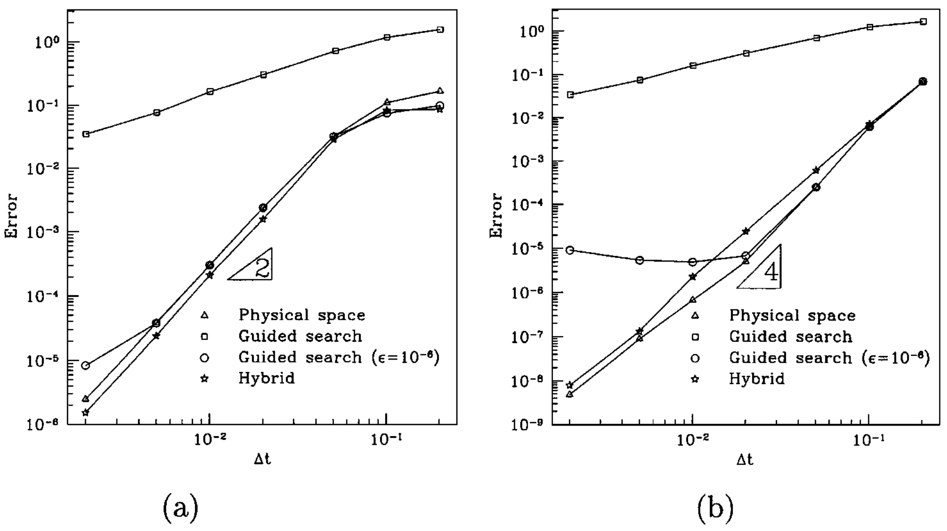


FIG. 16. Error versus time step in logarithmic scale for all algorithms. (a) RK2 scheme, (b) RK4 scheme.

guided search error reaches 10^{-6}, becomes predominant, and the overall error saturates. Finally, the guided search algorithm without error checking once again converges at a considerably slower rate.

Figure 17 compares the average CPU cost per time step for each particle. As the time step is reduced, the cost per step is also reduced for all schemes except the guided search without error checking which demonstrates a relatively time step independent speed. The reason for the dependence of the other scheme is the reduction in boundary intersections and their corresponding searching and iterative procedures. Unlike in the previous analytic domain computations, the particles now do not necessarily remain within the curved prismatic element close to the domain walls and so we do not see the very large difference between the guided search ($\epsilon = 10^{-6}$) and hybrid algorithms over the physical space algorithm for

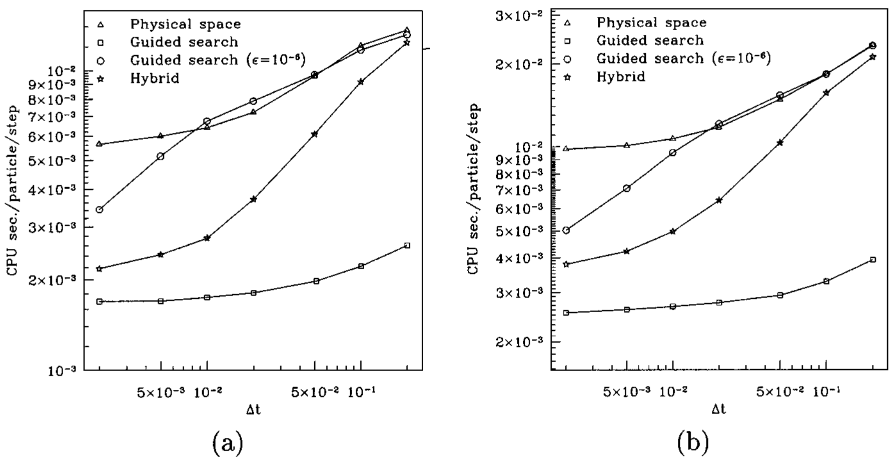


FIG. 17. Average CPU time per particle per time step in seconds as a function of time step for all algorithms. (a) RK2 scheme, (b) RK4 scheme.

large time steps. Using a time step of $\Delta t = 0.2$, the guided search without error checking is more than five times faster for both the RK2 and RK3 schemes but is inaccurate. As the time step is reduced, the guided search ($\epsilon = 10^{-6}$) and hybrid algorithms demonstrate a faster reduction in CPU time than the physical space algorithm. This is to be expected since the number of Newton–Raphson iterations required at each substep is also reduced. With a time step of $\Delta t = 0.002$, the hybrid algorithm is 2.6 times faster than the physical space algorithm for both the RK2 and RK4 schemes. At this time step, the guided search ($\epsilon = 10^{-6}$) is 1.6 and 1.9 times faster than the physical space algorithm for the RK2 and RK4 schemes, respectively. The better error of the hybrid algorithm over the guided search, with and without error checking, and a CPU cost which lies in between these schemes clearly makes this the most attractive approach to particle tracking independent of the Runge–Kutta scheme.

7. CONCLUSION

This paper has discussed alternative approaches to calculate particle trajectories using high-order spatial approximations on unstructured meshes and a Runge–Kutta integration in time. The Runge–Kutta schemes presented here have used a fixed integration time step. The role of variable time stepping has not been discussed but all these schemes could be combined with different temporal strategies to introduce time step error control such as embedded Runge–Kutta methods with local extrapolation [9].

Particle tracking algorithms on high-order meshes that use either the physical space or the parametric space rely on nonlinear procedures to calculate the trajectories. This increases considerably the calculation cost when compared with such implementations using meshes of linear elements.

To reduce the computational cost, a novel alternative hybrid approach has been proposed. This scheme advances a particle in both the physical and the parametric space within an element and uses a linear searching algorithm, the *guided search*, to move across elements. The guided search utilizes piecewise linear trajectories based upon the linear substeps of the Runge–Kutta schemes and therefore does not require nonlinear iterations. We have shown that this procedure is exact for elements with a constant Jacobian of the elemental mapping.

The guided search has been implemented in conjunction with particle tracking schemes using the physical or parametric spaces, and their performance has been assessed using a set of analytical and computational, linear and high-order, velocity fields.

For particle tracking in the physical space, it has been found that the guided search could provide reasonable estimates of the final position of the particle which, combined with suitable error checking, can produce a two- to threefold increase in speed on model problems. However, the best approach is obtained by combining particle tracking in the parametric space with a guided search using the velocity in physical space across boundaries. This scheme has also shown between a two- to threefold speed-up in both analytical and geometrically complex model problems.

ACKNOWLEDGMENTS

This research has benefited from the financial contribution of the Bupa Foundation and the provision of computer resources on the SGI Origin at the Biomedical Visualization Center and the Fujitsu AP3000 at the Parallel Computing Center, both at Imperial College. Their support is gratefully acknowledged.

REFERENCES

1. AMTEC Engineering Inc., *Tecplot Version 8.0 User's Manual* (1999).
2. J. Bonet and J. Peraire, An alternating digital tree (ADT) algorithm for geometric searching and intersection problems, *Int. J. Num. Meth. Eng.* **31**(1) (1990).
3. D. L. Darmofal and R. Haimes, An analysis of 3-D particle path integration algorithms, *J. Comput. Phys.* **123**, 182 (1996).
4. D. E. Edwards, *Scientific Visualization: Current Trends and Future Directions*, Technical Paper 92-0068 (AIAA Press, Washington, DC, 1992).
5. A. Goublomme, B. Draily, and M. J. Crochet, Numerical prediction of extrudate swell of a high-density polyetilene, *J. Non-Newtonian Fluid Mech.* **44**, 171 (1992).
6. R. Haimes and M. Giles, *VISUAL3: Interactive Unsteady 3D Visualization*, Technical Paper 91-0794 (AIAA Press, Washington, DC, 1991).
7. IBM Corporation, Thomas J. Watson Research Center, *IBM Visualization Data Explorer version 3.1.4, User's Guide*, 7th ed. (September 1997).
8. D. Kincaid and W. Cheney, *Numerical Analysis*, 2nd ed. (Brooks/Cole, Pacific Grove, CA, 1996).
9. J. D. Lambert, *Numerical Methods for Ordinary Differential Systems, The Initial Value Problem* (Wiley, New York, 1993).
10. W. R. Schowalter, *Mechanics of Non-Newtonian fluids* (Pergamon, Elmsford, NY, 1978).
11. G. Em Karniadakis and S. J. Sherwin, *Spectral/hp Element Methods for CFD* (Oxford Univ. Press, London, 1999).
12. S. J. Sherwin and J. Peiró, Mesh generation in curvilinear domains using high-order elements, *Int. J. Num. Meth. Eng.*, in press.
13. J. Sun and R. J. Tanner, Computation of steady flow past a sphere in a tube using a PTT integral model, *J. Non-Newtonian Fluid Mech.* **54**, 179 (1994).
14. C. Upson, T. Faulhaber Jr., D. Kamins, D. Laidlaw, D. Schegel, J. Vroom, R. Gurwitz, and A. van Dam, The Application Visualisation System: A computational environment for scientific visualization, *IEEE Comp. Graph. Appl.* 30 (1989).
15. P. P. Walatka, P. G. Buning, L. Pierce, and P. A. Elson, *PLOT3D User's Manual* (NASA TM-101067, 1990).
16. T. C. E. Warburton, *Spectral/hp Methods on Polymorphic Multi-Domains: Algorithms and Applications*, Ph.D. thesis (Brown University, 1998).
17. T. C. E. Warburton, S. J. Sherwin, and G. Em Karniadakis, Unstructured *hp*/spectral elements: Connectivity and optimal ordering, in *Proceedings of the International Conference on Computational Mechanics, Hawaii*, 1995.