INTERACTIVE SIMULATION AND VISUALIZATION OF MASSLESS, MASSED, AND EVAPORATING PARTICLES

Darin Diachin
Department of Mechanical Engineering
Northwestern University
Evanston, IL 60208

Lori Freitag
Mathematics and Computer Science Division
Argonne National Laboratory
Argonne, IL 60439

Daniel Heath
Computer Science Department
Iowa State University
Ames, Iowa 50013

Jim Herzog
Scientific Computing and Computational Mathematics Department
Stanford University
Stanford, CA 94305

William Michels
Nalco Fuel Tech
1001 Frontenac Rd
Naperville, IL 60566

Address Queries to:

Lori Freitag
MCS 221/C236
Argonne National Laboratory
Argonne IL 60439
(630) 252-7246

ABSTRACT

Most software packages available for particle tracing focus on visualizing steady or unsteady vector fields by using massless particle trajectories. For many applications, however, the use of massed and evaporating particles would provide a model of physical processes that could be used in product testing or design. In this article we describe the TrackPack toolkit, which provides an integrated interface for computing massless, massed, and evaporating particle trajectories in steady flow. In all cases, we assume a noncoupled model and compute particle trajectories through an existing vector field by numerically integrating with forward Euler, fourth-order Runge-Kutta, or an analytic streamline calculation. The TrackPack software effort was motivated by an industrial application to model pollution control systems in industrial boilers. We briefly describe the project and the visualization environment, and we demonstrate the necessity for massed, evaporating models in this application.

1 INTRODUCTION

A large number of industrial applications use particle dynamics to reduce costs in product design, development, and testing cycles. Such applications include spray dryers, injective pollution control systems, spray coolers, and fuel atomizers. For these applications, liquid or solid particulate matter is suspended in a gas flow, and the particle temperature and velocity histories are required to analyze the application design. Computing and displaying the trajectories of massless particles through a velocity field is a commonly used technique for the visualization of both steady and unsteady fluid flows (for example, see [2, 9, 12, 14, 15]). However, massless particle trajectories are not sufficient to accurately represent the particulate motion in the fluid; the particle's mass, velocity, and other physical parameters must be included in the model.

One numerical approximation used to solve these applications is noncoupled, two-phase flow models for which it is assumed that the particles do not significantly affect fluid flow. This assumption is best applied in cases where the mass and volume of particulates are considerably less than the mass of continuum fluid, such as for soot particles in gaseous combustion by-products or a collection of particulate matter in an electrostatic precipitator. For these cases an engineer typically will compute the velocity field for the gas flow, inject the particles into system, compute and analyze the particle trajectories, and modify the problem parameters accordingly until application design or testing is complete.

In certain applications, however, this noncoupled assumption is not appropriate. For instance, in many spray dryers the injected liquid produces considerable cooling of the gases and will dramatically influence gas flows in the vicinity of the nozzle. In those applications, the influence of the particulates must be included as source terms in the governing momentum, energy, and mass balance equations. Furthermore, injected particles will influence the turbulent properties of the continuum. To compute the solution of fully coupled steady-state particle/gas phase problems, one must accumulate the influence of the particles for each cell. Those effects can then be included into the governing balance equations as cell-by-cell sources of momentum, energy, and so forth.

In this article, we present an integrated system for the interactive simulation and visualization of massless, massed, and evaporating particles in a *noncoupled* flow system. This system consists of three components: a software package called TrackPack, which computes the particle dynamics of massless, massed, and evaporating particles in steady flow; the display device and graphics software; and a communication layer that passes information between the two. The display device used for the applications in this article is the immersive Cave Automatic Virtual Environment (CAVE) developed at the University of Illinois at Chicago [3]. The TrackPack software is linked to the CAVE visualization environment by using the CAVEcomm message-passing library developed at Argonne National Laboratory [7]. With this software, the engineer can easily define particle characteristics and sources from within the visualization environment and communicate these changes to the simulation process, which could be running remotely on a separate computer. The new results are calculated and returned for visualization and further modification in a matter of seconds.

We have organized the remainder of this article as follows. In Section 2, we describe the mathematical models used in the TrackPack software to study the dynamics of massless, massed, and evaporating particles in steady flow. The user may solve the differential equations by using a number of different numerical methods, which are also briefly discussed in Section 2. In Section 3, we present the architecture of the TrackPack software, focusing on the API and the collection of query routines necessary to obtain information from the user's data. In Section 4, we briefly describe the visualization environment, which consists of a CAVE immersive virtual reality device, a number of standard visualization techniques, and customized toolkits for particulate interaction. The communication mechanism used to link the visualization and computational processes is also

discussed in this section. In Section 5, we describe the industrial application that motivated the development of TrackPack and show the effect of mass on particle trajectory and time residence. Finally, in Section 6 we summarize the current state of the project and discuss directions for future research.

2 COMPUTING PARTICLE DYNAMICS

The dynamic behavior of massless, massed, and evaporating particles in fluid flow depends on a number of physical properties of the flow, including the fluid velocity vector field, fluid temperature, and density. In addition, the dynamics also depend on the particle's mass, density, temperature, position, and velocity. The TrackPack software provides an integrated approach to the computation of massless, massed, and evaporating particle trajectories in a noncoupled, steady flow system. In this section we briefly describe the mathematical models used in the TrackPack software for computing the three types of trajectories and the numerical methods used to integrate the differential equations.

2.1 Differential Equations

Massless particle paths in steady flow, or streamlines, are calculated by following the computed vector field by integrating the system

$$\frac{d\vec{x}}{dt} = \vec{V}_p \qquad \text{with} \tag{1}$$

$$\vec{V}_p = \vec{V}_g, \tag{2}$$

where $\vec{x}$ and $\vec{V}_p$ are the particle position and velocity vectors, respectively, and $\vec{V}_g$ is the fluid velocity vector given by the CFD solution data at the point $\vec{x}$.

The massed particle model includes dependencies in the formula for $\vec{V}_p$ for the forces on the particle resulting from fluid resistance and gravity. The system of equations governing massed particles is given by Equation (1) and

$$\frac{d\vec{V}_p}{dt} = \frac{18\mu_g(\vec{V}_g - \vec{V}_p)}{\rho_p D^2} + \frac{\rho_p - \rho_g}{\rho_p}\vec{g}. \tag{3}$$

Here, μ_g is the viscosity of the fluid, ρ is the density, D is the diameter of a particle that is assumed to be spherical, and $\vec{g}$ is the gravitational acceleration vector. This formula applies to low Reynolds number flows and does not include the secondary effects found at higher Reynolds numbers.

To include the effect of evaporation in the model, we start with Equations (1) and (3). To efficiently account for the processes of heat and mass transfer, we make some simplifying assumptions. We assume that the evaporation is heat transfer limited and that the droplet heating time is short compared with the droplet evaporation time. This latter assumption is valid for very small particles. Thus, the temperature of the particle rises to its boiling point and then begins to evaporate. This process is described by the equation

$$\frac{dT_p}{dt} = \frac{N_{Nu}\pi k D(T_g - T_p)}{(m_p c_p^p)}, \tag{4}$$

where N_{Nu} is the Nusselt number, k is the thermal conductivity of the fluid, T_p and T_g are respectively the temperature of the particle and the gas, and c_p^p is the specific heat of the particle. Once

a particle reaches its boiling temperature, all further heat gains from the fluid cause mass loss from evaporation without further changes in temperature. The evaporation is described by the equation

$$\frac{dm_p}{dt} = \frac{N_{Nu}\pi k D(T_g - T_p)}{H_v}, \tag{5}$$

where H_v is the heat of vaporization of the particle.

2.2 Numerical Solution

A number of numerical integration techniques are available to solve the systems of differential equations given in the preceding subsection (see [4] for an overview). In general, given any system of ODEs

$$\frac{d\vec{x}}{dt} = \vec{f}(\vec{x}(t))$$

and initial condition $\vec{x}_0$, the numerical methods provide an approximate solution in the form of a sequence of values $\vec{x}_i$ for $i = 0, \ldots, N$, with $\vec{x}_i \approx \vec{x}(t_i)$ for some discrete time t_i. We note that for the case of massless particles, the function $\vec{f}$ is simply the velocity field and $\vec{x}$ is the particle's position in physical space. For massed and evaporating particles, the function $\vec{f}$ is a vector containing expressions from Equations 1, 3, 4, and 5 as appropriate.

Two commonly used integration techniques are the forward Euler and fourth-order Runge-Kutta (RK4) methods. The forward Euler technique is given by

$$\vec{x}_i = \vec{x}_{i-1} + h_i\ \vec{f}(\vec{x}_{i-1}), \quad i = 1, \ldots, N,$$

where the ith time step, h_i, is given by $t_i - t_{i-1}$. RK4 is defined by

$$\begin{aligned}
\vec{x}_i &= \vec{x}_{i-1} + \frac{1}{6}\left(F_1 + 2F_2 + 2F_3 + F_4\right), \\
F_1 &= h_i \vec{f}(\vec{x}_{i-1}), \\
F_2 &= h_i \vec{f}(\vec{x}_{i-1} + F_1/2), \\
F_3 &= h_i \vec{f}(\vec{x}_{i-1} + F_2/2), \\
F_4 &= h_i \vec{f}(\vec{x}_{i-1} + F_3).
\end{aligned}$$

The accuracy and stability of these methods critically depend on the time step h_i. For massless particle trajectories in linearized flow fields, these stability requirements are well understood [4], and TrackPack includes a variation of the criteria given in Darmofal and Haimes to ensure both accuracy and stability [4]. The criteria they use bounds local errors arising in massless particle integration through the linearized flow field

$$\frac{d\vec{x}}{dt} = \frac{\mathrm{d}\vec{u}}{\mathrm{d}\vec{x}}\vec{x} \tag{6}$$

and, therefore, can be used to control the global error for trajectories in this flow field.

To facilitate the discussion of stability and accuracy of integration techniques for this flow field, we first give Darmofal and Haimes's notation for pertinent quantities.

- The eigenvalues of the velocity tensor, $\partial\vec{u}/\partial\vec{x}$, are λ_i. Estimates of the eigenvalues are generally more computationally efficient to compute, and we denote these estimates $\tilde{\lambda}_i$.

- $g(\lambda_i h)$ is a scheme-dependent growth factor. For Equation 6 the exact growth factor is $g_e(\lambda_i h) = e^{\lambda_i h}$. For the forward Euler and RK4 methods, the growth factors are the linear polynomial and fourth-order polynomial approximations to g_e, respectively.
- $E(\lambda_i h) = \frac{|g(\lambda_i h) - g_e(\lambda_i h)|}{|g_e(\lambda_i h)|}$ is the growth factor error which we control by bounding the time step size.

Using these definitions, Darmofal and Haimes choose the time step, h_i, to be

$$h_i = min\left(\frac{\eta l}{|\vec{u}|}, \frac{|\lambda_i h|_\epsilon}{|\tilde{\lambda}_{max}|}\right).$$

The first quantity, $\frac{\eta l}{|\vec{u}|}$, ensures that a minimum number of time steps are taken in each cell and contains a measure of the local cell size, l , the allowable fraction of the cell size the particle may travel in a single step, η, and the magnitude of the local velocity, $|\vec{u}|$. The second quantity, $\frac{|\lambda_i h|_\epsilon}{|\tilde{\lambda}_{max}|}$, results in a time step that maintains a bound on the growth factor error function, $E(\lambda_i h)$, where $|\lambda_i h|_\epsilon$ is the smallest magnitude $\lambda_i h$ for which $E(\lambda_i h) < \epsilon$.

When computing massless particle trajectories in TrackPack, we use a variation of the criterion given above to ensure that the maximum error per unit time (rather than at each time step) is bounded. Thus, we normalize the growth factor error by h so that $E(\lambda_i h)/h < \epsilon$. The resulting equation for h is nonlinear, and we use the method of false position to find the solution [8]. We note that the growth factor error evaluated in this way can be used to bound time steps in linear flow fields or flow fields for which we are using linear interpolation techniques.

For nonlinear flow fields, massed particles, and evaporating particles, the theory for stability and accuracy of the integration techniques is less well understood, and further research must be done to quantify the growth error functions. To maintain accuracy and stability for the nonlinear ODEs in TrackPack, we use heuristic constraints to ensure that relative changes in particle characteristics are bounded. In particular, we choose the time step that results in a change of less than 10 percent in particle velocity, temperature, and mass. We also enforce that a minimum number of time steps are taken in each cell.

We emphasize that the forward Euler technique provided in TrackPack must be used with caution. In general, this first-order technique is more prone to error and stability problems than is RK4. In fact, using the growth error function given above, we found that maintaining small global errors in a massless particle trajectory required excessively small time steps for the forward Euler technique, which led to large round-off error [6]. In practice, we have found that for the application problems considered here, only slight differences exist in the trajectories computed by forward Euler and RK4. Thus, the method is useful for depicting trends in large-scale flow features in near real time and can be used for initial analysis and product design. However, we recommend that final analysis and design be performed with RK4.

2.3 Analytic Solution Using Linear Tetrahedra

In addition to these numerical methods, we also provide an analytic solution technique for massless particle trajectories in linear tetrahedra (see also [13]). This algorithm is based on a specialized version of the fourth-order Runge-Kutta (SRK4) method described in [15]. In SRK4 the computation at each time step is reduced to a matrix-vector multiplication and a vector-vector addition, provided the tetrahedron geometry and data are static and a constant time step is used. We note that the time step used with SRK4 must be carefully chosen to maintain stability within the tetrahedron.

The analytic approach uses linear interpolation to derive the equation for massless particle trajectories. The differential equation that defines particle position within the tetrahedron is given by

$$\begin{aligned} \frac{d\vec{x}(t)}{dt} &= A\vec{x}(t) - A\vec{x}_0 + \vec{u}_0, \\ A &= UB, \end{aligned} \tag{7}$$

where $U = [\vec{u}_1 - \vec{u}_0, \quad \vec{u}_2 - \vec{u}_0, \quad \vec{u}_3 - \vec{u}_0]$ is the matrix of physical velocity differences at the tetrahedal vertices, $\vec{u}_0$ is the velocity at the tetrahedron's origin, and B contains the transformation from the canonical tetrahedron to physical space.

The solution of Equation 7 depends on the rank of the matrix A. Diachin and Herzog give a complete solution for rank zero matrices through rank three matrices in [6]. In each case the computation of the streamline through the tetrahedron is reduced to a 3×3 matrix-vector product plus a vector addition

$$\vec{x}(t+h) = H\vec{x}(t) + \vec{d}, \tag{8}$$

where H and $\vec{d}$ are characteristic of the tetrahedron and time step and are computed only once.

Results show that this method is as efficient as the SRK4 method described in [15] but is not restricted to the same stability condition for the time step [6]. Moreover, for linear velocity fields, this method is faster than forward Euler using a constant time step and gives the exact solution for an arbitrary time step.

3 SOFTWARE ARCHITECTURE

Based on the techniques described in the preceding section, we have developed a portable software package called TrackPack to compute the trajectories of massless, massed, and evaporating particles. The TrackPack code is written primarily in ANSI C and C++, but uses Fortran77 for the numerical kernels of the integration routines. This code has been successfully installed and run on a variety of computer architectures, including the IBM SP system, SGIs, and Sun workstations.

The TrackPack library and the application code interact at two distinct levels. At a high level, the user interacts with the library API to set various parameters and to request particle trajectories. At a low level, the user is required to write six query routines that allow the TrackPack library to obtain the information it needs from the user's data structures. Through the use of these routines we have created a particle tracing library that is independent of mesh type or user data structures. In this section we give an overview of the TrackPack software by describing these two levels in some detail.

3.1 Use of TrackPack at the Application Level

Several API routines are available that allow the user to set a variety of parameters for particle tracing, including the solver used, the desired interpolation scheme, the default time step used, whether adaptive time stepping is used, the minimum acceptable time step, and the maximum number of time steps per trajectory. In addition, two functions allow the user to request particle trajectories and return the trajectory with either equally spaced points (useful for the animation of large numbers of streamlines) or at every computed point. Note that equally spaced points can be returned independent of whether adaptive time stepping is used. A simple example showing the use of the TrackPack API is given in Figure 1.

```
(1.)  PTsetTimestepAdaptFlag(1);
(2.)  PTsetTimestep(.00001);
(3.)  PTsetNumStepsPerCell(6);
(4.)  PTsetAvgGlobalErrorPerUnitTime(.000001);
(5.)  PTsetMaxNumTimeSteps(1000);
(6.)  PTsetInterpolation(TRILINEAR_INTERPOLATION);
(7.)  PTsetODEsolver(RUNGE_KUTTA_4_ODE_SOLVER);
(8.)  PTgetStreamline(MASSLESS_PARTICLE, TEMPERATURE,
          (void *) data, particle,
          &streamline, &numPts);
```

Figure 1: Application use of TrackPack showing typical calls to the library routines

The first four lines of Figure 1 control the time step size by setting the adaptive time stepping flag to true, the minimum time step used to 10^{-5}, the minimum number of time steps used in each cell to six, and the average global error per unit time to 10^{-6}. The maximum number of time steps to be computed for this trajectory is set to 1000 in line five. In line six, the user sets the interpolation routine to a trilinear scheme, which is currently implemented only for hexahedral meshes. For general use, the user can also choose linear interpolation on tetrahedral cells. If the user data is contained in hexahedral cells, the decomposition to tetrahedral cells is managed by the TrackPack library using the technique described in [11]. In line seven, the user sets the solver option to the RK4 integration scheme. As discussed in the preceding section, the user can also choose to use a first-order forward Euler integration routine or the analytic solver for massless particles.

Finally, in line eight, the user requests a trajectory of a massless particle with interpolated temperature values at the streamline points, where `TEMPERATURE` is an integer defined by the user. Massed or evaporating particles can also be requested simply by changing the first argument of this call to `MASS_PARTICLE` or `EVAPORATING_PARTICLE`, respectively. To prevent the need for global data structures, the `data` argument is a void pointer to the user's data structure that is passed through to the query routines. This data structure generally contains information about the computational space, including the mesh and the discrete solution of the vector and scalar fields. In particular, this data structure should contain enough information to enable the program to locate any coordinate position in a cell of the mesh and return appropriate scalar and vector quantities requested by the user (such as `TEMPERATURE`). The `particle` argument is a data structure defined by TrackPack that contains information pertaining to the current position, mass, temperature, velocity, cell location, and time stepping quantities of the particle. Finally, the `streamline` and `numPts` arguments return the streamline position coordinates, interpolated velocity and scalar fields, and the number of points in the streamline.

3.2 Query Routines

The particle tracing routines in TrackPack depend on information contained in the user data structure `data`. To allow the maximum flexibility in the applications with which TrackPack can be used, we have designed the library to interact with the application code at a low level using a set of predefined query routine calls. The structure of the query routines was designed assuming the general case in which the user data is contained in *cells* (e.g., bricks, hexahedra, tetrahedra) with the velocity and scalar fields defined at the vertices of the cells.

To provide information to the TrackPack library, the user must write six routines. These routines include three simple data collection routines to gather velocity, scalar, and coordinate information from the vertices of a mesh cell into the double arrays used in the Fortran kernels of TrackPack. In addition, there are two data initialization routines: one for the particle data structure (which contains a user-defined cell structure) and another for the user-defined cell data structure.

Finally, the most complicated routine that the user must write is a search routine that returns a pointer to the cell that contains a given particle position. We assume that the information contained in the user-defined `data` structure is sufficient to locate the cell for the coordinate given. In addition to the `data` data structure, a pointer to the previous cell location (if any) is an input argument to this query routine. This can significantly reduce the cost of the search routine because at each time step, t_i, the particle is often either in the same cell or in a neighboring cell of its position at the previous time step, t_{i-1}.

4 THE INTERACTIVE VIRTUAL ENVIRONMENT

To visualize and interact with the results of the TrackPack calculations, we have developed an environment using the CAVE immersive display device, several standard flow visualization techniques, and customized toolkits designed for use with the TrackPack software. We have discussed this work in some detail in [5], and we give only a brief description here.

The visualization environment is based on the CAVE technology developed at the Electronic Visualization Laboratory at the University of Illinois at Chicago [3]. Users are immersed in the virtual environment by stepping into a ten-foot cube that has stereo images projected onto two walls and the floor. Several users may be immersed simultaneously in the same virtual environment and interact with the same computational model. One user is tracked by an electromagnetic tracking system, and the image orientation is calculated with respect to the head position of that user. Objects in the CAVE are manipulated by the user by means of a hand held wand, a three-dimensional analogue of the mouse on current computer workstations.

We have provided several options for data visualization that can be used to obtain insight into vector and scalar data fields resulting from the computational simulations of the fluid flow. In particular these toolkits include static vector fields, interactively requested animated streamlines, flow fields, and isosurfaces. Once the flow characteristics are understood, the engineer can study the dynamics of particles injected into the flow, using toolkits designed for the request of groups of massed or evaporating particles emanating from a single point, as is typical of injector configurations. We give a more detailed discussion of the use of these toolkits in the next section in the context of a particular industrial application.

We envisioned the primary use of TrackPack to be the computation of large numbers of particle trajectories for product design or product coverage evaluation. While new trajectories are being computed, the user should have the freedom to visualize the computational space and/or existing trajectories without CPU competition from the particle tracing calculations. Thus, the optimal performance of the interactive system is obtained when the TrackPack calculations are separated from the visualization software. Hence, these components must be able to communicate with each other in an effective, low-cost manner. In addition, this communication layer must be portable so that the numerical calculations in TrackPack can be performed on a variety of architectures. The CAVEcomm library [7], developed at Argonne National Laboratory, provides the performance aspects needed from the software perspective, and we use this as our message-passing interface. The CAVEcomm communication library uses a client-server model in which a broker mediates the communication between TrackPack and one or more CAVE environments. Following the initial

handshake, the two applications are able to interact directly without mediation from the broker; the visualization process requests general particulate information directly from TrackPack. In addition, the ability to connect multiple CAVE environments to the particle tracking software and to each other allows remote collaboration to be possible (see [5] for a description of our use of this feature).

5 AN INDUSTRIAL APPLICATION

The TrackPack software effort was motivated by a joint project between Argonne National Laboratory (ANL) and Nalco Fuel Tech (NFT) to design pollution and slag control systems for commercial boilers and incinerators. These systems use an injection-based process to spray noncatalytic reagents directly into the combustion flue gases, where they react to reduce pollutants or slag. Optimal performance of the system is obtained by careful placement of the injectors in the boiler with respect to the flue gas temperatures, velocity fields, and locations of slag buildup. Critical to the design of this system, then, is a tool that enables engineers to interactively place the injectors and obtain a quick evaluation of spray coverage and time residence of the injected particles.

The velocity and scalar fields used in this application were generated by using the PHOENICS CFD package [1]. The computational mesh is a static, Cartesian grid containing roughly 150,000 cells, and the query search routine for finding the cell containing a given coordinate is a simple index calculation. We use trilinear interpolation for this application, which enables the interpolated velocity fields to vary continuously between volume cells, thereby ensuring that the trajectories are a smooth representation of the discrete data.

For each pollution control system designed, a virtual boiler geometry is constructed by using blueprint information. A typical municipal waste incinerator unit is shown in Figure 2. The TrackPack software is used to compute streamlines for the visualization of the flue gas flows. These streamlines are colored by a user-defined scalar value or by an optimal temperature window scheme. The optimal temperature window coloring scheme highlights the areas of ideal injector placement with bright colors and indicates regions where the temperature is too hot with lavender and regions that are too cold with dark purple. These areas can also be highlighted through the use of isosurfaces. In Figure 3 we show the upper and lower temperature bounds for optimal performance of the injectors in this unit, 1850° F and 1650° F, respectively.

Once the optimal areas for injection are identified, the engineer can place up to 25 injectors on the exterior side walls of the boiler and evaluate the spray coverage. Representative sprays for each injector are calculated by using a statistical model consisting of 500 massed, evaporating particles emanating from a singe point in a 30 degree cone [5]. The trajectories are displayed by using continuous paths, which can be colored by injector source, by a user-defined scalar quantity such as temperature or particle mass, or by the optimal temperature window scheme.

Existing injectors can be relocated by selecting them with the wand and dragging them to the new location. As the injector location changes, 20 representative particle trajectories are computed and visualized as quickly as possible (about half a second). Hence, the engineer can quickly evaluate the potential of a new injector location. We note that the optimal temperature window coloring scheme is particularly useful during this phase of the design process. In addition, the specific spray configuration for each injector can be modified to study the effects of changes in the initial particle mass, speed, and distribution.

In this application, the use of massed, evaporating particles is critical to the effective evaluation of injector location. In Figure 4, we show the same injector location with three different spray configurations that vary only in the initial mass of the injected particles. From right to left, the particles in the frames of Figure 4 are an average of 50 μm, 250 μm, and 750 μm in initial diameter.

All particles have an initial speed of 30 m/s. The optimal temperature window scheme is used to color the trajectories, and, for maximum injector effectiveness, the evaporation of the particles (and hence the release of the chemical reagents) would take place primarily where the sprays are brightly colored.

In the case where the particle size is very small, evaporation is almost immediate, and the injected spray covers very little of the flue gas flow in the boiler. As the particles increase in mass, they are no longer subject to immediate evaporation, and we clearly see that their initial momentum carries them to the interior of the boiler domain. Finally, in the third frame, we see the effect of injecting particles that are too large. In this case, the momentum of the particles carry them across to the opposite wall and very little chemical reagent is released in the optimal zone. Massless, nonevaporating particles would have no momentum to carry them to the boiler interior and would flow tangent to the velocity field at the point of entry. In addition, a nonevaporating particle trajectory would traverse the entire boiler interior and no information could be obtained regarding the time residence of the particle. Hence, the massed, evaporating particle model is critical to the evaluation of product coverage and time residence in the boiler.

6 CONCLUDING REMARKS

In this article we have described an interactive system for the modeling and analysis of massless, massed, and evaporating particle dynamics in noncoupled, two-phase, low Reynolds number flow models. We gave the mathematical models used to describe these particles and the numerical routines used to integrate the resulting systems of ODEs. These routines have been implemented in the TrackPack software that provides a simple API to the particle dynamics calculations and has been designed to be independent of particular user data structures and mesh types. We showed the use of this software in an industrial application where the modeling of massed and evaporating particles is critical to the effective design of pollution control systems. Portions of this system have been incorporated into the Nalco Fuel Tech production environment and have significantly reduced the time required to analyze injector configurations for pollution control systems.

We are currently working to expand the range of applications that this tool could be used with by developing the software necessary to solve the coupled flow model. Advanced applications are being developed to solve the governing fluid dynamics calculations in parallel using adaptive, unstructured mesh computations [10]. We anticipate that the combination of adaptive mesh technologies and parallel computing will allow real-time interaction with more complex models.

ACKNOWLEDGMENTS

We would like to thank the referees for their insightful comments that resulted in significant improvements to this paper. This work was supported by the Mathematical, Information, and Computational Sciences Division subprogram of the Office of Computational and Technology Research, U.S. Department of Energy, under Contract W-31-109-Eng-38.

REFERENCES

[1] PHOENICS Reference Manual (CHAM/TR200), Phoenics, London, 1991.

[2] S. Bryson and C. Levit. The virtual windtunnel: An environment for the exploration of three-dimensional unsteady flows. In *Proceedings of the Visualization '91 Conference*, pages 17–24. IEEE Computer Society Press, 1991.

[3] C. Cruz-Neira, D. J. Sandin, and T. A. DeFanti. Surround-screen projection-based virtual reality: The design and implementation of the CAVE. In *ACM SIGGRAPH 93 Proceedings*, pages 135–142. ACM, 1993.

[4] D. L. Darmofal and R. Haimes. An analysis of 3D particle path integration algorithms. *Journal of Computational Physics*, 123:182–195, 1996.

[5] D. Diachin, L. Freitag, D. Heath, J. Herzog, B. Michels, and P. Plassmann. Remote engineering tools for the design of pollution control systems for commercial boilers. *International Journal of Supercomputing Applications*, 10.2:208–218, 1996.

[6] D. Diachin and J. Herzog. Analytic streamline calculations for linear tetrahedra. In *13th AIAA Computational Fluid Dynamics Conference*, pages 733–742, 1997.

[7] T. L. Disz, M. E. Papka, M. Pellegrino, and R. L. Stevens. Sharing visualization experiences among remote virtual environments. In *Proceedings of the International Workshop on High Performance Computing for Computer Graphics and Visualization*, pages 135–142. Springer-Verlag, 1995.

[8] C. Gerald and P. Wheatley. *Applied Numerical Analysis, Fifth Edition.* Addison-Wesley Publishing Company, 1994.

[9] R. Haimes. pv3: A distributed system for large-scale unsteady CFD visualization. In *AIAA 32nd Aerospace Scienced Meeting and Exhibit*, number AIAA 94-0321, 1994.

[10] M. Jones and P. Plassmann. Computational results for parallel unstructured mesh computations. *Computing Systems in Engineering*, 5(4-6):297–309, 1994.

[11] D. Kenwright and D. Lane. Optimization of time-dependent particle tracing using tetrahedral decomposition. In *Proceedings of the Visualization '95 Conference*, pages 321–328. IEEE Computer Society Press, 1995.

[12] D. Lane. Scientific visualization of large scale unsteady fluid flow. In *Scientific Visualization Surveys, Methodologies and Techniques*, pages 125–145. IEEE Computer Society Press, 1996.

[13] K. Sawada. Visualization of unsteady vortex motion in the flow over a delta wing. In *Proceedings of the 5th International Symposium on Computational Fluid Dynamics*, volume 3, pages 69–74, 1993.

[14] S. Shirayama. Processing of computed vector fields for visualization. *Journal of Computational Physics*, 106:30–41, 1993.

[15] S.K. Ueng, K. Sikorski, and Kwan-Lui Ma. Fast algorithms for visualizing fluid motion in steady flow on unstructured grids. In *Proceedings of the Visualization '95 Conference*, pages 313–319. IEEE Computer Society Press, 1995.

Darin Diachin is a graduate student in the Theoretical and Applied Mechanics Program at Northwestern University. His interests include the parallel solution of PDEs using reproducing kernel

particle methods, analytic streamline calculations, and interactive visualization tools. He graduated with a B.S. in applied mathematics from the Colorado School of Mines in December 1994 and obtained an M.S. in applied mathematics from Northwestern University in December 1997.

Lori Freitag is a staff scientist in the Mathematics and Computer Science Division at Argonne National Laboratory. Her current research interests include developing interactive, immersive real-time visualization environments, scalable algorithms for unstructured mesh computation, and solution of large-scale applications within the SUMAA3d project. She received a Ph.D. in applied mathematics from the University of Virginia in 1992.

Daniel Heath is a graduate student in the Computer Science Department at Iowa State University. His primary research interests include the development of a three-dimensional GUI toolkit for the CAVE and the development of visualization tools for computational fluid dynamics applications. He graduated from Edinboro University of Pennsylvania in 1994 with a B.A. in computer science.

James Herzog is a graduate student at Stanford University in the Scientific Computing/Computational Mathematics program. His primary research interests include developing portable, general-purpose computational libraries for particle dynamics and the parallel solution of PDEs. He received a B.S. in mathematics from Bowling Green University in May 1996.

William Michels is the manager of advanced computing systems for Nalco Fuel Tech, Naperville, IL. His responsibilities focus on the development and use of new computational tools to assist in the design of state-of-the-art pollution control systems for combustors and incinerators. He holds a Ph.D. from the University of Minnesota (1987) and a B.S. from the University of Texas (1980), both in chemical engineering.