SAND2012-5336C

Virtual Radiometer Model

Isaac Hunsaker[*], Philip J. Smith[†], David Glaze[‡], Jeremy Thornock[§]

Department of Computational Thermal and Fluid Dynamics

Abstract

There exists a general need to compare radiative fluxes from experimental radiometers with fluxes computed in Thermal/Fluid simulations. However, prior to the release of the virtual radiometer model, the simulation suite lacked the ability to predict fluxes to objects with small view angles. As a result, validation efforts have suffered. This model was developed as a validation tool for users to be able to specify arbitrary view angles, orientations, and locations of multiple radiometers, and receive as the output, high-accuracy radiative fluxes to these radiometers. The virtual radiometer model is a Reverse Monte Carlo Ray Tracing approach adapted to meet these user specifications and runs on unstructured meshes.

Contents

1 Governing Equations **3**

2 User-Specified View Angle and Orientation **4**
2.1 User-Specified View Angle 4
2.2 User-Specified Orientation 7

3 Ray Marching In Unstructured Meshes **8**

4 Verification and Validation **10**
4.1 Verification of Ray Distribution 10
4.2 Verification of Participating Media Physics 10
4.3 Verification of Ray Convergence 12
4.4 Verification of Segment Length Convergence 14
4.5 Validation 15

5 Efficiency Considerations **16**

[*]PhD Candidate, Department of Chemical Engineering.
[†]Professor, Chair, Institue for Clean and Secure Energy.
[‡]Member Technical Staff, Sandia National Laboratories.
[§]Research Assistant Professor, Department of Chemical Engineering.

6 Future Work **17**
6.1 Parallel Capable 17
6.2 From Post-Process To Run-Time 17
6.3 Automation of Multiple Timesteps 17
6.4 Reflecting Rays 18
6.5 Use of Boundary Conditions for Wall Temperatures 18

7 User Tips **18**

Nomenclature

ϵ	=	Statistical error
κ	=	Absorption coefficient
Ω	=	Solid Angle
ϕ	=	Radiometer rotation angle about the x axis
ϕ_r	=	Ray azimuthal angle
ψ	=	Radiometer rotation angle about the z axis
σ^2	=	Statistical variance
θ	=	Radiometer rotation angle about the y axis
θ_r	=	Ray polar angle
θ_v	=	Radiometer View Angle
A	=	Rotation matrix
b	=	x,y,z coordinates rotated into the appropriate orientation
C	=	Computed Solution
E	=	Exact Solution
F	=	View factor
I_{in}	=	Incoming intensity
I_{out}	=	Outgoing Intensity
ir	=	Ray index number
L	=	Length of one side of a cube
l	=	A point along a ray
N	=	Number of rays traced per radiometer

n = Radiometer normal vector

q = Radiative flux

$Q3$ = 3rd Quadrant of a 2D Cartesian Grid

$Q4$ = 4th Quadrant of a 2D Cartesian Grid

q_{eff} = Effective emissive power

R = Uniformly distributed random number

subscript b = Black body

subscript cv = Control Volume

subscript i = Incident

subscript n = Net

subscript o = Outgoing

subscript w = Wall

T = Temperature

x = x,y,z coordinates prior to rotation into the appropriate orientation

1 Governing Equations

The governing equation for reverse monte carlo ray tracing in nonhomogeneous, participating media was developed by Walters and Buckius. Specifically,

$$I_{i,k} = \int_0^{l_k} I_{b,cv}[T(l')]\kappa(l')exp[-\int_{l'}^{l_k}\kappa(l'')dl'']dl' + I_{o,sur}(T_w)exp[-\int_{l_w}^{l_k}\kappa(l')dl']. \tag{1}$$

In a discretized domain, we can assume piecewise homogeneity and pose equation 1 in the following form,

$$I_{i,k} = \sum_{m=1}^{M}\left(I_{b,cv}(T_m)(exp[-\int_{lm2}^{l_k}\kappa(l')dl'] - exp[-\int_{lm1}^{l_k}\kappa(l')dl'])\right) + I_{o,sur}(T_w)exp[-\int_{l_w}^{l_k}\kappa(l')dl']. \tag{2}$$

See (Sun, 2009) for the full derivation. The intensities from each of the rays of a radiometer can then be weighted according to the solid angle that each ray subtends. Assuming uniform distribution of the rays, each ray will subtend $\frac{\Omega}{N}$ steridians, where Ω is the solid angle of the radiometer and N is the number of

rays used in the simulation. The radiative flux can then be calculated from the intensities of the rays, weighted by the discretized solid angle,

$$q_i = \frac{\Omega}{N} \sum_{r=1}^{N} I_i(ir) cos(\theta(ir)). \tag{3}$$

Some of the radiometers used in experimental measurements are calibrated against a black body that subtends the entire field of view of the radiometer. These radiometers therefore report a value that is an effective emissive power,

$$q_{eff} = \frac{q_i}{(1 - cos(\theta_v/2))cos(\theta/2)}.$$

The output of the model contains both q_i and q_{eff}.

Note: Experimental radiometers measure a net flux, $q_n = q_i - q_o$. Traditionally in radiation texts, $q_n = q_o - q_i$. In the fire sciences, however, because $q_o << q_i$ it is generally accepted to report a positive flux *to* a radiometer, neglecting q_o. Justification for this is increased when the radiometers are liquid cooled, decreasing q_o as is the case with the experimental radiometers mentioned in section 4.5.

2 User-Specified View Angle and Orientation

Perhaps the most novel feature of this algorithm is the capability to handle, at run-time, a user-specified orientation and view angle for the radiometer. The implementation of these two features is described in the following two subsections.

2.1 User-Specified View Angle

The view angle will define a solid angle about which rays must be generated and uniformly distributed. To accomplish this, two random numbers must be generated to span this two dimensional surface. The naive approach would be to generate two random numbers between 0 and 1, and simply scale them by a factor of 2π for the azimuthal angle ϕ, and a factor of θ_v for the polar angle, where $\theta_v = V/2$, where V is the view angle of the radiometer. Unfortunately, this will lead to non-uniformly distributed rays. To demonstrate, consider a solid angle of 4π *sr.*, a sphere. If one were to attempt to generate a series of rays according to the naive approach, the azimuthal and polar angles of the rays would be described as follows,

$$\phi_r = 2\pi R_1 \tag{4}$$

$$\theta_r = \pi R_2, \tag{5}$$

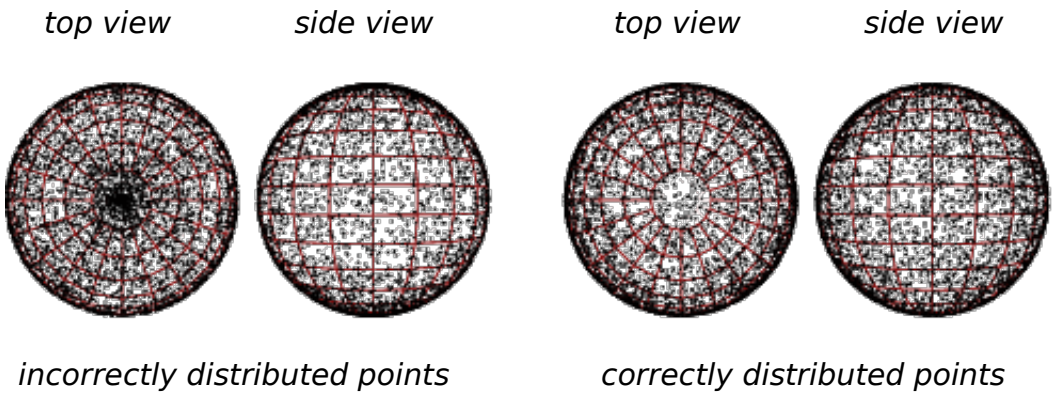

Figure 1: $left$: The naive approach of generating random numbers produces incorrectly distributed points clustered near the poles. $right$: Correctly distributed points show no clustering throughout the surface of the sphere. http://mathworld.wolfram.com/SpherePointPicking.html.

where R_1 and R_2 are two independent random numbers uniformly distributed between 0 and 1. This approach would lead to the distribution shown on the left of figure 1.

The clustering around the poles is a result of the surface area of spherical objects not being proportional to $\delta\theta$. To account for this, we must be able to produce a random number within the range of $cos(\pi)$ to $cos(0)$ (evaluated as -1 to 1), then take the arccosine of that number. Figure 2 demonstrates why this is necessary.

With this correct implementation of picking random points on a sphere, the azimuthal and polar angles are assigned as,

$$\phi_r = 2\pi R_1 \tag{6}$$

$$\theta_r = acos(2R_2 - 1), \tag{7}$$

where $(2R_2 - 1)$ yields a random number uniformly distributed between -1 and 1. Similarly, to generate random points on a hemisphere the polar angle for a given ray would be assigned as $acos(R_2)$ where R_2 has a range of 0 to 1 and corresponds to polar angles between 0 and $\pi/2$. Then, to generate points on an arbitrary solid angle that the user-specified view angle subtends, one needs to ensure that the random numbers have a range of $cos(-\theta_r)$ to $cos(\theta_r)$. This is accomplished by setting the range equal to

$$range = 1 - cos(\theta_v). \tag{8}$$

Then, to generate rays within the user-defined solid angle, the azimuthal and polar angles for a given ray are assigned as

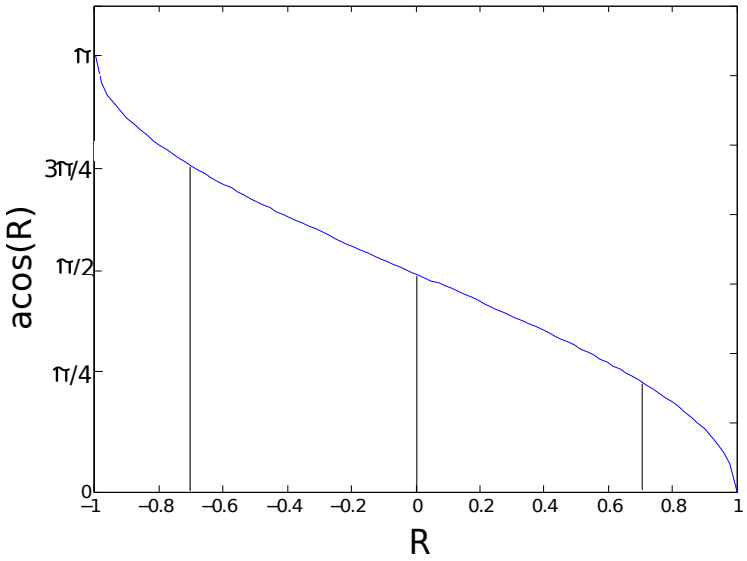

Figure 2: Distributing random points on a sphere requires the scaling by *arccosine* of the random number R, where $R = 2R_2 - 1$, such that R has a range of -1 to 1. Notice that the the range of random numbers that produces a value of θ between 0 and $\pi/4$ is 0.707 to 1. This range of possible random numbers is less than half the size of that which produces a value of θ between $\pi/2$ and $\pi/4$, specifically 0 to 0.707. This is consistent with the fact that the surface area on a sphere between $\theta = \pi/4$ *to* $\pi/2$ is more than double that of of the surface on a sphere between $\theta = 0$ *to* $\pi/4$. Scaling a random number in this manner correctly causes the probability of picking a point within a given $\delta\theta$ to be proportional to the area that $\delta\theta$ subtends in the unit sphere. This works because the differential area of an infinitesimalsolid angle is given by $d\Omega = sin(\theta)d\phi d\theta = -d\phi(cos(\theta))$.

$$\phi_r = 2\pi R_1 \tag{9}$$

$$\theta_r = acos\left(cos(\theta_v) + range * R_2\right). \tag{10}$$

These generated points, with the location of the radiometer as their origin, define the direction vectors of the rays which can then be traced through the domain.

2.2 User-Specified Orientation

From a developer's standpoint, it would be most convenient if the orientation of radiometers used in experiments were always aligned in the same direction. This, however, is not the case. To handle the variability in the radiometer orientation, the virtual radiometer model must be able to take as an input, any user-defined direction normal vector, and adjust the orientation of the rays accordingly. To accomplish this, a matrix that transforms any Cartesian point about a vector comprised of rotation angles is employed. This matrix is defined as

$$A = \begin{bmatrix} cos\theta cos\psi & -cos\phi sin\psi + sin\phi sin\theta cos\psi & sin\phi sin\psi + cos\phi sin\theta cos\psi \\ cos\theta sin\psi & cos\phi cos\psi + sin\phi sin\theta sin\psi & -sin\phi cos\psi + cos\phi sin\theta sin\psi \\ -sin\theta & sin\phi cos\theta & cos\phi cos\theta \end{bmatrix} \tag{11}$$

where ϕ, θ, and ψ represent the counterclockwise rotation angles about the x,y, and z axes, respectively. To relieve the user of the burden of determining these three rotation angles, the virtual radiometer model takes as an input, the normal vector of the radiometer and computes these rotations automatically. This is accomplished as follows,

$$\theta = acos\left(\frac{n_z}{\sqrt{n_x^2 + n_y^2 + n_z^2}}\right) \tag{12}$$

$$\psi = acos\left(\frac{n_x}{\sqrt{n_x^2 + n_y^2}}\right), \tag{13}$$

where n_x, n_y, and n_z represent the components of the vector normal of the radiometer. Note that there will never be a need to calculate a rotation about the x axis. All possible rotations can be accomplished using the other two, while fixing ϕ at 0. Due to the constraints of *arccosine*, the value of ψ must be adjusted if n_x and n_y are in the 3rd or 4th quadrants of the xy plane. Specifically,

$$if(n_x, n_y) \in Q_3 \ : \ \psi = \pi/2 + \psi_{calculated} \tag{14}$$

$$if(n_x,n_y) \in Q_4 \ : \ \psi = 2\pi - \psi_{calculated} \tag{15}$$

At this point, the rotation angles can be applied to the direction vectors of the rays via the matrix multiplication of

$$[A]x = b, \tag{16}$$

where x is the pre-rotated direction vector of a ray in Cartesian coordinates, and b is the resulting direction vector.

3 Ray Marching In Unstructured Meshes

When performing ray tracing on structured meshes, the marching algorithm (the algorithm that determines which cell will be referenced and in what order) is relatively simple due to the geometric relations that exist between the orientation of the ray and the surface normals of the cells. Optimization of this algorithm also becomes trivial as there are only 6 possible face normal vectors, which can further be reduced to three once the signs of the components of the vector are known. A 2D representation of structured-mesh ray marching is illustrated in figure 3. However, when the mesh in use is unstructured, the geometric relationships between the ray orientation and the surface normals of the element faces become complicated, and may vary greatly from step to step as illustrated in figure 4. The tracing of rays in an unstructured mesh therefore requires the implementation of one of two possible methods. The first involves walking the node connectivity by calculating intersections between the current location of the ray and the faces of the elements. The second involves selecting points *apriori* along the defined ray direction, and querying the elements to find which element owns those coordinates. While the first method will undoubtedly result in a more efficient algorithm, the development burden is much greater than that of the second as it requires information regarding the elements' neighbors, node extents, face normal vectors, etc. The second approach, however, takes advantage of existing element search routines and is much less developmentally intensive. The existing element search routine was developed by Greg Wagner and includes useful speedups such as overlaying the mesh with a coarse, structured grid. This requires the initial search to find not the element that contains a given set of parametric coordinates, but rather the "bucket" that contains these coordinates. Once this bucket has been identified, only the elements within the bucket are queried, rather than all the elements in the entire domain. For these reasons, the second approach was selected to handle ray marching in unstructured meshes.

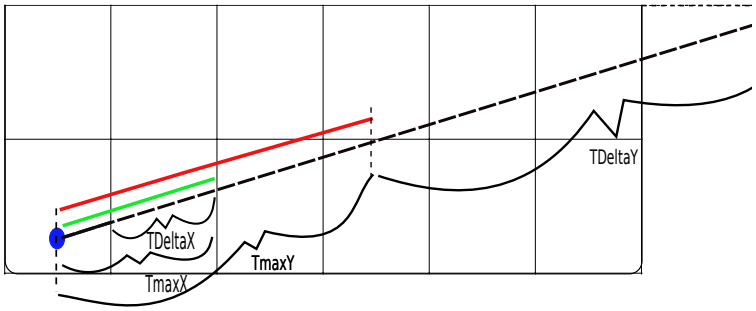

Figure 3: Ray marching in structured grids takes advantage of simple geometries to determine the next element to be referenced along a ray.

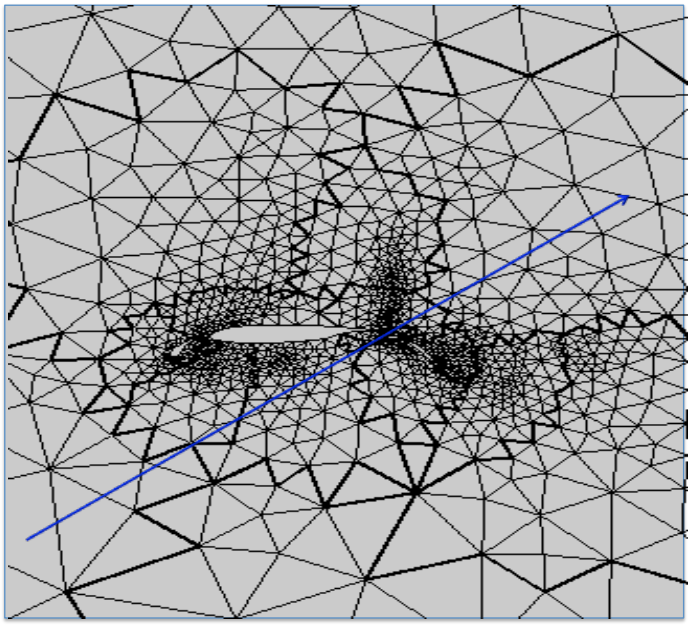

Figure 4: Ray marching in unstructured meshes becomes complicated due to the lack of simple geometric relationships between the ray orientation and the surface normals of the element faces.

4 Verification and Validation

4.1 Verification of Ray Distribution

To verify that the randomly generated rays were uniformly distributed over the solid angle of the radiometer, and that equation 3 was implemented correctly, the model was used to solve for the view factor between a circular disc and an infinitesimal surface as shown in figure 5. For a disc of radius r separated from the infinitesimal area by a distance h, the analytical solution as given by Modest for the view factor is

$$F = \frac{1}{(h/r)^2 + 1}. \tag{17}$$

The L2 error norm, defined as $\frac{\sqrt{(E-C)^2}}{E}$, where E is the exact solution and C is the computed solution, was shown to decrease with an increase in the number of rays such that at 100,000 rays, the L2 error norm was approximately 0.001. Because the distribution of the rays is of importance particularly for wide angle gauges, this L2 error norm, under the name of rayError, is printed to the log file when values of the view angle exceed $atan(0.5)$, or approximately 51°. The ray error can then be used as a metric to assess the confidence in other computed values such as the radiative-heat flux. This verification test does not exercise the participating media portion of the algorithm, so the 3D case described by Burns and Christon was employed to verify these features as explained in section 4.2.

4.2 Verification of Participating Media Physics

Exact solutions to radiation problems involving nonhomogeneous, nonisothermal, emitting, absorbing media are difficult to come by. The integro-differential Radiative Transport Equation becomes prohibitively complex for these situations, limiting the availability of analytical solutions. There are, however, a handful of well documented semi-exact solutions for participating-media radiation problems. One such solution was given by Burns and Christon in 1997. This case is described by a three dimensional cube with cold black walls, with an absorption coefficient, κ, that varies according to following the trilinear function,

$$\kappa L = 0.9\left(1 - 2 \mid \frac{x}{L} \mid\right)\left(1 - 2 \mid \frac{y}{L} \mid\right)\left(1 - 2 \mid \frac{z}{L} \mid\right) + 0.1 \tag{18}$$

where L is the length of the cube, and x, y, and z are the distances from the center of the Cartesian domain. A visualization of this absorption coefficient is given in figure 6. The semi-exact solution was computed via the Discrete Ordinates Method on a 41^3 domain. The number of ordinate directions was increased until the radiative flux divergence, $\nabla \cdot q$, converged to 10^{-5}.

The intensities computed by the virtual radiometer model were used to compute the radiative flux divergence at various locations in the domain for which a solution was given by Burns and Christon. The radiative flux divergence can be described in terms of intensity as follows

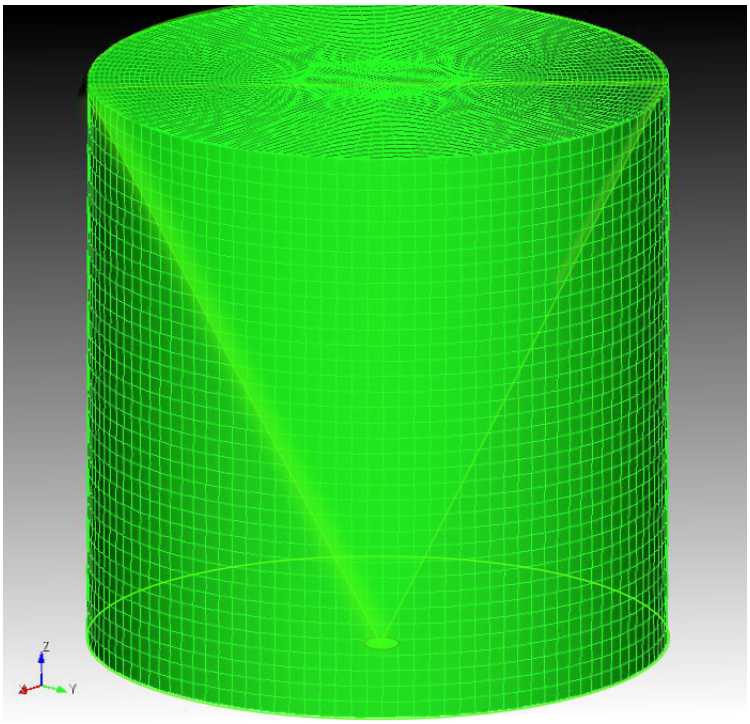

Figure 5: Schematic representation of the view factor of a circular disk as viewed by a point at the bottom of a cylinder.

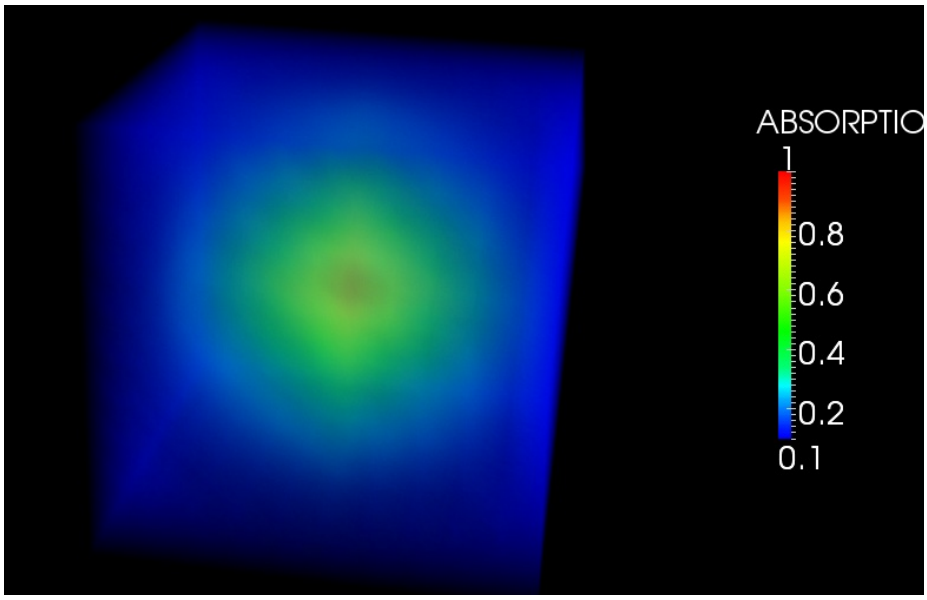

Figure 6: Absorption coefficient as specified in the case described by Burns and Christon. The absorption coefficient varies according to the trilinear function as described in equation 18 such that near the edge of the domain, the absorption coefficient evaluates to nearly 0.1, and at the center evaluates to 1.0.

$$\nabla \cdot q = \kappa \left(4\pi I_{b,out} - \int_{\Omega} I_{in} d\Omega \right), \tag{19}$$

or in discretized form as

$$\nabla \cdot q = \kappa (4\pi I_{b,out} - \sum_{ir=0}^{N} \left(I_{in}(ir) \frac{4\pi}{N} \right), \tag{20}$$

where I_{in} is described by equation 2. Although this case solves for the flux divergence rather than a surface flux, because it is a function of the incoming intensity, this case exercises the participating media functionality of the algorithm. The resulting flux divergences at the specified locations were in excellent agreement with the solutions of Burns and Christon and are illustrated in figure 7.

4.3 Verification of Ray Convergence

Convergence of the solution to the incident flux as a function of the number of rays used in the simulation is demonstrated by figure 9. Here, the virtual radiometer model was run on data produced by an 18" aluminum propellant

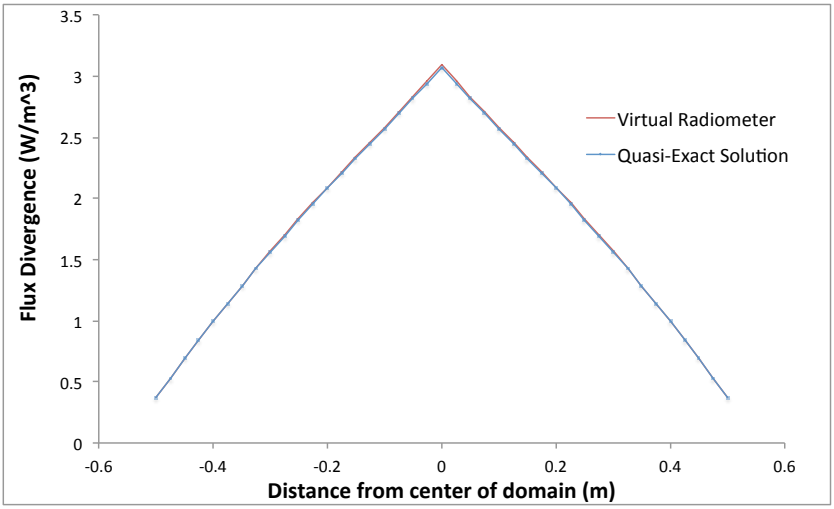

Figure 7: Results of the participating media physics verification test. The virtual radiometer model demonstrates excellent agremment with the quasi-exact solution. This plot was produced using 10,000 rays per radiometer, and resulted in an L2 error norm of 0.00305.

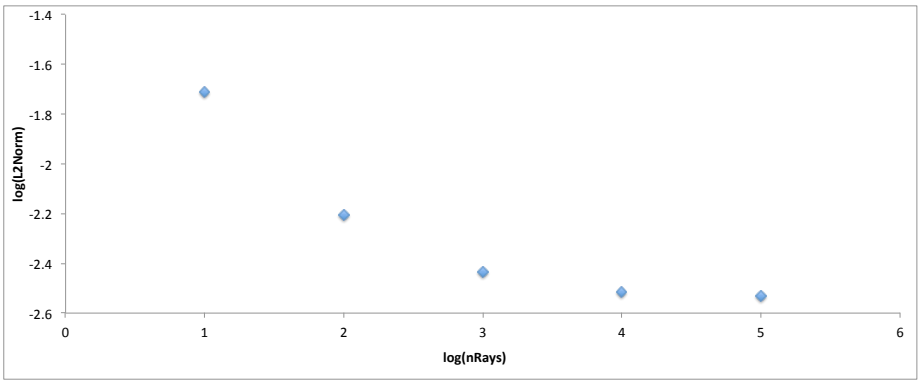

Figure 8: Convergence of the virtual radiometer results relative to the quasi-exact solution given by Burns and Christon. The flux divergences given by the virtual radiometer model at 41 spatial locations were compared with those of the quasi-exact solution. An L2 error norm of these points was calculated at increasing ray numbers ranging from 10 to 100,000. Note that as the number of rays increases, the difference between the two solutions doesn't converge to zero. This is due to the fact that the quasi-exact solution has a non-zero error which will result in an offset in the convergence. In fact, it is quite possible that at 100,000 rays, the solution produced by the virtual radiometer model is more correct than the quasi-exact solution.

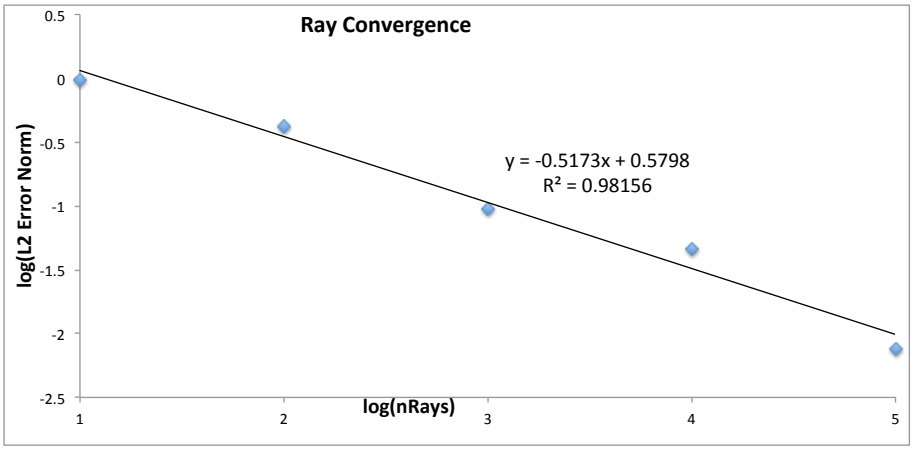

Figure 9: The accuracy of the model improves with an increase in the number of rays per radiometer. The solution converges at the expected order of approximately $-\frac{1}{2}$.

fire. The resulting fluxes produced from varying ray numbers were compared to the converged values as computed by a simulation with 1.4 million rays. The expected rate of convergence can be calculated by recognizing that due to the statistical nature of the random rays, the variance of the solution is proportional to the inverse of the number of rays,

$$\sigma^2 \propto \frac{1}{N}. \tag{21}$$

The error, which is proportional to the square root of the variance is therefore

$$\epsilon \propto N^{-\frac{1}{2}}. \tag{22}$$

Taking the log of both sides of equation 22 and moving the exponent to the beginning of the RHS yields

$$log(\epsilon) \propto -\frac{1}{2} log(N). \tag{23}$$

The expected slope is therefore $-\frac{1}{2}$ which is within 4% of the calculated slope of ray convergence of -0.5173.

4.4 Verification of Segment Length Convergence

The segment length convergence was tested as indicated by figure 10. Again, the computed values were compared with the converged solution of 1.4 million rays. Note that the x axis of the figure is the log of the inverse of the segment length; e.g. the value "2" corresponds to a segment length of 10^{-2} m. The interesting feature of this plot is the lack of convergence for segment lengths

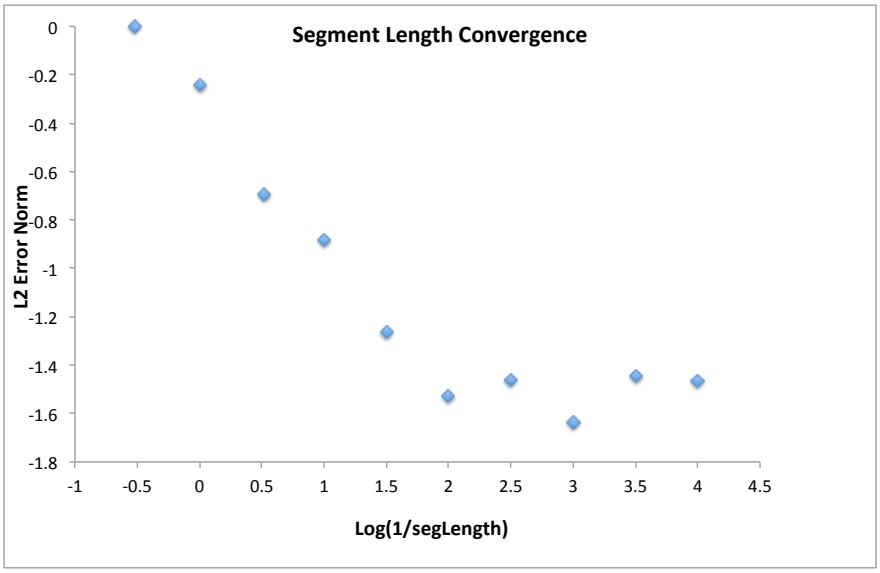

Figure 10: The accuracy of the model improves down to a segment length of approximately the size of the smallest elements in the mesh. Any further refinement of the segment length results in essentially no change in the convergence.

smaller than this value. For the mesh on which this test was performed, 10^{-2} m corresponds to the length of the smallest elements. The lack of convergence beyond this point indicates that there is no new information gained by shrinking the segment length smaller than the smallest elements. This can serve as a rule of thumb for users, specifically we recommend that the user specifies a segment length no smaller than the smallest elements in their mesh.

4.5 Validation

We performed validation testing against experimental data to compare the results of the virtual radiometer model with physical reality. The experimental measurements to compare against came from testing that was performed in the Old Flame Facility. During this testing, a series of aluminum propellant blocks was allowed to combust to completion. Each test consisted of varying sized propellant blocks with radiometers and spectrometers placed in varying arrangements. Allen Ricks has created a Fuego input file that is designed to match the parameters of one of these cases, specifically, the 18" propellant block upward burn case. We used the results of this Fuego simulation as input for the virtual radiometer model and calculated radiative fluxes at the same locations of the radiometers in the experimental setup. The results are summarized in figure 11 which demonstrates the acceptable agreement between experimental

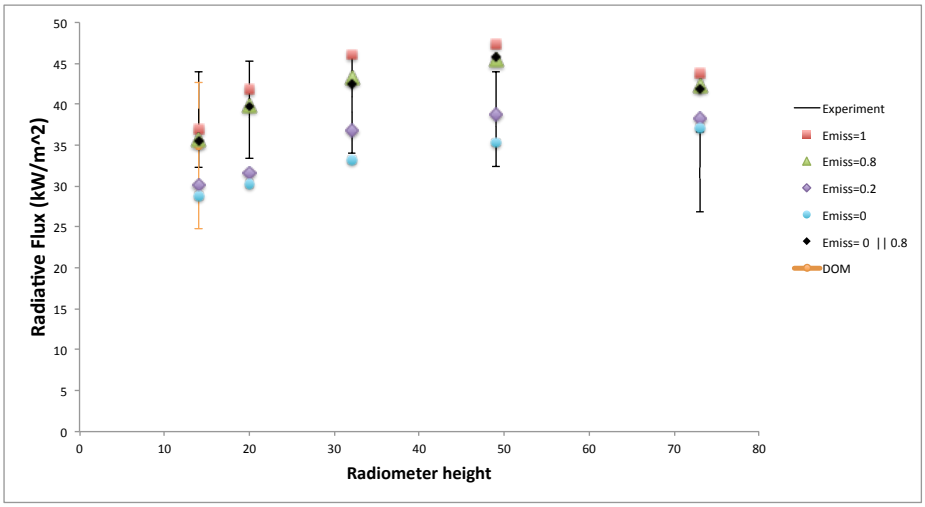

Figure 11: Validation results indicate acceptable agreement between the experimental results (black bars) and model results at varying emissivities (colored dots).

and simulation data.

5 Efficiency Considerations

Several efforts have been made to ensure that the virtual radiometer model runs efficiently. The choice of the random number generator (RNG) is one such effort. The RNG we selected is the Mersenne Twister algorithm, which produces high-fidelity equidistributed random numbers at a rate of $O(10^7)/s$ (Hunsaker, 2011).

The virtual radiometer algorithm also avoids division, and removes constants outside of summation loops. This is demonstrated in the calculation of the flux from the incident intensities, where if we assume uniformly distributed rays, $d\Omega$ is constant and equal to $\frac{\Omega}{N}$. Therefore

$$q_i = \frac{\Omega}{N} \sum_{r=1}^{N} I_i(r) cos(\theta(r)), \tag{24}$$

which, for a case with 5 virtual radiometers of 100,000 rays each, requires 500,000 fewer divisions and multiplications than the following mathematically-equivelant expression

$$q_i = \sum_{r=1}^{N} I_i(r) cos(\theta(r)) \frac{\Omega}{N}. \tag{25}$$

Perhaps the most aggressive speedup of the virtual radiometer model came when the decision was made to store the rays from the first time step for re-use in all subsequent time steps. It is important to note that the field values of temperature and absorption coefficient are not stored–these values are updated dynamically for each time step and are accessed via the C++ vector-stored parametric coordinates and master elements along the rays. Referencing the vectors in this manner rather than performing element searches along the rays results in a 5X speedup of the algorithm.

6 Future Work

The virtual radiometer model is a work in progress. The following are planned modifications of the model

6.1 Parallel Capable

The virtual radiometer model currently runs in serial. There are plans to make the model parallel capable, although it is not known when this will be accomplished. However, the overhead of running the model in serial should be negligible relative to the overhead of a thermal/ fluid simulation. As an example, the 18 inch propellant fire fuego simulation required 8 processors over 24 hours to compute a simulation of 2.5 seconds of mother-nature time. Conversely, when run using the output from this fuego simulation, the virtual radiometer model required a single processor only 74 seconds per time step when run with 100,000 rays, and a mere 0.8 seconds per time step with 1,000 rays. So unless the user requires radiometer flux data for thousands of time steps, serial runs of the virtual radiometer model should not add significantly to the overall simulation time.

6.2 From Post-Process To Run-Time

The virtual radiometer model is currently a post-processing tool. It therefore requires the user to output temperature and absorption coefficient field data for each time step of interest of the thermal/ fluid simulation. We recognize that this is not ideal, and plan to change the model from a post-processing tool to a run-time tool.

6.3 Automation of Multiple Timesteps

Currently if the user wishes to run the model at multiple time steps, a command that stitches together multi-processor information at a given time step is required. Clearly this will need to be automated when section 6.2 is implemented.

6.4 Reflecting Rays

The virtual radiometer model currently cannot handle reflecting rays. In the current version of the code, a ray will terminate when it reaches a boundary, or when the optical thickness is sufficiently high such that a subsequent step along the ray would result in less than 1% of the radiative intensity reaching the radiometer, whichever comes first. This means that for non-black boundaries and/or optically-thin media, the model will not account for the additional radiation that could have reached the origin had the ray been allowed to reflect and continue through the domain. To better represent the physics, we have plans to implement reflection conditions that will allow rays to continue traveling after an intersection with a boundary.

6.5 Use of Boundary Conditions for Wall Temperatures

When handling wall emission, the virtual radiometer model uses the specified emissivity from the input file along with the temperature of the last element traversed by a ray before the ray terminated at a boundary. Although this may be a good approximation in many cases, in scenarios where there is a sharp temperature gradient near the boundary, this approximation can lead to inaccuracies. We plan to update the model to be able to recognize on which boundary a ray has terminated, and use the appropriate boundary conditions (temperature and emissivity) for wall emission calculations.

7 User Tips

To run the virtual radiometer model, place the following block within the input file, while acknowledging the following bold-faced tips.

BEGIN VIRTUAL RADIOMETER MODEL rad01
EMISSIVITY = 1
SEGMENT LENGTH = .01
ORIENTATION = -1 0 0
LOCATION = 1.524 0 0.3556 $This is in meters at 5 feet from the center of the table and 14 inches high
VIEW ANGLE = 180
NO OF RAYS = 10000
END

For multiple radiometers, simply duplicate the above block and adjust the name "rad01" to "rad02", "rad03", etc.

Because the model currently uses only a single value for the wall emissivities, specify the emissivity in the input file to be that of the hottest boundary. Emissive power scales with T^4, so an erroneous emissivity of a cold boundary will have negligible effects, while that of a hot boundary could lead to major inaccuracies. When the future work described in section 6.4 is implemented, this tip will become obsolete.

As indicated in section 4.4, we recommend that the size of the segment length (in meters) be approximately the same size as the length of the smallest elements in the mesh.

Specify the orientation as the x,y,z components of the vector that represents the normal of the radiometer sensor surface. This vector need not be unitized.

Specify the location as the x,y,z coordinates (in meters) of the radiometer.

Specify the view angle of the radiometer to be 2X delta theta as indicated in figure 12. In most of this document, delta theta is represented by θ_v**.**

Specify the number of rays with a number at least as great as the recommended minimum of $100 + 10\theta_v$**.** Take care when assigning the number of rays. As indicated in figure 9, a factor of 100 increase in the number of rays results in a factor of 10 decrease in the error, yet this comes at the full cost of 100X increase in the simulation run-time. Furthermore, depending on the memory constraints of the processor, segment faults may begin appearing for ray numbers greater than approximately 150,000.

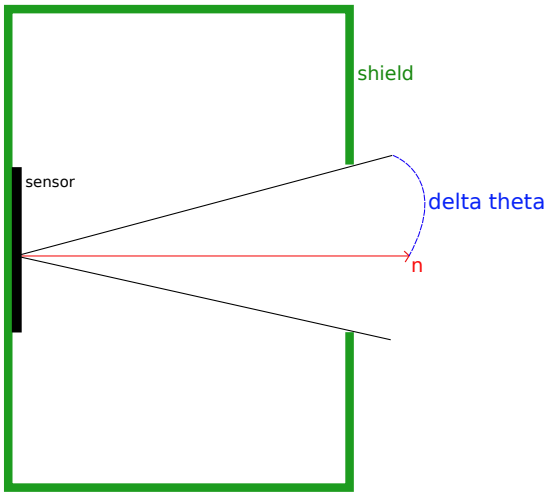

Figure 12: Simple schematic indicating delta theta. The view angle of the radiometer is 2X delta theta.

Bibliography

D. V. Walters and R. O. Buckius, "Rigorous development for radiation heat transfer in nonhomogeneous absorbing, emitting, and scattering media," Inter-

national Journal of Heat and Mass Transfer, vol. 35, no. 12, pp. 3323–3333, 1992.

X. Sun, "Reverse Monte Carlo Ray Tracing for Radiative Heat Transfer in Combustion Systems," University of Utah PhD Dissertation, 2009.

S. P. Burns and M. A. Christon, "Spatial domain-based parallelism in large-scale, participating-media, radiative transport applications," Numerical Heat Transfer, Part B, vol. 31, pp. 401–421, 1997

M. Modest. Radiative heat transfer. Academic Press, 2003.

I. Hunsaker, T. Harman, J. Thornock, P. J. Smith, "Efficient parallelization of RMCRT for large scale LES combustion simulations," 20th AIAA Computational Fluid Dynamics Conference, Honolulu, Hawaii, June, 2011.