Development and Verification of Enclosure Radiation Capabilities in the CHarring Ablator Response (*CHAR*) Code

Giovanni Salazar[*]
NASA Lyndon B. Johnson Space Center
2101 NASA Parkway, Houston, TX 77058

Justin C. Droba[*]
NASA Lyndon B. Johnson Space Center, Houston, TX 77058
JSC Engineering, Technology, and Science (JETS): Jacobs Technology and HX5, LLC

Brandon Oliver[*] and Adam J. Amar[*]
NASA Lyndon B. Johnson Space Center
2101 NASA Parkway, Houston, TX 77058

With the recent development of multi-dimensional thermal protection system (TPS) material response codes, the capability to account for surface-to-surface radiation exchange in complex geometries is critical. This paper presents recent efforts to implement such capabilities in the CHarring Ablator Response (*CHAR*) code developed at NASA's Johnson Space Center. This work also describes the different numerical methods implemented in the code to compute geometric view factors for radiation problems involving multiple surfaces. Verification of the code's radiation capabilities and results of a code-to-code comparison are presented. Finally, a demonstration case of a two-dimensional ablating cavity with enclosure radiation accounting for a changing geometry is shown.

Nomenclature

$A \subset B$	The set A is a subset of the set B
$a \in A$	a is an element of a set A
$A \mapsto B$	A maps to B
$\mathbf{u} \cdot \mathbf{v}$	Standard dot product of $\mathbf{u}$ and $\mathbf{v}$
∇f	Gradient of scalar function f
$\nabla \cdot \mathbf{v}$	Divergence of vector field $\mathbf{v}$
$\mathbf{u} \perp \mathbf{v}$	$\mathbf{u}$ is orthogonal to $\mathbf{v}$
α	Absorptivity
Γ	Domain boundary (m^2) or volume fraction in virgin material
ϵ	Emissivity
ε	Very small quantity (Section V)
θ_i	Angle between face normal and segment S connecting two faces i and j in view factor definition
$\boldsymbol{\kappa}$	Permeability tensor (m^2)

[*]Applied Aeroscience and CFD Branch, AIAA Member.

μ	Dynamic viscosity $(\mathrm{Pa}\cdot\mathrm{sec})$ or Lamé's second parameters (Pa)
$\boldsymbol{\nu}, \hat{\boldsymbol{\nu}}$	Normal vectors
$\boldsymbol{\xi}$	Separating axis
$\Pi_{\boldsymbol{\xi}}$	Orthogonal projection operator onto $\boldsymbol{\xi}$
ρ	Density $(\mathrm{kg}/\mathrm{m}^3)$ or reflectance
σ	Stefan-Boltzmann constant $(\mathrm{W}/\mathrm{m}^2\cdot\mathrm{K}^4)$ or stress (Pa)
τ	Any of several tolerances (Section V)
φ	Porosity
Ω	Domain volume (m^3)
$\dot{\omega}$	Mass source $(\mathrm{kg}/\mathrm{m}^3\cdot\mathrm{sec})$
A	Face area (m^2)
$\mathbb{B}$	"Mother box," the box to which all other tree boxes belong (head node of tree)
$\mathcal{B}$	General bounding box (node in tree)
C	Monte Carlo confidence level
E	Monte Carlo error
E_i	Emittance from face i $(\mathrm{W}/\mathrm{m}^2)$
e_o	Total internal energy (J/kg)
erf	Error function
$\mathcal{F}$	Finite element face
F_{ij}	View factor between faces i and j
F_i^{space}	View factor to space from face i
G_i	Irradiation on face i $(\mathrm{W}/\mathrm{m}^2)$
H^1	Set of square-integrable functions with first derivatives also square-integrable
H_0^1	Set of H^1 functions which vanish on the boundary
h_o	Total enthalpy (J/kg)
J_i	Radiosity from face i $(\mathrm{W}/\mathrm{m}^2)$
$\mathcal{K}$	Convex set or geometric object (e.g., finite element face)
K	Kernel $(1/\mathrm{m}^2)$
$\boldsymbol{k}$	Thermal conductivity tensor (W/mK)
L	Length of interval of projection of a convex hull onto a separating axis $\boldsymbol{\xi}$
ℓ	Length of interval of projection of an object $\mathcal{K}$ onto a separating axis $\boldsymbol{\xi}$
m	Number of rays casted from face i which intersect face j in Monte Carlo view factor computation
$\dot{m}$	Mass flux $(\mathrm{kg}/\mathrm{m}^2\cdot\mathrm{sec})$
$\bar{N}$	Number of nodes on a given face
N	Total number of rays cast from a given face in Monte Carlo view factor approach
$\mathbf{n}_i$	i^{th} node of a face or other geometric object
P	Pressure (Pa)
$\mathbf{p}$	Origin of ray (translated outside box)
pid	Processor ID
$\dot{Q}$	Energy source $(\mathrm{W}/\mathrm{m}^3)$
$\mathbf{q}$	Origin of ray on face (original origin)
q	Heat flux $(\mathrm{W}/\mathrm{m}^2)$
$\mathbb{R}$	The set of real numbers
$\mathbb{R}^d$	The set of d-tuples of real numbers
R_i	Reflection term in Sequential Reflections approach $(\mathrm{W}/\mathrm{m}^2)$
$\mathbf{r}$	Ray direction or position vector
r	Random number used in Monte Carlo view factor approach

S	Segment connecting centroid of two faces in view factor definition (m)
T	Temperature (K)
T_{eq}	Equivalent temperature (K)
t	Time (sec)
$\boldsymbol{v}$	Velocity $(\mathrm{m}/\mathrm{sec})$
$t_0^{(i)}$	Entry time of i^{th} coordinate into a box $\mathcal{B}$
$t_1^{(i)}$	Exit time of i^{th} coordinate out of a box $\mathcal{B}$
$\mathbf{x}_{\max}$	Maximal coordinate of a box $\mathbb{B}$ (upper right corner in 2D)
$\mathbf{x}_{\text{mid}}$	Midpoint of $x_{\min}$ and $x_{\max}$, centroid of box
$\mathbf{x}_{\min}$	Minimal coordinate of a box $\mathbb{B}$ (lower left corner in 2D)

Subscripts and Superscripts

∞	Far-field state
$cond$	Denotes conduction term
g	Quantity of the gas
IR	Infrared
m	Quantity of the mesh
$nonIR$	Belonging to frequency spectrum different than infrared
$rerad$	Denotes reradiation term
s	Quantity of the solid
w	Wall state

Per convention, vector quantities are denoted in **boldface**. The i^{th} coordinate of a vector $\mathbf{v}$ is denoted $v^{(i)}$.

I. Introduction

During re-entry, a spacecraft's thermal protection system (TPS) is exposed to different modes of heat transfer such as convection, conduction and radiation. When assessing the material thermal response in complex geometries, which result in partial or total enclosures, the radiation exchange of thermal energy between the different parts of an enclosure must be properly accounted for. Enclosures such as those resulting on a vehicle's TPS from micrometeoroid and orbital debris (MMOD) impact are of particular importance to spacecraft designers.

With the recent development of multi-dimensional thermal material response codes, including the capabilities to account for radiative heating is essentially a requirement. This paper presents the recent efforts to implement surface-to-surface radiation exchange capabilities in the CHarring Ablator Response (*CHAR*) code developed at NASA's Johnson Space Center. This work also describes the different numerical methods implemented in the code to compute view factors for radiation problems involving multiple surfaces. Furthermore, verification of the code's radiation capabilities are demonstrated by comparing solutions to analytical results and to other codes.

II. Governing Equations

The radiation capabilities described in this work are implemented within the *CHAR* framework. *CHAR* is a 1D/2D/3D material thermal response code which solves general heat transfer problems on decomposing charring ablators as well as non-decomposing, non-charring TPS materials, in serial or parallel. The general governing equations being solved in *CHAR* are briefly outlined in this section for completeness; however, a more complete description can be found in [1].

The equations that govern the solid/gas system of the porous charring ablator include energy and mass conservation equations for the solid as well as the Navier-Stokes equations as applied to all of the gaseous species considered. In the general case, it is possible that the pyrolysis gases react with the remaining solid, or deposit residue (coke) on the solid, but these phenomena are neglected. Under the assumptions that the pyrolysis gas is in thermochemical equilibrium and the solid and gas are in thermal equilibrium, the solid energy equation on a moving mesh reduces to a nodal mixture energy equation given by

$$\left.\frac{\partial(\rho e_o)}{\partial t}\right|_{node} = \nabla\cdot(\boldsymbol{k}\nabla T) - \nabla\cdot\left(\varphi\rho_g h_{o_g}\boldsymbol{v}_g\right) + \dot{Q} + \boldsymbol{v}_m\cdot\nabla(\rho e_o) \tag{1}$$

where ρ, e_o, φ, h_o, $\boldsymbol{v}$, and $\dot{Q}$ denote density, total energy, porosity, total enthalpy, velocity, and volumetric energy source, respectively, and the subscript g denotes a quantity with respect to the pyrolysis gases. And $\boldsymbol{v}_m$ denotes the mesh velocity at a node. Since ablators in general can be anisotropic materials, the thermal conductivity, $\boldsymbol{k}$, is a second order tensor.

If it is assumed that all solid decomposition results in pyrolysis gas generation, the gases are free to flow through the porous medium, and the gases occupy all of the pore space, then the nodal gas mass conservation equation, including mesh convection terms, is given by

$$\left.\frac{\partial(\varphi\rho_g)}{\partial t}\right|_{node} = \dot{\omega}_g - \nabla\cdot(\varphi\rho_g\boldsymbol{v}_g) + \boldsymbol{v}_m\cdot\nabla(\varphi\rho_g) \tag{2}$$

where the porous flow gas velocity is given by a porous flow law such as Darcy's law:

$$\boldsymbol{v}_g = -\frac{\boldsymbol{\kappa}}{\varphi\mu}\nabla P \tag{3}$$

The solid mass conservation equation is solved on a stationary mesh, and is simply

$$\left.\frac{\partial\rho_s}{\partial t}\right|_{\boldsymbol{x}} = \dot{\omega}_s \tag{4}$$

CHAR discretizes the governing equations according to the Galerkin finite element method. Multiplying the energy equation, Eq. (1), by a suitable test function, v, and integrating over the domain Ω while integrating the second and third terms by parts to give the natural boundary condition terms, the weak statement becomes: Find $\rho e_o \in H^1$ such that

$$\int_\Omega\left[v\frac{\partial(\rho e_o)}{\partial t} + \nabla v\cdot(\boldsymbol{k}\nabla T) - \nabla v\cdot\left(\varphi\rho_g h_{o_g}\boldsymbol{v}_g\right) - v\boldsymbol{v}_m\cdot\nabla(\rho e_o) - v\dot{Q}\right]d\Omega + \oint_\Gamma\left(v h_{o_g}\dot{m}_g + v\dot{q}_{cond_s}\right)d\Gamma = 0 \quad \forall v\in H_0^1 \tag{5}$$

where the boundary mass flux due to gas convection is

$$\dot{m}_g = (\varphi\rho_g)\,\boldsymbol{v}_g\cdot\hat{\boldsymbol{\nu}} \tag{6}$$

and the boundary heat flux is

$$q_{cond_s} = -\boldsymbol{k}\nabla T\cdot\hat{\boldsymbol{\nu}} \tag{7}$$

and $\hat{\boldsymbol{\nu}}$ is the element face normal vector.

Likewise, a Galerkin weak statement can be developed for the gas mass conservation equation, Eq. (2): Find $\varphi\rho_g \in H^1$ such that

$$\int_\Omega\left(\frac{\partial(\varphi\rho_g)}{\partial t}v - \nabla v\cdot(\varphi\rho_g\boldsymbol{v}_g) - v\boldsymbol{v}_m\cdot\nabla(\varphi\rho_g) + \dot{\omega}_s v\right)d\Omega + \oint_\Gamma v\dot{m}_g d\Gamma = 0 \quad \forall v\in H_0^1 \tag{8}$$

In *CHAR*, the system of equations is advanced in time according to first and second order implicit time integrators. The nonlinear set of governing equations are solved in parallel via Newton's method with exact complex-perturbation Jacobians, and several options are available via the PETSc library [2] to solve the implicit linear system.

There are many different boundary conditions available for the energy and gas equations in *CHAR* such as specified convective heating, specified heat flux, specified temperature, contact conduction, specified pressure, specified mass flux, thermochemical ablation using B' tables, specified ablation, CFD coupling, surface melting, etc. This work focuses on the code's enclosure radiation capabilities, which are described in further detail in the following sections.

III. Enclosure Radiation

CHAR models the surface-to-surface radiation exchange within enclosure assuming each surface behaves as a diffuse-gray surface. That is, each surface is a diffuse emitter, a diffuse absorber, and a diffuse reflector, and opaque. In addition, the emissivity and absorptivity are treated only as functions of temperature, and only cases with non-participating media are considered.

At each surface, the net radiative heat flux from a face, q_i, is the flux emitted, E_i, less the amount of the incident radiation, G_i:

$$q_i = E_i - \alpha_i G_i \tag{9}$$

where the emitted energy from each face is assumed to be entirely due to IR radiation, which is given by the well-known relationship,

$$E_i = \sigma \epsilon_i T_i^4 \tag{10}$$

CHAR has the capability to model the energy exchange in an enclosure due to two distinct spectral "groups": Infrared (IR) and non-Infrared (nonIR) radiation. IR-radiation is the most common mode of modeling the radiation exchange within an enclosure. However, modeling the heating due to a nonIR radiation source can be of great importance in problems involving solar UV radiation, or radiation originating from the shock-layer of a re-entry vehicle, for instance. Expanding the absorbed flux into IR and nonIR components, Eq. (9) can be rewritten as

$$q_i = E_i - \left(\alpha_i^{IR} G_i^{IR} + \alpha_i^{nonIR} G_i^{nonIR}\right) \tag{11}$$

where the flux emitted is always implied to be due to IR-energy as defined in (10).

The total IR incident radiation, G_i^{IR}, is a function of the energy (radiosity) leaving each of the other surfaces in the enclosure. Assuming an opaque surface, The IR-component radiosity from a surface i is given by the energy emitted by that surface, and the IR energy reflected off of that surface as follows

$$J_i^{IR} = E_i + \rho_i^{IR} G_i^{IR} \tag{12}$$

with an IR reflectance given by

$$\rho_i^{IR} = 1 - \alpha_i^{IR} \tag{13}$$

For either IR or nonIR radiation, the incident radiation can then be determined from the total radiation leaving all N surfaces in the enclosure

$$A_i G_i = \sum_{j=1}^{N} A_j F_{ji} J_j \tag{14}$$

where F_{ji} is the geometric view factor (or configuration factor), which is a measure of how much energy originating from surface i is incident on surface j. By using the reciprocity relationship, Eq. (14) can be re-written as

$$G_i = \sum_{j=1}^{N} F_{ij} J_j \tag{15}$$

which holds for the IR and nonIR spectral groups.

In problems where a partial enclosure is considered, each surface could potentially have a view to the "space" outside the enclosure, and therefore an additional contribution to the incident radiation. For a given surface i, this contribution is accounted for through a view factor to space $F_{i,space}$ and a corresponding farfield temperature T_∞ and/or a farfield non-IR flux q_∞^{nonIR}. The IR incident radiation term can be modified to account for the IR farfield source radiation as follows

$$G_i^{IR} = \sum_{j=1}^{N} F_{ij} J_j^{IR} + F_{i,space} \sigma T_\infty^4 \tag{16}$$

Similarly, the nonIR incident radiation, accounting for the farfield flux contribution, is now

$$G_i^{nonIR} = \sum_{j=1}^{N} F_{ij} J_j^{nonIR} + F_{i,space} q_\infty^{nonIR} \tag{17}$$

and the nonIR radiosity from a face is given by

$$J_i^{nonIR} = \rho_i^{IR} G_i^{nonIR} \tag{18}$$

where the only driving potential for the exchange of nonIR energy within the enclosure originates from the farfield flux contribution.

In order to solve for the desired net radiative flux from a face, q_i, the temperature at each face in the enclosure must be known either explicitly, or implicitly by solving for temperature as part of a solution of the radiative exchange. In *CHAR*, the temperature field is lagged one time step, and therefore the temperatures from the previous time step are used to solve for the radiative exchange at the current time step. With a known temperature at each face, the challenge then becomes to solve for the incident radiation and the radiosity at each face, from (16) and (17). Two approaches implemented in *CHAR* to solve for this system of equations are presented in the subsequent sections.

Within *CHAR*, the enclosure radiation boundary condition is cast in the same form as a typical reradiation flux, given by

$$q_{rerad} = \sigma\epsilon\left(T_w^4 - T_\infty^4\right) \tag{19}$$

where ϵ is the emissivity of the solid surface. However, to incorporate the effects of reradiating faces, T_∞ is replaced by T_{eq}, resulting in

$$q_{rerad} = \sigma\epsilon\left(T_w^4 - T_{eq}^4\right) \tag{20}$$

The radiation exchange is computed from the temperature of the visible participating faces using the methods described in the following sections. By recasting (11) into the same form as (20), and assuming $\alpha_i^{IR} = \epsilon_i$ as stated by Kirchoff's law for a body in thermal equilibrium,

$$q_{rerad} = \sigma\epsilon_i\left(T_w^4 - \frac{1}{\sigma}G_i^{IR} - \frac{\alpha_i^{nonIR}}{\epsilon_i\sigma}G_i^{nonIR}\right) \tag{21}$$

where the equivalent farfield temperature is given by

$$T_{eq}^4 = \frac{1}{\sigma}G_i^{IR} + \frac{\alpha_i^{nonIR}}{\epsilon_i\sigma}G_i^{nonIR} \tag{22}$$

The T_{eq} for each participating boundary face is computed at the beginning of each time step. Accepting the boundary condition to be explicit in T_{eq} permits the FEM implementation of this term to be identical to that of *CHAR*'s reradiation boundary condition. Prior to the assembly of the FEM matrix, the full matrix of view factors is computed. Only boundary faces that are specified as having enclosure radiation are currently considered to participate in radiation exchange.

A. Radiosity Matrix Equation Approach

Substituting (16) into (12), and rearranging, the system of equations can be re-cast in matrix form as:

$$\begin{bmatrix} 1-\rho_1^{IR}F_{11} & -\rho_1^{IR}F_{12} & \cdots & -\rho_1^{IR}F_{1N} \\ -\rho_2^{IR}F_{21} & 1-\rho_2^{IR}F_{22} & \cdots & -\rho_2^{IR}F_{2N} \\ \vdots & \vdots & \ddots & \vdots \\ -\rho_N^{IR}F_{N1} & 1-\rho_N^{IR}F_{N2} & \cdots & 1-\rho_N^{IR}F_{NN} \end{bmatrix} \begin{bmatrix} J_1^{IR} \\ J_2^{IR} \\ \vdots \\ J_N^{IR} \end{bmatrix} = \begin{bmatrix} \epsilon_1\sigma T_1^4 + \rho_1^{IR}F_{1,space}\sigma T_\infty^4 \\ \epsilon_2\sigma T_2^4 + \rho_2^{IR}F_{2,space}\sigma T_\infty^4 \\ \vdots \\ \epsilon_N\sigma T_N^4 + \rho_N^{IR}F_{N,space}\sigma T_\infty^4 \end{bmatrix} \tag{23}$$

Solving for the radiosities, the incident radiation at each face can then be computed using (16). Similarly, the nonIR radiosities can be found by substituting (17) into (18) and recasting in a similar fashion,

$$\begin{bmatrix} 1-\rho_1^{nonIR}F_{11} & -\rho_1^{nonIR}F_{12} & \cdots & -\rho_1^{nonIR}F_{1N} \\ -\rho_2^{nonIR}F_{21} & 1-\rho_2^{nonIR}F_{22} & \cdots & -\rho_2^{nonIR}F_{2N} \\ \vdots & \vdots & \ddots & \vdots \\ -\rho_N^{nonIR}F_{N1} & 1-\rho_N^{nonIR}F_{N2} & \cdots & 1-\rho_N^{nonIR}F_{NN} \end{bmatrix} \begin{bmatrix} J_1^{nonIR} \\ J_2^{nonIR} \\ \vdots \\ J_N^{nonIR} \end{bmatrix} = \begin{bmatrix} \rho_1^{nonIR}F_{1,space}q_\infty^{nonIR} \\ \rho_2^{nonIR}F_{2,space}q_\infty^{nonIR} \\ \vdots \\ \rho_N^{nonIR}F_{N,space}q_\infty^{nonIR} \end{bmatrix} \tag{24}$$

Currently, *CHAR* uses the SuperLU_Dist package through PETSc to solve each matrix system by using a parallel direct solve. Since the problem typically only involves faces in the boundary where the enclosure radiation is being considered, the size of the matrix is manageable for most problems. Also, the direct solver results in the most accurate and robust solution to these matrix systems. However, options to allow the user to select an iterative solver (such as GMRES, conjugate gradient method, etc.) will be implemented in the near future to allow for more flexibility.

B. Sequential Reflections Approach

An alternate iterative approach has been implemented in *CHAR* to directly solve for the incident radiation. Substituting the IR radiosity ((12)) into the incident IR radiation term ((16)) a few times yields:

$$
\begin{aligned}
G_i^{IR} &= F_{i,space}\sigma T_\infty^4 + \sum_j F_{ij} J_j^{IR} \\
G_i^{IR} &= F_{i,space}\sigma T_\infty^4 + \sum_j F_{ij}\left(E_j + \rho_j^{IR} G_j^{IR}\right) \\
G_i^{IR} &= F_{i,space}\sigma T_\infty^4 + \sum_j F_{ij}E_j + \sum_j F_{ij}\rho_j^{IR}\left(\sum_k F_{jk} J_k^{IR}\right) \\
G_i^{IR} &= F_{i,space}\sigma T_\infty^4 + \sum_j F_{ij}E_j + \sum_j F_{ij}\rho_j^{IR}\sum_k F_{jk}\left(E_k + \rho_k^{IR} G_k^{IR}\right) \\
G_i^{IR} &= F_{i,space}\sigma T_\infty^4 + \sum_j F_{ij}E_j + \sum_j F_{ij}\rho_j^{IR}\sum_k F_{jk}E_k + \sum_j F_{ij}\rho_j^{IR}\sum_k F_{jk}\rho_k^{IR}\left(\sum_m F_{km} J_m^{IR}\right) \\
G_i^{IR} &= F_{i,space}\sigma T_\infty^4 + \sum_j F_{ij}E_j + \sum_j F_{ij}\rho_j^{IR}\sum_k F_{jk}E_k + \sum_j F_{ij}\rho_j^{IR}\sum_k F_{jk}\rho_k^{IR}\sum_m F_{km}\left(E_m + \rho_m^{IR} G_m^{IR}\right)
\end{aligned}
$$

$$
\overbrace{G_i^{IR}}^{\text{incident radiation}} = \overbrace{F_{i,space}\sigma T_\infty^4}^{\text{farfield contribution}} + \overbrace{\sum_j F_{ij}E_j}^{\text{direct emission}} + \overbrace{\sum_j F_{ij}\rho_j^{IR}\sum_k F_{jk}E_k}^{\text{first reflection}} + \overbrace{\sum_j F_{ij}\rho_j^{IR}\sum_k F_{jk}\rho_k^{IR}\sum_m F_{km}\left(E_m + \rho_m^{IR} G_m^{IR}\right)}^{\text{subsequent reflections}} \tag{25}
$$

It would be desirable to develop an iterative method to compute the total incident radiation including reflections. This is done by sequentially evaluating terms in this expression (where the number of terms is the number of reflections a particular emitted photon is allowed to experience before being ignored). With a few observations of the expression, this is feasible. Notice that each term contains something similar to

$$G_{e,i}^{0,IR} = \sum_j F_{ij}E_j, \tag{26}$$

which represents the source of the energy in the IR radiation field. This can be evaluated without regard for any reflections. Substituting this in, the overall expression reduces to:

$$G_i^{IR} = G_{e,i}^{0,IR} + \sum_j F_{ij}\rho_j^{IR} G_{e,j}^{0,IR} + \sum_j F_{ij}\rho_j^{IR}\sum_k F_{jk}\rho_k^{IR} G_{e,k}^{0,IR} + \text{remainder}. \tag{27}$$

The amount of incident radiation from the first reflection is

$$R_i^{0,IR} = \sum_j F_{ij}\rho_j^{IR} G_{e,j}^{0,IR} \tag{28}$$

and subsequent reflections are given by

$$R_i^{n,IR} = \sum_j F_{ij}\rho_j R_j^{(n-1),IR}. \tag{29}$$

Substituting in for the overall incident radiation:

$$G_i^{IR} = G_{e,i}^{0,IR} + R_i^{0,IR} + \sum_j F_{ij}\rho_j^{IR} R_j^{0,IR} + \text{remainder} \tag{30}$$

$$G_i^{IR} = G_{e,i}^{0,IR} + \sum_{n=0}^{N} R_i^{n,IR}. \tag{31}$$

Because ρ is always less than 1, the values for $R_i^{n,IR}$ will converge to zero. From this, it is apparent that by making multiple passes through the communicating faces, the total incident radiation for a given number of reflections can be built up based on surface temperatures and the incident radiation from the previous reflection number, and that this value will converge as more reflections are taken into account. A similar approach is used for the nonIR reflections, with a farfield contribution due to q_∞ instead of T_∞.

IV. Radiation View Factors

The geometric view factor, sometimes also known as a configuration factor, is purely a geometric quantity that describes the fraction of energy leaving a finite area A_1 being intercepted by area A_2. Mathematically, the exact definition of the view factor from element face i to element face j is given by

$$F_{ij} = \frac{1}{A_i} \int_{A_i} \int_{A_j} \frac{\cos\theta_i \cos\theta_j}{\pi S^2} dA_j dA_i, \tag{32}$$

where the terms are defined in Figure 1. Clearly, any view factor value should be between 0 and 1. In a complete enclosure, the sum of all the view factors from a given face i should add up to 1. That is,

$$\sum_{j=1}^{n} F_{ij} = 1 \tag{33}$$

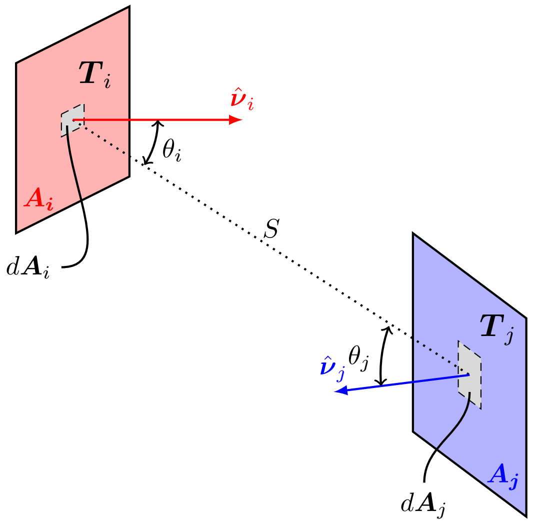


Figure 1. Definition of terms used in computing view factor from face i to face j, F_{ij}.

Similarly, in a partial enclosure, the sum of all the view factors from i to every face j, and the view factor that i has to the space external to the enclosure, must also add up to 1. In addition, the well-known reciprocity relationship

$$A_i F_{ij} = A_j F_{ji} \tag{34}$$

between two faces (i and j) must also be satisfied.

Several different approaches to compute view factors have been developed and can be found in the literature [References??]. For simple geometries, analytical formulas for view factors can be derived. Unfortunately, such closed form solutions do not exist for most practical problems of interest, and therefore, it is desired to have a robust and accurate manner to compute view factors for complex geometries. In addition, computational cost can also be an important consideration for problems with many faces, and/or problems in which view factors must be recomputed often. Two different methods for computing view factors have been implemented in *CHAR* to date, and they are presented in the following sections.

A. Double Area Integral Method

The first method implemented to calculate view factors is to numerically integrate Eq. (32) by dividing participating element faces into K sub-faces, where K is a user input. The integrals can now be expressed as summations.

$$F_{ij} = \frac{1}{A_i} \sum_{l=1}^{K} \sum_{m=1}^{K} \frac{\cos(\theta_i)_l \cos(\theta_j)_m}{\pi S^2_{l\to m}} (A_j)_m (A_i)_l \, b_{lm} \tag{35}$$

where the angles and areas are defined with respect to the sub-faces. The outer "l" summation is over sub-faces on element face i, and the inner "m" summation is over sub-faces on element face j. Consequently, the total areas of the element faces are given by

$$A_i = \sum_{l=1}^{K} (A_i)_l \tag{36}$$

and

$$A_j = \sum_{m=1}^{K} (A_j)_m \tag{37}$$

In *CHAR*, the user can optionally specify that a search be performed to verify that the $R_{l \to m}$ paths are not obstructed by any other face, also known as "shadowing". If an obstruction is found, the term b_{lm} is set to 0, and the view factor contribution from those sub-faces is therefore 0. Similarly, if two faces have normals that do not point toward each other, b_{lm} is also set to 0. If the view factors for face i do not sum to 1.0, the difference is made up by a view factor to a 'far-field' face with temperature T_∞ and/or a nonIR radiation source flux q_∞.

This method attempts to account for partially shadowed faces by checking the path between each possible pair of nodes between two sub-faces. However, the accuracy of the view factor between two partially shadowed faces increases as more sub-faces are used. The double area integral approach is most accurate when computing view factors for small faces which are far apart. As pointed out in [3], the method is not as accurate when considering two faces which share an edge, or faces which are close to each other. Still, this method can result in fast and accurate computation of view factors in *CHAR* when used appropriately.

Finally, the double area integral approach naturally respect the reciprocity relationship given by (34). However, integration errors introduced by the simplification of the view factor definition given by (35) can result in violation of the additive property given by (33).

B. Monte Carlo Ray Tracing Method

The second method available to compute view factors is a Monte Carlo ray-tracing algorithm. This approach has been studied extensively for computing view factors and for computer graphics applications. To compute view factors, a large number of statistical experiments are carried out from any given face by randomly selecting the origin of a ray (or photon). Once a total number of rays N are casted from an element face i, the view factor to any other face j is given by

$$F_{ij} = \frac{m}{N} \tag{38}$$

where m is the total number of rays emitted from face i which intersected face j.

CHAR supports view factor computation on quadrilaterial or triangular faces in 3D problems. However, for the purposes of selecting a random origin of each ray to be emitted, quadrilaterals are split into 2 triangular faces by randomly selecting the diagonal to split along. Then, the origin of the ray to be emitted on the selected triangular face is given by [4]

$$\boldsymbol{P_{origin}} = (1 - \sqrt{r_1})\mathbf{n_1} + \sqrt{r_1}(1 - r_2)\mathbf{n_2} + \sqrt{r_1} r_2 \mathbf{n_3} \tag{39}$$

where $\mathbf{n_1}$, $\mathbf{n_2}$ and $\mathbf{n_3}$ are the vertices of the selected triangle, and r_1 and r_2 are two random numbers generated.

To initialize the pseudo-random number generator used in *CHAR*, each processor gets assigned a unique seed according to [5]

$$seed = |((t \times 181)((pid - 83) \times 359))\%104729| \tag{40}$$

where t is the CPU time in seconds obtained with the C++ *time()* command, and pid is the processor's ID. This seeding scheme results in one seed collision every 10,000 job submissions.

By treating each face as a diffuse emitter, each ray emission direction is also randomly selected according to a cosine probability distribution as governed by Lambert's law. In 3D geometries, where the radiating faces are 2D face elements, the warping function used to generate the direction of the ray emitted is given by [6]

$$(\chi, \psi) = (\cos^{-1}(\sqrt{1 - r_3}), 2\pi r_4) \tag{41}$$

where χ and ψ represent the polar angle with respect to the face normal, and the azimuthal angle around the normal of the element, respectively. In 2D problems, where the radiating face elements are edges, the angle of the ray relative to the face normal is given by,

$$\chi = \sin^{-1}(r_5) \tag{42}$$

where r_3, r_4 and r_5 are uniformly distributed random numbers between [0,1].

The Monte Carlo approach is especially powerful in dealing with complex geometries as it implicitly takes into account shadowing and partial shadowing. However, due to the non-deterministic nature of this approach, the analyst must ensure sufficiently many rays are cast from each face to correctly resolve the view factors. It is also well known that according to probability theory, the error in this Monte Carlo approach is inversely proportional to the square root of number of rays emitted for a given confidence interval and view factor,

$$E = \sqrt{2}\mathrm{erf}^{-1}(C)\sqrt{\frac{1 - F_{ij}}{NF_{ij}}}, \tag{43}$$

where C is the confidence level, and N is the number of rays emitted. Therefore, to decrease the error by a factor of 2, the number of rays emitted must be increased by a factor of 4. This highlights the large computational expense which typically comes with this Monte Carlo approach to compute view factors. Still, (43) provides a way to estimate the number of rays required to resolve a given view factor down to a desired level of error. *CHAR* provides the user an option to use (43) to determine the number of rays to be emitted by specifying a confidence level, a maximum error, and a minimum view factor to resolve.

This monte carlo approach for computing view factors inherently respects the view factor sum property given by (33). However, due to its stochastic nature, it does not satisfy the reciprocity stated by (34). Approaches to smooth view factors to better satisfy both of these properties have been developed in the literature. For example, Daun, Morton and Howell [7] show a smoothing approach using a constrained maximum likelihood estimation algorithm. However, no view factor smoothing has been implemented in *CHAR* at this time. Currently, *CHAR* supports view factor computation in 2D and 3D problems using this Monte Carlo approach.

V. Efficient Octree Traversal

To alleviate the computational cost of the many ray-intersections and searches involved with the Monte Carlo view factor approach, an octree structure and efficient octree traversal algorithm have been implemented in *CHAR*. In an "exhaustive search" implementation, the entire list of boundary faces must be searched, which results in enormous computational expense. This section presents a very efficient alternative based on quad- and octrees that reduces the runtime *significantly*, and makes running of problems with a large number of radiating faces and/or moving mesh feasible. The section is composed of three groups of subsections:

1. **Building the Tree (subsections A-D)**

   An octree works by recursively partitioning a volume of three-space into successively smaller and nested regions and ("boxes") associating a set of data with each of those subregions. It is the three-dimensional analog to a binary search tree. In this application, the data is a collection of element faces. We place a face inside a box if the face intersects the box nontrivially (i.e., non-zero volume). It is possible (and likely) for a face to belong to more than one box. The text of these sections focuses on the development of an efficient intersection test based on the separating axis theorem from convex analysis.

2. **The Traversal Algorithm (subsections (E-J)**

   A ray is shot from one of the faces. The next step is to determine which of the tree boxes the ray pieces along its forward trajectory and search only the contents of those boxes for the closest (if any) face of intersection. The tree is used to identify those boxes and traversal follows a top-down approach: every face is part of the largest box that bounds the entire domain. From there, we determine to which of the children (subboxes) the ray enters. If a child box has no children of its own, then we include its contents among what we will eventually search. Otherwise, the process continues recursively and treats each child box as though it were the mother box. These sections detail the means by which we do this, a clever scheme based on entry and exit times due to [8].

3. **Performance (subsection K)**

   We conclude by presenting several viewfactor-only problems that highlight the immense gains realized by ultilization of an octree search over an exhaustive search. A single 2D problem shows that a quadtree does yield notheworthy, if not outstanding, speed-ups of 2-3x. For 3D problems, the performance gain increases with the size of the problem. For modest problems, we see speed-up of 5-10x. For a medium size problems, the factor reaches up over 20x. Finally, for the largest problems, the speedometer cruises up over a whopping 100x. Such impressive speed increases reduce hardware requirements and allow local execution of problems that would otherwise require massively-parallel network clusters to complete in an acceptable time period.

A. The Separating Axis Test (SAT)

The following fundamental result from convex analysis is of the utmost use to use:

Theorem 1 (Hyperplane Separation Theorem)

Let $\mathcal{A} \subset \mathbb{R}^d$ be closed and $\mathcal{K} \subset \mathbb{R}^d$ be compact with both convex. Then $\mathcal{A} \cap \mathcal{K} = \emptyset$ if and only if there exists a hyperplane $\mathcal{P} = \left\{\mathbf{x} \in \mathbb{R}^d : \mathbf{x} \cdot \mathbf{p} = c\right\}$ for some $c \in \mathbb{R}$ and $\mathbf{p} \in \mathbb{R}^d \backslash \{\mathbf{0}\}$ such that either

1. $\mathbf{p} \cdot \mathbf{a} > c$ for all $\mathbf{p} \in \mathcal{P}$, $\mathbf{a} \in \mathcal{A}$ and $\mathbf{p} \cdot \mathbf{k} < c$ for all $\mathbf{p} \in \mathcal{P}$, $\mathbf{k} \in \mathcal{K}$
2. $\mathbf{p} \cdot \mathbf{a} < c$ for all $\mathbf{p} \in \mathcal{P}$, $\mathbf{a} \in \mathcal{A}$ and $\mathbf{p} \cdot \mathbf{k} > c$ for all $\mathbf{p} \in \mathcal{P}$, $\mathbf{k} \in \mathcal{K}$

In plain language, $\mathcal{A}$ is on one side of $\mathcal{A}$ and $\mathcal{K}$ is on the other and there is a gap between them. Such a $\mathcal{P}$ is called a *separating hyperplane*.

Rather than attempt to compute a point of intersection directly—we only care *if* there is intersection, not where—we may use the Hyperplane Separation Theorem to check for the existence of a separating hyperplane. If none can be found, then we may conclude intersection. The test for the hyperplane is expensive, so it is fortunate that the existence of a separating hyperplane is equivalent to the existence of a separating axis:

Definition 1 (Separating Axis)

Let $\mathcal{P} \in \mathbb{R}^d$ be a separating hyperplane. $\boldsymbol{\xi} \subset \mathbb{R}^d$ is a *separating axis* if $\boldsymbol{\xi} \perp \mathcal{P}$.

Because a hyperplane is always of dimension $d-1$, $\dim \boldsymbol{\xi} = 1$ whenever $\boldsymbol{\xi}$ is a separating axis; we may identify any separating axis with a spanning element in $\mathbb{R}^d$. It is so named because the orthogonal projections of the basis vectors of $\mathcal{A}$ and $\mathcal{K}$ onto $\boldsymbol{\xi}$ will result in non-overlapping intervals. This leads to an obvious result:

Theorem 2

Let $\mathcal{A} \subset \mathbb{R}^d$ be closed and $\mathcal{K} \subset \mathbb{R}^d$ be compact with both convex. Then there exists a separating hyperplane for $\mathcal{A}$ and $\mathcal{K}$ if and only if there exists a separating axis between them.

It is worth noting that some of the objects we are interested in testing for intersection—boxes and segments in particular—are bounded with a centroid about which they are symmetric. We can therefore devise the following simple and efficient check, dubbed the separating axis test (SAT), for a candidate $\boldsymbol{\xi}$:

Algorithm 1 (Separating Axis Test for Symmetric Objects)

1. Let $\mathbf{c} \in \mathcal{K}$ and $\mathbf{c}' \in \mathcal{K}'$ be the centroids of $\mathcal{K}$ and $\mathcal{K}'$.
2. Let $\Pi_{\boldsymbol{\xi}}(\mathbf{x})$ denote the projection of $\mathbf{x} \in \mathbb{R}^d$ onto $\boldsymbol{\xi}$. Define

$$\ell \triangleq \max_{0 \le i \le \bar{N}-1} |\Pi_{\boldsymbol{\xi}}(\mathbf{n}_i - \mathbf{c})| \qquad \ell' \triangleq \max_{0 \le i \le \bar{N}-1} |\Pi_{\boldsymbol{\xi}}(\mathbf{n}'_i - \mathbf{c}')| \qquad L \triangleq |\Pi_{\boldsymbol{\xi}}(\mathbf{c} - \mathbf{c}')|$$

3. If $\ell + \ell' < L$, then $\boldsymbol{\xi}$ is a separating axis.

In plain language, the symmetric SAT computes the maximum length of the projections of the centroid–node radii onto $\boldsymbol{\xi}$. The sum of these values is then compared to the distance between the projections of the centroids onto $\boldsymbol{\xi}$. If $\ell + \ell' < L$, then the sets must be disjoint. The process is illustrated in Figure 2 below.

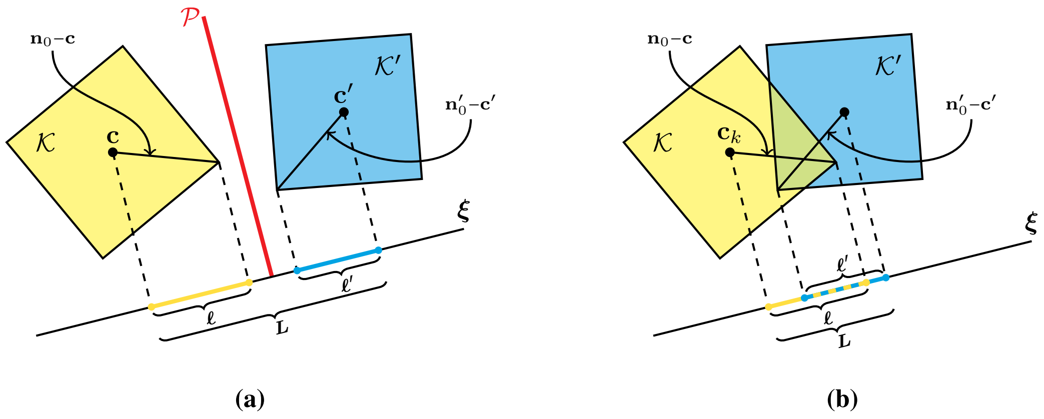


Figure 2. Separating Axis Test in Action. In (a), the objects $\mathcal{K}$ and $\mathcal{K}'$ are separated by the hyperplane $\mathcal{P}$ (not unique). Here, we see that the intervals $\left[\Pi_{\boldsymbol{\xi}}(\mathbf{c}), \Pi_{\boldsymbol{\xi}}(\mathbf{n}_0)\right]$ (yellow) and $\left[\Pi_{\boldsymbol{\xi}}(\mathbf{n}_0'), \Pi_{\boldsymbol{\xi}}(\mathbf{c}')\right]$ (blue) are disjoint; this is equivalent to $\ell + \ell' < L$. If $\mathcal{K}$ and $\mathcal{K}'$ touch at a single point, then $\ell + \ell' = L$. In (b), we have intersection so $\boldsymbol{\xi}$ is not a separating axis. We see that the overlap causes $\ell + \ell' > L$.

When the objects are not both symmetric, the picture changes: we cannot project node–centroid radii because the object's projection will no longer be symmetric about the projected centroid. Instead, we must project each node of the object onto $\boldsymbol{\xi}$ and then compute the maximum distance among these projections. In theory, we must also project the convex hull onto $\boldsymbol{\xi}$, although in practice we can avoid this.

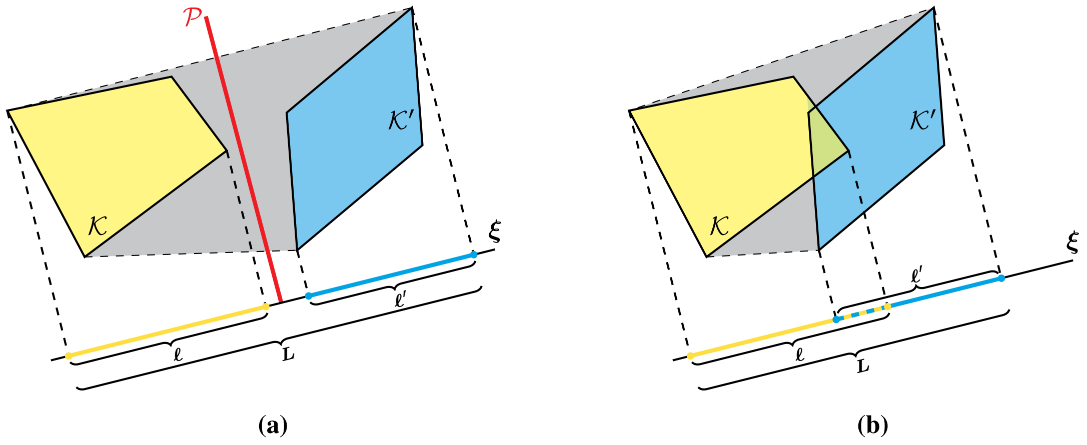


Figure 3. Separating Axis Test in Action, Non-Symmetric Objects. Here, ℓ is the length of the projection of $\mathcal{K}$ onto $\boldsymbol{\xi}$, similarly for ℓ'. The convex hull of $\mathcal{K}$ and $\mathcal{K}'$ is shown in gray; the length of its projection onto $\boldsymbol{\xi}$ is L. In (a), we see a separating hyperplane $\mathcal{P}$ because $\ell + \ell' < L$. In (b), we have intersection so $\boldsymbol{\xi}$ is not a separating axis. We see that the projections overlap and cause $\ell + \ell' > L$.

In the test for symmetric objects, the point $\Pi_{\boldsymbol{\xi}}(\mathbf{c})$ is implicitly chosen to be the zero coordinate on the $\boldsymbol{\xi}$ axis. There is no such obvious candidate for the non-symmetric test. However, this does not matter: we need only compute the largest distance between pairs of projected nodes. Because $\max_{x \in I}(a + x) = a + \max_{x \in I} x$, the zero point will ultimately subtract out and is accordingly not relevant. Because the objects are convex, the extreme points of the convex hull are nodes in one of the objects; we can bypass computing the convex hull by locating the maximum and minimum points among each subset of projections.

Instead, we can compute the lengths of the projections directly by projecting each node onto $\boldsymbol{\xi}$ and then finding the length of the interval formed by the collection. Algorithm 2 below explicates the technique:

Algorithm 2 (Separating Axis Test for Non-Symmetric Objects)

1. Let $\{\mathbf{n}_i\}_{i=0}^{\bar{N}-1}$ and $\{\mathbf{n}'_i\}_{i=0}^{\bar{N}'-1}$ be the nodes of $\mathcal{K}$ and $\mathcal{K}'$, respectively. Compute
$$p_i \triangleq |\Pi_{\boldsymbol{\xi}}(\mathbf{n}_i)| \qquad p'_i \triangleq |\Pi_{\boldsymbol{\xi}}(\mathbf{n}'_i)|$$
2. Compute the lengths of the intervals of the projection in $\boldsymbol{\xi}$ coordinate space:
$$\begin{aligned} \ell &\triangleq \max_{0\le i\le \bar{N}} p_i - \min_{0\le i\le \bar{N}} p_i \\ \ell' &\triangleq \max_{0\le i\le \bar{N}} p'_i - \min_{0\le i\le \bar{N}} p'_i \\ L &\triangleq \max\Big\{ \max_{0\le i\le \bar{N}} p_i, \max_{0\le i\le \bar{N}} p'_i\Big\} - \min\Big\{ \min_{0\le i\le \bar{N}} p_i, \min_{0\le i\le \bar{N}} p'_i\Big\} \end{aligned}$$
3. If $\ell + \ell' < L$, then $\boldsymbol{\xi}$ is a separating axis.

There are six ways in which intersection can occur: face–face, face–edge, face–node, edge–edge, edge–node, or node–node. While covered by other tests (because a node is on a face or edge), node intersections are handled first for a quick check: if any nodes are contained in the box, then we can skip the remaining checks.

While it is easy for a human to draw a separating hyperplane by inspection, automating the process requires identifying a (preferably small) finite number of possible separating axes. An obvious set of candidates are given by the faces of the objects $\mathcal{K}$ and $\mathcal{K}'$. By continuity of the distance metric $\rho : \mathbb{R}^d \times \mathbb{R}^d \to \mathbb{R}^+$, if $\rho(\mathcal{K}, \mathcal{K}') > 0$, then we can perturb one of $\mathcal{K}$ or $\mathcal{K}'$ by $\varepsilon \ll 1$ and maintain a positive distance between the perturbed set and the original other. The separating axes corresponding to the faces are the face normals.

In 2D, the collection of face normals is sufficient; in 3D, however, testing only face normals can result in false detection. The figure below displays the potential for failure.

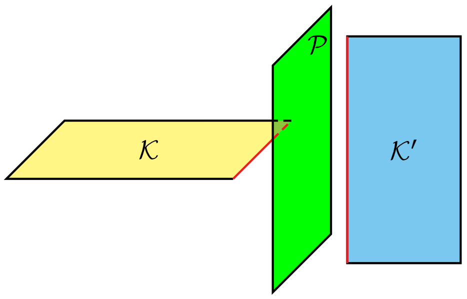


Figure 4. Insufficiency of Face Normals. Here, the face normal of $\mathcal{K}$ lies in the same plane as $\mathcal{K}'$ and the face normal of $\mathcal{K}'$ lies in the same plane as $\mathcal{K}$. Hence, neither is a separating axis for these objects. The separating hyperplane $\mathcal{P}$ is spanned by one edge from each of $\mathcal{K}$ and $\mathcal{K}'$, indicated in red.

While the figure above is drawn for two quadrilaterals, it is precisely the situation that can be encountered in edge–edge intersections. If we tested only face normals for the above figure, we'd find no separating axis between them and would erroneously conclude intersection. In this case, the hyperplane is spanned by two edges. Since the separating axis is orthogonal to the hyperplane, a third class of candidate separating axes is generated by the set of cross products of each edge from one object with each edge from the other.

To summarize the above paragraphs, the following are the candidate separating axes:

1. The face normals of $\mathcal{K}$.
2. The face normals of $\mathcal{K}'$.
3. The cross product of an edge from $\mathcal{K}$ with an edge from $\mathcal{K}'$ (all possible combinations, **3D only**).

The above list seemingly forms a large set of possibilities to test. Because the SAT considers only projections, the number of tests required can be greatly reduced by discarding parallel axes. For example, for a triangle and a bounding box, a naïve compilation of the above list results in twenty-five axes to test: six from normals of the box, one from the triangle normal, and eighteen from edges. However, normals from opposing faces of the box are (anti)parallel and only three edges have unique direction. This analysis reduces the number of required tests to just thirteen (four from normals, nine from pairing edges).

The discussion of this section is based on the code, algorithms, and explanations given in [9], although additional detail has been provided here and the results have been specialized to the specific cases and geometries relevant to viewfactor calculations.

B. Executing the SAT in 2D

In 2D, faces and edges coincide. Per the preceding section, the candidates for separating axes are the normals from the sides of the box $\mathcal{B}$ and normal from the "face." Because tree boxes are always axis-aligned, the normals are the coordinate axes $\boldsymbol{\xi}_1 = (1,0)^T$ and $\boldsymbol{\xi}_2 = (0,1)^T$. The edge vector is given by the directed segment from $\mathbf{n}_0$ to $\mathbf{n}_1$, or

$$\mathbf{w} = \frac{1}{2}(\mathbf{n}_1 - \mathbf{n}_0)$$

The factor of one-half is merely for convenience. The third candidate is the perpendicular to $\mathbf{w}$, or $\boldsymbol{\xi}_3 = \left(-e^{(1)}, e^{(0)}\right)^T$. Because the box and segment are symmetric, we use Algorithm 1. Letting $\mathcal{K} = \mathcal{B}$, where $\mathcal{B}$ denotes a box,

$$\mathbf{d} \triangleq \mathbf{n}_0 - \mathbf{c} = \frac{1}{2}(\mathbf{x}_{\max} - \mathbf{x}_{\min})$$

which is half of the box diagonal. The final piece is then the centroid-to-centroid vector

$$\mathbf{m} \triangleq \mathbf{c} - \mathbf{c}' = \frac{1}{2}(\mathbf{n}_0 + \mathbf{n}_1) - \mathbf{x}_{\text{mid}}$$

where $\mathbf{x}_{\text{mid}} = \frac{1}{2}\left(\mathbf{x}_{\max} + \mathbf{x}_{\min}\right)$ is the centroid of the box. The projection operator Π onto $\boldsymbol{\xi}$ is given by

$$\Pi_{\boldsymbol{\xi}}(\mathbf{x}) = \frac{\boldsymbol{\xi}\cdot\mathbf{x}}{|\boldsymbol{\xi}|}\boldsymbol{\xi} \tag{44}$$

When the norm of the projections is taken, the $|\boldsymbol{\xi}|$ will cancel. Based on the above, we have

Axis	$\boldsymbol{\xi}$	$\boldsymbol{\ell}$	$\boldsymbol{\ell}'$	$\boldsymbol{L}$
1	$(1,0)^T$	$\lvert d^{(0)}\rvert$	$\lvert w^{(0)}\rvert$	$\lvert m^{(0)}\rvert$
2	$(1,0)^T$	$\lvert d^{(1)}\rvert$	$\lvert w^{(1)}\rvert$	$\lvert m^{(1)}\rvert$
3	$\left(-w^{(1)}, w^{(0)}\right)^T$	$\lvert d^{(0)}w^{(1)}\rvert$	$\lvert d^{(1)}w^{(0)}\rvert$	$\lvert w^{(0)}m^{(1)} - w^{(1)}m^{(0)}\rvert$

Table 2. Vital Information for SAT (2D). This table contains all of the quantities needed to formulate an algorithm for intersection of faces and boxes in 2D.

Thanks to the definition of $\mathbf{e}$, the quantities ℓ, ℓ', and L each contain a factor of $\frac{1}{2}$ that may be removed. In addition, because $d^{(i)} > 0$ for every i, we remove the absolute values from these quantities. With these considerations and Table 2, we may present a streamlined 2D insertion algorithm:

Algorithm 3 (Insertion in 2D)

Let $\mathbf{n}_0$ and $\mathbf{n}_1$ be the nodes that comprise the edge (face) to be inserted.

1. If either $\mathbf{n}_0 \in \mathcal{B}$ or $\mathbf{n}_1 \in \mathcal{B}$, add the face to the box and exit.
2. Otherwise, compute the following quantities:

$\mathbf{w} \triangleq \mathbf{n}_1 - \mathbf{n}_0$	Edge vector
$\mathbf{d} \triangleq \mathbf{x}_{\max} - \mathbf{x}_{\min}$	Box diagonal
$\mathbf{m} \triangleq \mathbf{n}_0 + \mathbf{n}_1 - \mathbf{x}_{\max} - \mathbf{x}_{\min}$	Segment midpoint to box centroid

3. Axis Test #1: if $\left|m^{(0)}\right| > d^{(0)} + \left|w^{(0)}\right|$, exit.
4. Axis Test #2: if $\left|m^{(1)}\right| > d^{(1)} + \left|w^{(1)}\right|$, exit.
5. Axis Test #3: if $\left|w^{(0)}m^{(1)} - w^{(1)}m^{(0)}\right| > \left|d^{(0)}w^{(1)}\right| + \left|d^{(1)}w^{(0)}\right|$ exit.
6. Reaching this point means no separating axis exists: add the face to the box.

C. Executing the SAT in 3D

A face $\mathcal{F}$ in 3D can be either a triangle or a quadrilateral. Let $\bar{N}$ be the number of nodes on the face and $\left\{\mathbf{n}_i\right\}_{i=0}^{\bar{N}-1}$ be the set of nodes; for a triangle, $\bar{N} = 3$ and for a quadrilateral, $\bar{N} = 4$. In general, $\mathcal{F}$ is not symmetric, so we must use Algorithm 2. Fortunately, the box *is* symmetric, so we can compute ℓ directly via Algorithm 1.

Per Algorithm 2, letting $\mathcal{K} = \mathcal{B}$, we must compute

$$\ell = \max_i(\mathbf{v}_i \cdot \boldsymbol{\xi}) - \min_i(\mathbf{v}_i \cdot \boldsymbol{\xi})$$

The nodes $\mathbf{v}_i$ of the box have jth component either $x_{\max}^{(j)}$ or $x_{\min}^{(j)}$. To maximize the first term in ℓ, we should choose the node $\mathbf{v}_i$ such that

$$v_i^{(j)} = \begin{cases} x_{\max}^{(j)} & \text{if } \xi^{(j)} \geq 0 \\ x_{\min}^{(j)} & \text{if } \xi^{(j)} < 0 \end{cases}$$

To minimize the second term, we should choose the node $\mathbf{n}_i$ such that

$$v_i^{(j)} = \begin{cases} x_{\min}^{(j)} & \text{if } \xi^{(j)} \geq 0 \\ x_{\max}^{(j)} & \text{if } \xi^{(j)} < 0 \end{cases}$$

As an example, suppose that the signs of $\boldsymbol{\xi}$ are $+, -, -$. Then per the above,

$$\max_i(\mathbf{v}_i \cdot \boldsymbol{\xi}) = x_{\max}^{(0)}\left|\xi^{(0)}\right| - x_{\min}^{(0)}\left|\xi^{(1)}\right| - x_{\min}^{(2)}\left|\xi^{(2)}\right| \tag{45}$$

$$-\min_i(\mathbf{v}_i \cdot \boldsymbol{\xi}) = x_{\min}^{(0)}\left|\xi^{(0)}\right| - x_{\max}^{(1)}\left|\xi^{(1)}\right| - x_{\max}^{(2)}\left|\xi^{(2)}\right| \tag{46}$$

While the above combine to give a nice expression involving the diagonal $\mathbf{d} = \mathbf{x}_{\max} - \mathbf{x}_{\min}$, it is preferable to leave them separate because both quantities are needed individually to compute L.

The face normals for the box are again the coordinate axes: $\boldsymbol{\xi}_1 = (1,0,0)^T$, $\boldsymbol{\xi}_2 = (0,1,0)^T$, $\boldsymbol{\xi}_3 = (0,0,1)^T$. We may compute the necessary dot products for ℓ, ℓ, and L explicitly for these axes with little effort. Let $\boldsymbol{\xi}_4 = \boldsymbol{\nu}$ be the normal of $\mathcal{F}$. We can appeal to Eq. (45) and Eq. (46) for ℓ, but because we know nothing about the components of $\boldsymbol{\nu}$, there is nothing we can do to simplify $\boldsymbol{\nu} \cdot \mathbf{n}_i$. This gives the table

Axis	$\boldsymbol{\xi}$	$\boldsymbol{\ell}$	$\boldsymbol{\ell'}$	$\boldsymbol{L}$
1	$(1,0,0)^T$	$d^{(0)}$	$\max n_i^{(0)} - \min n_i^{(0)}$	$\max\left\{x_{\max}^{(0)}, \max n_i^{(0)}\right\} - \min\left\{x_{\min}^{(0)}, \min n_i^{(0)}\right\}$
2	$(0,1,0)^T$	$d^{(1)}$	$\max n_i^{(1)} - \min n_i^{(1)}$	$\max\left\{x_{\max}^{(1)}, \max n_i^{(1)}\right\} - \min\left\{x_{\min}^{(1)}, \min n_i^{(1)}\right\}$
3	$(0,0,1)^T$	$d^{(2)}$	$\max n_i^{(2)} - \min n_i^{(2)}$	$\max\left\{x_{\max}^{(2)}, \max n_i^{(2)}\right\} - \min\left\{x_{\min}^{(2)}, \min n_i^{(2)}\right\}$
4	$\boldsymbol{\nu}$	$M - m$	$\max \boldsymbol{\nu}\cdot\mathbf{n}_i - \min \boldsymbol{\nu}\cdot\mathbf{n}_i$	$\max\left\{M, \max \boldsymbol{\nu}\cdot\mathbf{n}_i\right\} - \min\left\{m, \min \boldsymbol{\nu}\cdot\mathbf{n}_i\right\}$

Table 3. Vital Information for SAT (3D), Part I. This table contains all of the quantities for testing separating axes corresponding to face normals from the box and face in 3D.

Appearing in the final row of Table 3, the values M and m are defined as

$$M \triangleq \nu^{(0)} \cdot \begin{cases} x_{\max}^{(0)} & \text{if } \nu^{(0)} \geq 0 \\ x_{\min}^{(0)} & \text{if } \nu^{(0)} < 0 \end{cases} + \nu^{(1)} \cdot \begin{cases} x_{\max}^{(1)} & \text{if } \nu^{(1)} \geq 0 \\ x_{\min}^{(1)} & \text{if } \nu^{(1)} < 0 \end{cases} + \nu^{(2)} \cdot \begin{cases} x_{\max}^{(2)} & \text{if } \nu^{(2)} \geq 0 \\ x_{\min}^{(2)} & \text{if } \nu^{(2)} < 0 \end{cases} \tag{47}$$

$$m \triangleq \nu^{(0)} \cdot \begin{cases} x_{\min}^{(0)} & \text{if } \nu^{(0)} \geq 0 \\ x_{\max}^{(0)} & \text{if } \nu^{(0)} < 0 \end{cases} + \nu^{(1)} \cdot \begin{cases} x_{\min}^{(1)} & \text{if } \nu^{(1)} \geq 0 \\ x_{\max}^{(1)} & \text{if } \nu^{(1)} < 0 \end{cases} + \nu^{(2)} \cdot \begin{cases} x_{\min}^{(2)} & \text{if } \nu^{(2)} \geq 0 \\ x_{\max}^{(2)} & \text{if } \nu^{(2)} < 0 \end{cases} \tag{48}$$

and are the results of explicitly computing $\max_i (\boldsymbol{\nu}\cdot\mathbf{v}_i)$ and $\min_i (\boldsymbol{\nu}\cdot\mathbf{v}_i)$, respectively, as in Eq. (45) and Eq. (46).

In 2D, no edge–edge intersections must be tested and normals alone suffice as candidate separating axes. We are not so fortunate in 3D. An edge $\mathbf{f}_i$, $0 \leq i \leq \bar{N} - 1$, of $\mathcal{F}$ is given in terms of its nodes:

$$\mathbf{f}_i \triangleq \mathbf{n}_{i+1} - \mathbf{n}_i$$

The $i + 1$ should be interpreted in terms of mod $N + 1$ so that the index does not go out of range when $i = \bar{N} - 1$; this ensures that the final edge $\mathbf{n}_{\bar{N}-1} \to \mathbf{n}_0$ is included. The edges of the box are again the coordinate axes: $\mathbf{b}_1 = (1,0,0)^T$, and $\mathbf{b}_2 = (0,1,0)^T$, and $\mathbf{b}_3 = (0,0,1)^T$. The candidate axes generated from an edge $\mathbf{f}_i$ are the cross products with each of the $\mathbf{b}_{i'}$, which we may compute explicitly:

$$\begin{aligned}
\boldsymbol{\xi}_{5,i} &= \mathbf{b}_1 \times \mathbf{f}_i = \left(0, -f_i^{(2)}, f_i^{(1)}\right)^T = \left(0, n_i^{(2)} - n_{i+1}^{(2)}, n_{i+1}^{(1)} - n_i^{(1)}\right)^T \\
\boldsymbol{\xi}_{6,i} &= \mathbf{b}_2 \times \mathbf{f}_i = \left(f_i^{(2)}, 0, -f_i^{(0)}\right)^T = \left(n_{i+1}^{(2)} - n_i^{(2)}, 0, n_i^{(0)} - n_{i+1}^{(0)}\right)^T \\
\boldsymbol{\xi}_{7,i} &= \mathbf{b}_3 \times \mathbf{f}_i = \left(-f_i^{(1)}, f_i^{(0)}, 0\right)^T = \left(n_i^{(1)} - n_{i+1}^{(1)}, n_{i+1}^{(0)} - n_i^{(0)}, 0\right)^T
\end{aligned} \tag{49}$$

For purposes of derivation, we may suppose that $\bar{N} = 4$; if $\bar{N} = 3$, the only other valid option, then expressions involving the final node $\mathbf{n}_3$ may be ignored. Using the rightmost side of the above, we can reduce the computational burden required to find ℓ'. We compute each of the inner products:

$$\left.\begin{aligned}
\alpha_{0,i} &\triangleq \mathbf{n}_0 \cdot \boldsymbol{\xi}_{5,i} = n_0^{(1)}\left(n_i^{(2)} - n_{i+1}^{(2)}\right) + n_0^{(2)}\left(n_{i+1}^{(1)} - n_i^{(1)}\right) \\
\alpha_{1,i} &\triangleq \mathbf{n}_1 \cdot \boldsymbol{\xi}_{5,i} = n_1^{(1)}\left(n_i^{(2)} - n_{i+1}^{(2)}\right) + n_1^{(2)}\left(n_{i+1}^{(1)} - n_i^{(1)}\right) \\
\alpha_{2,i} &\triangleq \mathbf{n}_2 \cdot \boldsymbol{\xi}_{5,i} = n_2^{(1)}\left(n_i^{(2)} - n_{i+1}^{(2)}\right) + n_2^{(2)}\left(n_{i+1}^{(1)} - n_i^{(1)}\right) \\
\alpha_{3,i} &\triangleq \mathbf{n}_3 \cdot \boldsymbol{\xi}_{5,i} = n_3^{(1)}\left(n_i^{(2)} - n_{i+1}^{(2)}\right) + n_3^{(2)}\left(n_{i+1}^{(1)} - n_i^{(1)}\right)
\end{aligned}\right\} \tag{50}$$

$$\left.\begin{aligned}
\beta_{0,i} &\triangleq \mathbf{n}_0 \cdot \boldsymbol{\xi}_{6,i} = n_0^{(0)}\left(n_{i+1}^{(2)} - n_i^{(2)}\right) + n_0^{(2)}\left(n_i^{(0)} - n_{i+1}^{(0)}\right) \\
\beta_{1,i} &\triangleq \mathbf{n}_1 \cdot \boldsymbol{\xi}_{6,i} = n_1^{(0)}\left(n_{i+1}^{(2)} - n_i^{(2)}\right) + n_1^{(2)}\left(n_i^{(0)} - n_{i+1}^{(0)}\right) \\
\beta_{2,i} &\triangleq \mathbf{n}_2 \cdot \boldsymbol{\xi}_{6,i} = n_2^{(0)}\left(n_{i+1}^{(2)} - n_i^{(2)}\right) + n_2^{(2)}\left(n_i^{(0)} - n_{i+1}^{(0)}\right) \\
\beta_{3,i} &\triangleq \mathbf{n}_3 \cdot \boldsymbol{\xi}_{6,i} = n_3^{(0)}\left(n_{i+1}^{(2)} - n_i^{(2)}\right) + n_3^{(2)}\left(n_i^{(0)} - n_{i+1}^{(0)}\right)
\end{aligned}\right\} \tag{51}$$

$$\left.\begin{aligned}
\gamma_{0,i} &\triangleq \mathbf{n}_0 \cdot \boldsymbol{\xi}_{6,i} = n_0^{(0)}\left(n_i^{(1)} - n_{i+1}^{(1)}\right) + n_0^{(1)}\left(n_{i+1}^{(0)} - n_i^{(0)}\right) \\
\gamma_{1,i} &\triangleq \mathbf{n}_1 \cdot \boldsymbol{\xi}_{6,i} = n_1^{(0)}\left(n_i^{(1)} - n_{i+1}^{(1)}\right) + n_1^{(1)}\left(n_{i+1}^{(0)} - n_i^{(0)}\right) \\
\gamma_{2,i} &\triangleq \mathbf{n}_2 \cdot \boldsymbol{\xi}_{6,i} = n_2^{(0)}\left(n_i^{(1)} - n_{i+1}^{(1)}\right) + n_2^{(1)}\left(n_{i+1}^{(0)} - n_i^{(0)}\right) \\
\gamma_{3,i} &\triangleq \mathbf{n}_3 \cdot \boldsymbol{\xi}_{6,i} = n_3^{(0)}\left(n_i^{(1)} - n_{i+1}^{(1)}\right) + n_3^{(1)}\left(n_{i+1}^{(0)} - n_i^{(0)}\right)
\end{aligned}\right\} \quad (52)$$

If we examine each of the above and fill in specific values of i, we learn that

$$\begin{array}{lll}
\alpha_{0,0} = \alpha_{1,0} & \beta_{0,0} = \beta_{1,0} & \gamma_{0,0} = \gamma_{1,0} \\
\alpha_{1,1} = \alpha_{2,1} & \beta_{1,1} = \beta_{2,1} & \gamma_{1,1} = \gamma_{2,1} \\
\alpha_{2,2} = \alpha_{3,2} & \beta_{2,2} = \beta_{3,2} & \gamma_{2,2} = \gamma_{3,2} \\
\alpha_{3,3} = \alpha_{0,3} & \beta_{3,3} = \beta_{0,3} & \gamma_{3,3} = \gamma_{0,3}
\end{array}$$

Therefore, we can eliminate one value from each maximum and minimum that define $r_{k'}$ and ρ, providing a modest boost to efficiency. Summarizing in a manner similar to Table 3, we have for each $0 \le i \le N-1$

Axis	$\boldsymbol{\xi}$	$\boldsymbol{\ell}$	$\boldsymbol{\ell'}$	$\boldsymbol{L}$
$5, i$	$\left(0, -f_i^{(2)}, f_i^{(1)}\right)^T$	$M_{5,i} - m_{5,i}$	$\max\limits_{j \neq i} \alpha_{j,i} - \min\limits_{j \neq i} \alpha_{j,i}$	$\max\left\{M_{5,i}, \max\limits_{j \neq i} \alpha_{j,i}\right\} - \min\left\{m_{5,i}, \min\limits_{j \neq i} \alpha_{j,i}\right\}$
$6, i$	$\left(f_i^{(2)}, 0, -f_i^{(0)}\right)^T$	$M_{6,i} - m_{6,i}$	$\max\limits_{j \neq i} \beta_{j,i} - \min\limits_{j \neq i} \beta_{j,i}$	$\max\left\{M_{6,i}, \max\limits_{j \neq i} \beta_{j,i}\right\} - \min\left\{m_{6,i}, \min\limits_{j \neq i} \beta_{j,i}\right\}$
$7, i$	$\left(-f_i^{(1)}, f_i^{(0)}, 0\right)^T$	$M_{7,i} - m_{7,i}$	$\max\limits_{j \neq i} \gamma_{j,i} - \min\limits_{j \neq i} \gamma_{j,i}$	$\max\left\{M_{7,i}, \max\limits_{j \neq i} \gamma_{j,i}\right\} - \min\left\{m_{7,i}, \min\limits_{j \neq i} \gamma_{j,i}\right\}$

Table 4. Vital Information for SAT (3D), Part II. This table contains all of the quantities for testing separating axes corresponding to edge–edge cross products in 3D.

in which we have introduced the quantities $M_{j,i} \triangleq \max_k\left(\boldsymbol{\xi}_{j,i} \cdot \mathbf{v}_k\right)$ and $m_{j,i} \triangleq \min_k\left(\boldsymbol{\xi}_{j,i} \cdot \mathbf{v}_k\right)$. Using the axis definitions Eq. (49) and reasoning similar to what produced Eq. (45) and Eq. (46), we can compute these values explicitly:

$$M_{5,i} = \left(n_i^{(2)} - n_{i+1}^{(2)}\right) \cdot \begin{cases} x_{\max}^{(1)} & \text{if } n_i^{(2)} \ge n_{i+1}^{(2)} \\ x_{\min}^{(1)} & \text{if } n_i^{(2)} < n_{i+1}^{(2)} \end{cases} + \left(n_{i+1}^{(1)} - n_i^{(1)}\right) \cdot \begin{cases} x_{\max}^{(2)} & \text{if } n_{i+1}^{(1)} \ge n_i^{(1)} \\ x_{\min}^{(2)} & \text{if } n_{i+1}^{(1)} < n_i^{(1)} \end{cases} \quad (53)$$

$$M_{6,i} = \left(n_{i+1}^{(2)} - n_i^{(2)}\right) \cdot \begin{cases} x_{\max}^{(0)} & \text{if } n_{i+1}^{(2)} \ge n_i^{(2)} \\ x_{\min}^{(0)} & \text{if } n_{i+1}^{(2)} < n_i^{(2)} \end{cases} + \left(n_i^{(0)} - n_{i+1}^{(0)}\right) \cdot \begin{cases} x_{\max}^{(2)} & \text{if } n_i^{(0)} \ge n_{i+1}^{(0)} \\ x_{\min}^{(2)} & \text{if } n_i^{(0)} < n_{i+1}^{(0)} \end{cases} \quad (54)$$

$$M_{7,i} = \left(n_i^{(1)} - n_{i+1}^{(1)}\right) \cdot \begin{cases} x_{\max}^{(0)} & \text{if } n_i^{(1)} \ge n_{i+1}^{(1)} \\ x_{\min}^{(0)} & \text{if } n_i^{(1)} < n_{i+1}^{(1)} \end{cases} + \left(n_{i+1}^{(0)} - n_i^{(0)}\right) \cdot \begin{cases} x_{\max}^{(2)} & \text{if } n_{i+1}^{(0)} \ge n_i^{(0)} \\ x_{\min}^{(2)} & \text{if } n_{i+1}^{(0)} < n_i^{(0)} \end{cases} \quad (55)$$

$$m_{5,i} = \left(n_i^{(2)} - n_{i+1}^{(2)}\right) \cdot \begin{cases} x_{\min}^{(1)} & \text{if } n_i^{(2)} \ge n_{i+1}^{(2)} \\ x_{\max}^{(1)} & \text{if } n_i^{(2)} < n_{i+1}^{(2)} \end{cases} + \left(n_{i+1}^{(1)} - n_i^{(1)}\right) \cdot \begin{cases} x_{\min}^{(2)} & \text{if } n_{i+1}^{(1)} \ge n_i^{(1)} \\ x_{\max}^{(2)} & \text{if } n_{i+1}^{(1)} < n_i^{(1)} \end{cases} \quad (56)$$

$$m_{6,i} = \left(n_{i+1}^{(2)} - n_i^{(2)}\right) \cdot \begin{cases} x_{\min}^{(0)} & \text{if } n_{i+1}^{(2)} \ge n_i^{(2)} \\ x_{\max}^{(0)} & \text{if } n_{i+1}^{(2)} < n_i^{(2)} \end{cases} + \left(n_i^{(0)} - n_{i+1}^{(0)}\right) \cdot \begin{cases} x_{\min}^{(2)} & \text{if } n_i^{(0)} \ge n_{i+1}^{(0)} \\ x_{\max}^{(2)} & \text{if } n_i^{(0)} < n_{i+1}^{(0)} \end{cases} \quad (57)$$

$$m_{7,i} = \left(n_i^{(1)} - n_{i+1}^{(1)}\right) \cdot \begin{cases} x_{\min}^{(0)} & \text{if } n_i^{(1)} \ge n_{i+1}^{(1)} \\ x_{\max}^{(0)} & \text{if } n_i^{(1)} < n_{i+1}^{(1)} \end{cases} + \left(n_{i+1}^{(0)} - n_i^{(0)}\right) \cdot \begin{cases} x_{\min}^{(2)} & \text{if } n_{i+1}^{(0)} \ge n_i^{(0)} \\ x_{\max}^{(2)} & \text{if } n_{i+1}^{(0)} < n_i^{(0)} \end{cases} \quad (58)$$

We conclude by presenting the complete algorithm for inserting of faces into boxes in 3D. The algorithm computes nothing until it is needed and is designed to maximize the efficiency of early exits.

Algorithm 4 (Insertion in 3D)

```
Let $\{\mathbf{n}_i\}_{i=0}^{\bar{N}-1}$ be the set of nodes on the face $\mathcal{F}$ to be inserted.
  1. If any $\mathbf{n}_i \in \mathcal{B}$, add the face to the box and exit.
  2. Compute the box diagonal $\mathbf{d} = \mathbf{x}_{\max} - \mathbf{x}_{\min}$.
  3. Enter loop to test box face normals as separating axes: for $i \in \{0, 1, 2\}$,
       $(L_1)$ Compute $M_i = \max_j n_j^{(i)}$ and $m_i = \min_j n_j^{(i)}$.
       $(L_2)$ Axis Test #(i+1): if $\max\{x_{\text{max}}^{(i)}, M_i\} - \min\{x_{\text{min}}^{(i)}, m_i\} > d^{(i)} + M_i - m_i$, exit.
  4. Let $\boldsymbol{\nu}$ be the normal of $\mathcal{F}$. Compute $M$ via Eq. (47) and $m$ via Eq. (48). Compute $M_\nu = \max_i \boldsymbol{\nu} \cdot \mathbf{n}_i$
     and $m_\nu = \min_i \boldsymbol{\nu} \cdot \mathbf{n}_i$.
  5. Axis Test #4: if $\max\{M, M_\nu\} - \min\{m, m_\nu\} > M - m + M_\nu - m_\nu$, exit.
  6. Enter loop over edges: for $i \in \{0, \dots, \bar{N} - 1\}$,
       $(L_1')$ For $j \in \{0, \dots, \bar{N} - 1\} \backslash \{i\}$, compute $\alpha_{i,j}$ via Eq. (50).
       $(L_2')$ Compute $A = \max \alpha_{j,i}$, $a = \min \alpha_{j,i}$, $M_{5,i}$ via Eq. (53), and $m_{5,i}$ via Eq. (56).
       $(L_3')$ Axis Test #(5,$i$): if $\max\{M_{5,i}, A\} - \min\{m_{5,i}, a\} > M_{5,i} - m_{5,i} + A - a + \tau$, exit.
       $(L_4')$ For $j \in \{0, \dots, \bar{N} - 1\} \backslash \{i\}$, compute $\beta_{i,j}$ via Eq. (51).
       $(L_5')$ Compute $B = \max \beta_{j,i}$, $b = \min \beta_{j,i}$, $M_{6,i}$ via Eq. (54), and $m_{6,i}$ via Eq. (57).
       $(L_6')$ Axis Test #(6,$i$): if $\max\{M_{6,i}, B\} - \min\{m_{6,i}, b\} > M_{6,i} - m_{6,i} + B - b + \tau$, exit.
       $(L_7')$ For $j \in \{0, \dots, \bar{N} - 1\} \backslash \{i\}$, compute $\gamma_{i,j}$ via Eq. (52).
       $(L_8')$ Compute $G = \max \gamma_{j,i}$, $g = \min \gamma_{j,i}$, $M_{7,i}$ via Eq. (55), and $m_{7,i}$ via Eq. (58).
       $(L_9')$ Axis Test #(7,$i$): if $\max\{M_{7,i}, G\} - \min\{m_{7,i}, g\} > M_{7,i} - m_{7,i} + G - g + \tau$, exit.
  7. Reaching this point means no separating axis exists: add the face to the box.
```

The mysterious τ that has suddenly appeared in the edge–edge axes is a small parameter added for numerical robustness. When an edge $\mathbf{f}_i$ is aligned with a coordinate axis (or very nearly so), the cross product is zero. Consequently, all projections onto this axis are zero; in perfect arithmetic, the comparison is $0 > 0$, which is false. Round-off error and other imprecisions may spoil this calculation and lead to a false positive.

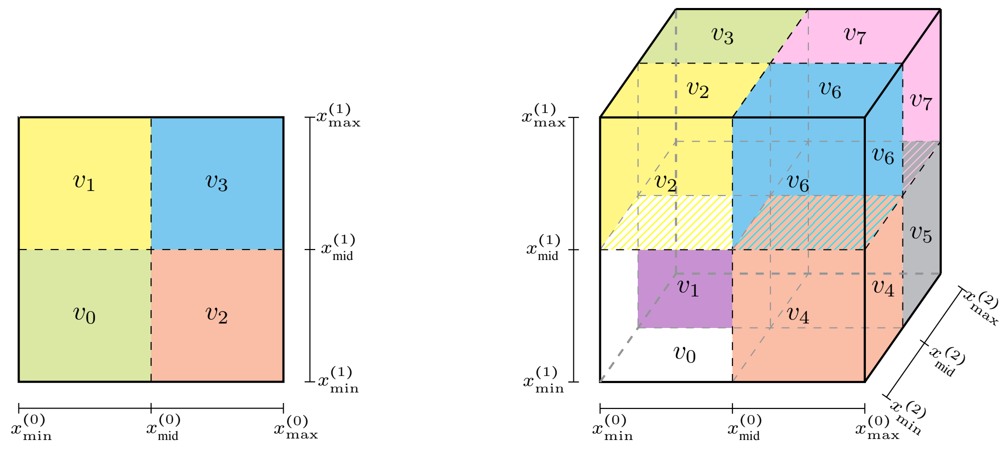
D. Child Boxes

We add data to a box one piece at a time. When the size of the data associated with a box reaches the maximum allowed bin size, we refine the box by dividing the box in two along each dimension, yielding four child boxes in 2D and eight in 3D. The data from the original box is removed and copies distributed to each of the newly-formed child boxes. A box may have an empty set of children or empty set of data but not both; a box with data is said to be *active*.

The child boxes are assigned numbers that indicate their position in the data structure that contains them (generally, a `vector` or `array` in the language of implementation). Perhaps the most natural ordering is based on binary counting: each bit of the child's number written as a binary string of length d gives whether the corresponding term in the Cartesian product describing the child's region of space should be the upper or lower half of the original. For example, for the fifth child, $5 \mapsto 101$ indicates that it is the region

$$\left[x_{\text{mid}}^{(0)}, x_{\text{max}}^{(0)}\right] \times \left[x_{\text{min}}^{(1)}, x_{\text{mid}}^{(1)}\right] \times \left[x_{\text{mid}}^{(2)}, x_{\text{max}}^{(2)}\right]$$

This is, however, not the optimal ordering as far as traversal is concerned. Instead, we use the ordering shown in Figure 5 below. The benefit of doing so will become apparent in the next section.



2D **3D**

Figure 5. Numbering of Child Boxes. The unconventional ordering of the child boxes, due to Revelles, Ureña, and Lastra [8] will become incredibly convenient in the traversal algorithm.

E. The Traversal Algorithm

The previous section is dedicated to building the tree: all the intersection methods are designed for determining whether faces belong in a particular box. With that critical matter settled, we can turn our attention to the crux of this report: speedy ray-box intersection. It is for this purpose we built the tree.

Determining whether a ray pierces a box is, in principle, identical to the logic that determines whether the edge between two nodes intersects a bounding box. We have developed an algorithm based on the Hyperplane Separation Theorem. Doing so created a highly robust and efficient method that identifies all possible types of intersections (face-face, edge-face, edge-edge, and vertex-anything) with minimal effort. However, for ray-face intersections, the SAT is a poor choice for determining intersection, as it gives no information about the location of intersection; consequently, every child box must be checked from scratch. As typical applications involve some hundred thousand rays, even a small gain in the efficiency of the ray-box intersect method can amplify performance significantly.

F. Intersection with Child Boxes

Remember that each box in the tree has exclusively either data ("active") or children. Once intersection is determined, if that box is not active, we proceed downward to child boxes. Tree traversal therefore goes:

1. If the ray pierces an active box, return that box's (non-empty) dataset.
2. Otherwise, determine the child boxes the ray pierces. Return to Step 1 above for each such box.

The recursive nature of traversal motivates a top-down approach based on entry and exit times. In this manner, we can use knowledge from the previous step to reduce the number of recursive calls that must be made.

Suppose we have a ray

$$\mathfrak{r}(t) = \mathbf{p} + t\mathbf{r}, \quad t \geq 0 \tag{59}$$

with the ray direction $\mathbf{r}$ normalized ($\mathbf{p}$ is the origin of the ray). We begin by computing the entry and exit times for the mother box $\mathbb{B}$ (the topmost and largest box) for each component:

$$t_0^{(i)} \triangleq \frac{x^{(i)}_{\text{entry}} - p_0^{(i)}}{r^{(i)}} \qquad t_1^{(i)} \triangleq \frac{x^{(i)}_{\text{exit}} - p_0^{(i)}}{r^{(i)}} \tag{60}$$

where the entry and exit locations are selected based on the sign of $r^{(i)}$:

$$x^{(i)}_{\text{entry}} \triangleq \begin{cases} x^{(i)}_{\min} & \text{if } r^{(i)} > 0 \\ x^{(i)}_{\max} & \text{if } r^{(i)} < 0 \end{cases} \qquad x^{(i)}_{\text{exit}} \triangleq \begin{cases} x^{(i)}_{\max} & \text{if } r^{(i)} > 0 \\ x^{(i)}_{\min} & \text{if } r^{(i)} < 0 \end{cases} \tag{61}$$

We immediately notice a hole in the selection when $r^{(i)} = 0$. This will addressed in Section H below. For now, it is sufficient to assume that $r^{(i)} \neq 0$. We also note that since $\mathbf{p} \in \mathbb{B}$, $t_0^{(i)} < 0$ for each i, entry times not well-suited for top-down traversal. Accordingly, it is necessary to move the origin $\mathbf{p}$ to a new point outside the box that is on the negative ray trajectory. A seemingly reasonable approach is to define $t^* \triangleq \max t_0^{(i)} - \delta$ for some one-size-fits-all $\delta > 0$ and set $\mathbf{p} \leftarrow \mathfrak{r}(t^*)$. This, however, does not take into account the physical size of the grid. If, for instance, $x_{\max}^{(i)} - x_{\min}^{(i)} = \mathcal{O}(10^{-4})$, then since $|r| = 1$, the new origin will be far outside $\mathbb{B}$. If the size is large, then the new $\mathbf{p}$ will be very close and Eq. (60) may be numerically ill-conditioned due to the subtraction of nearly equal numbers in the numerator. Instead, we execute Algorithm 5:

Algorithm 5

1. Set $\mathbf{q} = \mathbf{p}$ to remember the original origin. Define

$$\Delta t \triangleq \frac{1}{5} \min_{1 \le i \le d} \left\{ \frac{x_{\max}^{(i)} - x_{\min}^{(i)}}{\left| r^{(i)} \right|} \right\} \tag{62}$$

2. While $\mathbf{p} \in \mathbb{B}$, set $\mathbf{p} \leftarrow \mathbf{p} - \mathbf{r}\Delta t$.
3. Set $t_{pq} = n\Delta t$, where n is the number of times Step 3 above was executed. This value gives the time required to reach the new origin $\mathbf{p}$ from the old one $\mathbf{q}$.

There is nothing magical about the factor of $\frac{1}{5}$; it is chosen to balance the desire to locate a point just outside the box (but not too near) with the effort required to do so. When $r^{(i)} \approx 0$, the division in Eq. (62) will be ill-conditioned. Special consideration is not needed, as the minimum taken in Eq. (62) snuffs out problematic division. Because $|\mathbf{r}| = 1$, if $\left| r^{(i)} \right| = \varepsilon \ll 1$ for some i, there exists i' such that

$$\left| r^{(i')} \right| \ge \sqrt{\frac{1 - \varepsilon^2}{2}} \tag{63}$$

Recall that child boxes are formed by dividing the parent box in half in each coordinate. Rather than each child box and compute entry and exit times for each via Eq. (60), we can compute the midpoint time for each coordinate from the entry and exit times of the parent box:

$$t_m^{(i)} \triangleq \frac{1}{2}\left[t_0^{(i)} + t_1^{(i)} \right] \tag{64}$$

The value $t_m^{(i)}$ gives the time at which the ray passes through the plane (line in 2D) $\mathbf{e}_i \cdot (\mathbf{x} - \mathbf{x}_{\text{mid}})$. Now suppose that $r^{(i)} > 0$ for each i and consider the following 2D example:

The ray in Figure 6 pierces boxes v_0, v_2, and v_3. The intervals of t for which $\mathfrak{r}(t)$ remains in each box are:

$$\begin{aligned}
v_0 : \quad & t \in \left[t_0^{(1)}, t_m^{(0)} \right] = \left[t_0^{(0)}, t_m^{(0)} \right] \bigcap \left[t_0^{(1)}, t_m^{(1)} \right] \\
v_2 : \quad & t \in \left[t_m^{(0)}, t_m^{(0)} \right] = \left[t_m^{(0)}, t_1^{(0)} \right] \bigcap \left[t_0^{(1)}, t_m^{(1)} \right] \\
v_3 : \quad & t \in \left[t_m^{(1)}, t_1^{(0)} \right] = \left[t_m^{(0)}, t_1^{(0)} \right] \bigcap \left[t_m^{(1)}, t_1^{(1)} \right]
\end{aligned}$$

The ray does not intersect v_1 because

$$\left[t_0^{(0)}, t_m^{(0)} \right] \bigcap \left[t_m^{(1)}, t_1^{(1)} \right] = \emptyset$$

From the above, we can easily collect for each box the necessary and sufficient condition for a ray to pierce it as well as the entry and exit times in Table 5 below.

While we have omitted a diagram such as Figure 6 for the three-dimensional case, it is not difficult to extrapolate up one dimension. Table 6 gives the entry and exit times for the child boxes in 3D.

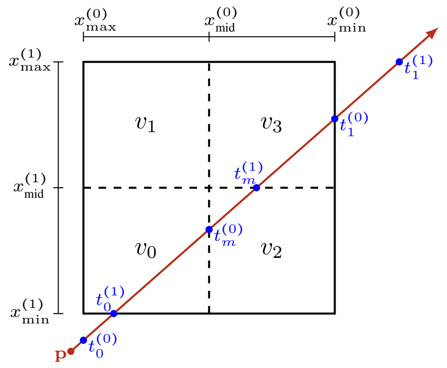


Figure 6. Sample Ray Piercing in 2D. Marked are the entry times $t_0^{(0)}$ and $t_0^{(1)}$, exit times $t_1^{(0)}$ and $t_1^{(1)}$, and the midpoint times $t_m^{(i)}$. These times indicate the value of t such that the ray Eq. (59) first intersects the lines $x^{(0)} = x^{(0)}_{\min}$, $x^{(1)} = x^{(1)}_{\min}$, $x^{(0)} = x^{(0)}_{\max}$, $x^{(1)} = x^{(0)}_{\max}$, $x^{(0)} = x^{(0)}_{\mathbf{mid}}$, and $x^{(1)} = x^{(1)}_{\mathbf{mid}}$, respectively.

Child Box	**Condition for Entry**	**Entry Times**		**Exit Times**	
v_0	$\max\left\{t_0^{(0)}, t_0^{(1)}\right\} < \min\left\{t_m^{(0)}, t_m^{(1)}\right\}$	$t_0^{(0)}$	$t_0^{(1)}$	$t_m^{(0)}$	$t_m^{(1)}$
v_1	$\max\left\{t_0^{(0)}, t_m^{(1)}\right\} < \min\left\{t_m^{(0)}, t_1^{(1)}\right\}$	$t_0^{(0)}$	$t_m^{(1)}$	$t_m^{(0)}$	$t_1^{(1)}$
v_2	$\max\left\{t_m^{(0)}, t_0^{(1)}\right\} < \min\left\{t_1^{(0)}, t_m^{(1)}\right\}$	$t_m^{(0)}$	$t_0^{(1)}$	$t_1^{(0)}$	$t_m^{(1)}$
v_3	$\max\left\{t_m^{(0)}, t_m^{(1)}\right\} < \min\left\{t_1^{(0)}, t_1^{(1)}\right\}$	$t_m^{(0)}$	$t_m^{(1)}$	$t_1^{(0)}$	$t_1^{(1)}$

Table 5. Child Box Entry in 2D. This table gives the condition necessary to pierce and new entry/exit times for each child box in 2D. The child boxes are numbered as displayed in Figure 5. All components of r must be nonnegative for this table to be valid.

Child Box	**Condition for Entry**	**Entry Times**			**Exit Times**		
v_0	$\max\left\{t_0^{(0)}, t_0^{(1)}, t_0^{(2)}\right\} < \min\left\{t_m^{(0)}, t_m^{(1)}, t_m^{(2)}\right\}$	$t_0^{(0)}$	$t_0^{(1)}$	$t_0^{(2)}$	$t_m^{(0)}$	$t_m^{(1)}$	$t_m^{(2)}$
v_1	$\max\left\{t_0^{(0)}, t_0^{(1)}, t_m^{(2)}\right\} < \min\left\{t_m^{(0)}, t_m^{(1)}, t_1^{(2)}\right\}$	$t_0^{(0)}$	$t_0^{(1)}$	$t_m^{(2)}$	$t_m^{(0)}$	$t_m^{(1)}$	$t_1^{(2)}$
v_2	$\max\left\{t_0^{(0)}, t_m^{(1)}, t_0^{(2)}\right\} < \min\left\{t_m^{(0)}, t_1^{(1)}, t_m^{(2)}\right\}$	$t_0^{(0)}$	$t_m^{(1)}$	$t_0^{(2)}$	$t_m^{(0)}$	$t_1^{(1)}$	$t_1^{(2)}$
v_3	$\max\left\{t_0^{(0)}, t_m^{(1)}, t_m^{(2)}\right\} < \min\left\{t_m^{(0)}, t_1^{(1)}, t_1^{(2)}\right\}$	$t_0^{(0)}$	$t_m^{(1)}$	$t_m^{(2)}$	$t_m^{(0)}$	$t_1^{(1)}$	$t_1^{(2)}$
v_4	$\max\left\{t_m^{(0)}, t_0^{(1)}, t_0^{(2)}\right\} < \min\left\{t_1^{(0)}, t_m^{(1)}, t_m^{(2)}\right\}$	$t_m^{(0)}$	$t_0^{(1)}$	$t_0^{(2)}$	$t_1^{(0)}$	$t_m^{(1)}$	$t_m^{(2)}$
v_5	$\max\left\{t_m^{(0)}, t_0^{(1)}, t_m^{(2)}\right\} < \min\left\{t_1^{(0)}, t_m^{(1)}, t_1^{(2)}\right\}$	$t_m^{(0)}$	$t_0^{(1)}$	$t_m^{(2)}$	$t_1^{(0)}$	$t_m^{(1)}$	$t_1^{(2)}$
v_6	$\max\left\{t_m^{(0)}, t_m^{(1)}, t_0^{(2)}\right\} < \min\left\{t_1^{(0)}, t_1^{(1)}, t_m^{(2)}\right\}$	$t_m^{(0)}$	$t_m^{(1)}$	$t_0^{(2)}$	$t_1^{(0)}$	$t_1^{(1)}$	$t_m^{(2)}$
v_7	$\max\left\{t_m^{(0)}, t_m^{(1)}, t_m^{(2)}\right\} < \min\left\{t_1^{(0)}, t_1^{(1)}, t_1^{(2)}\right\}$	$t_m^{(0)}$	$t_m^{(1)}$	$t_m^{(2)}$	$t_1^{(0)}$	$t_1^{(1)}$	$t_1^{(2)}$

Table 6. Child Box Entry in 3D. This table gives the condition necessary to pierce and new entry/exit times for each child box in 3D. The child boxes are numbered as displayed in Figure 5. All components of r must be nonnegative for this table to be valid.

G. Rays with Negative Components

Table 5 and Table 6 are valid only when $r^{(i)} \geq 0$ for all i. To see how things change when directions are negative, we consider reversing the direction of the ray in Figure 6:

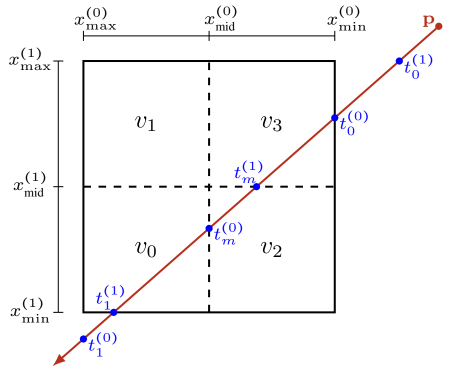


Figure 7. Sample Ray with Negative Directions. This ray is the reversal of the one in Figure 6.

In Figure 7 above, the ray still enters the same set of boxes but does so in the reverse order (obviously). In this case, the entry and exit times are not given by Table 5; rather, they are

$$\begin{aligned}
v_3 : \quad & t \in \left[t^{(1)}_0, t^{(0)}_m\right] = \left[t^{(0)}_0, t^{(0)}_m\right] \bigcap \left[t^{(1)}_0, t^{(1)}_m\right] \\
v_2 : \quad & t \in \left[t^{(1)}_m, t^{(0)}_m\right] = \left[t^{(0)}_0, t^{(0)}_m\right] \bigcap \left[t^{(1)}_m, t^{(1)}_1\right] \\
v_0 : \quad & t \in \left[t^{(1)}_m, t^{(0)}_1\right] = \left[t^{(0)}_m, t^{(0)}_1\right] \bigcap \left[t^{(1)}_m, t^{(1)}_1\right]
\end{aligned}$$

If the conditions listed in the table are used to determine intersection, the result would be all failures—no intersection would be detected at all. It follow from the above that when both components of $\mathbf{r}$ are negative, the intersection condition and times for v_3 are those of v_0 from Table 5; for v_2, of v_1; and for v_0, of v_3.

While the knee-jerk reaction is that it is necessary to reformulate Table 5 for each possible combination of component signs, the clever approach is to simply relabel the boxes. In Figure 7, if we simply map $v_0 \rightleftarrows v_3$ and $v_1 \rightleftarrows v_2$, then Table 5 becomes valid again. In general, we can find the mapping by letting

$$a \triangleq \begin{cases} 2s(0) + s(1) & \text{if } d = 2 \\ 4s(0) + 2s(1) + s(2) & \text{if } d = 3 \end{cases} \tag{65}$$

where

$$s(i) \triangleq \begin{cases} 0 & \text{if } r^{(i)} \geq 0 \\ 1 & \text{if } r^{(i)} < 0 \end{cases} \tag{66}$$

Written as a binary string of length d, a encodes the signs of the components of $\mathbf{r}$, with 1 representing a negative value and 0 a positive or zero. The new box labels are then given by

$$\ell \mapsto \ell \oplus a \tag{67}$$

where ℓ is the integer label of the original box ($\oplus$ is bitwise `xor`). In the example, $a = 3$, or $a_{\text{bin}} = 11$, so that

Original Label (Decimal)	**Original Label (Binary)**	**New Label (Binary)**	**New Label (Decimal)**
0	00	$00 \oplus 11 = 11$	3
1	01	$01 \oplus 11 = 10$	2
2	10	$10 \oplus 11 = 01$	1
3	11	$11 \oplus 11 = 00$	0

just as was previously indicated.

While not demonstrated, Eq. (67) is also compatible with the numbering scheme in 3D shown in Figure 5. The idiosyncratic child box ordering displays its power: it allows for reuse of Table 5 and Table 6 with a very simple relabeling scheme. This clever trick is due to Revelles, Ureña, and Lastra [8]; our automatic entry/exit point selection Eq. (61) eliminates the need to reflect the origin $\mathbf{p}$, an improvement on their technique.

H. Rays with Zero Components: The Infinite Arithmetic Module

As mentioned, Eq. (61) provides no selection of entry and exit locations when $r^{(i)} = 0$. Moreover, $\mathbf{p}$ will be much closer to one of $\mathbf{x}_{\min}$ or $\mathbf{x}_{\max}$. If $\left|r^{(i)}\right| = \varepsilon \ll 1$, the division by this small number in Eq. (60) will amplify this difference significantly. As $t_m^{(i)}$ is computed at each recursive level, it is possible for a slight error to accumulate to the point where valid intersections are no longer properly detected.

When $\left|r^{(i)}\right| < \varepsilon$, there will be little change in the ith component during the ray's traversal across $\mathbb{B}$. Recalling that $|\mathbf{r}| = 1$, it follows that a small $r^{(i)}$ is exceptionally unlikely to cause exit from $\mathbb{B}$. To illustrate, suppose $r^{(i)} = \varepsilon > 0$ and that exit is induced by this component. The total travel time across the box is

$$t^* \triangleq \min_{1\le i\le d} t_1^{(i)} - \max_{1\le i\le d} t_0^{(i)}$$

and satisfies the relation

$$\begin{aligned} x_{\max}^{(i)} &= p^{(i)} + t^* r^{(i)} \\ &= p^{(i)} + t^* \varepsilon \end{aligned}$$

Let i' be the index of largest component in magnitude of $\mathbf{r}$ and suppose that $r^{(i')} > 0$. For exit not to occur first in component i', we need to have

$$p^{(i')} + \tau r^{(i')} < x_{\max}^{(i')} \tag{68}$$

But by Eq. (63), we have

$$\begin{aligned} p^{(i')} + \tau r^{(i')} &\ge p^{(i)} + t^* \sqrt{\frac{1-\varepsilon^2}{2}} \\ &= p^{(i')} + \frac{x_{\max}^{(i)} - p^{(i)}}{\varepsilon} \sqrt{\frac{1-\varepsilon^2}{2}} \\ &= p^{(i')} + \left(x_{\max}^{(i)} - p^{(i)}\right)\sqrt{\frac{1}{2\varepsilon^2} - 1} \end{aligned}$$

In light of the above, Eq. (68) becomes

$$x_{\max}^{(i')} - p^{(i')} > \left(x_{\max}^{(i)} - p^{(i)}\right)\sqrt{\frac{1}{2\varepsilon^2} - 1}$$

The above implies that $p^{(i')}$ must be several orders of magnitude closer to $x_{\max}^{(i')}$ than $p^{(i)}$ is to $x_{\max}^{(i)}$. In the other cases (combinations of positive and negative direction components), the above will instead involve $x_{\min}$ and a reversal of the inequality. The key measure remains the relative distance from origin to exit location.

Because the ith component is not likely to be the limiting component, we simply set

$$t_0^{(i)} = -\infty \qquad\qquad t_1^{(i)} = +\infty$$

whenever $\left|r^{(i)}\right| < \tau$ for some tolerance τ. These indicate that

$$x_{\min}^{(i)} \le \mathfrak{r}^{(i)}(t) \le x_{\max}^{(i)}$$

for all t. When $r^{(i)} \ne 0$, this is not technically true, but, for all intents and purposes, it is for our problem.

When infinite values are present in $t_0^{(i)}$ and $t_1^{(i)}$, $t_m^{(i)}$ as computed by Eq. (64) is obviously not well-defined. To remedy, we set

$$t_m^{(i)} = \begin{cases} +\infty & \text{if } p^{(i)} < x_{\text{mid}}^{(i)} \\ -\infty & \text{if } p^{(i)} \geq x_{\text{mid}}^{(i)} \end{cases} \tag{69}$$

The logic of this section, which we dub the "infinite arithmetic module," can result in misidentified intersections in very rare scenarios (exampled above). However, due to the stochastic nature of the Monte Carlo algorithms, isolated failures for the occasional misbehaving ray are not problematic.

I. Ray–Face Intersect Test

Once we've traced the ray's path through the tree boxes, we must determine which faces are intersected among those collected and which of these is the closest. In 2D, determining whether a ray and a segment intersect is rather simple. In 3D, we again call upon the SAT to formulate a ray–face intersection test.

The reader has undoubtedly tired of detailed explanations regarding the SAT, so we merely state the methodology without further justification. For a triangle, we compute the signed volumes of the tetrahedra formed by two of the triangle edges with the ray direction. If all three of these quantities do **not** share the same sign, we can conclude the existence of a separating hyperplane. Algorithm 6 below details the process.

Algorithm 6 (Line–Triangle Intersection)

Let $\mathbf{n}_0$, $\mathbf{n}_1$, and $\mathbf{n}_2$ be the vertices (nodes) of the triangle.

1. For each $0 \leq i \leq 2$, put $\mathbf{v}_i \triangleq \mathbf{n}_i - \mathbf{p}$.
2. Compute the following quantities (volumes of tetrahedra):

$$u \triangleq \mathbf{v}_1 \cdot (\mathbf{r} \times \mathbf{v}_2) \qquad v \triangleq -\mathbf{v}_0 \cdot (\mathbf{r} \times \mathbf{v}_2) \qquad w \triangleq \mathbf{v}_1 \cdot (\mathbf{r} \times \mathbf{v}_0)$$

3. If $uv > 0$ and $uw > 0$, then the line intersects the triangle.

It is important to note that the above test is not a ray–triangle method but rather a line–intersect triangle. Consequently, the above will detect intersection with faces along the backward trajectory. Intersection is correct only if the ray direction and face normal point in opposite directions—mathematically, $\boldsymbol{\nu} \cdot \mathbf{r} < 0$, where $\boldsymbol{\nu}$ is the outward-facing normal of the face. Care must be taken when the normal is constructed via edge-edge cross-product.

A quadrilateral may be viewed as two triangles glued together. An easy algorithm for quadrilaterals comes from applying Algorithm 6 to the two subtriangles formed by dividing the quadrilateral along its diagonal:

Algorithm 7 (Line–Quadrilateral Intersection)

Let $\mathbf{n}_0$, $\mathbf{n}_1$, $\mathbf{n}_2$, and $\mathbf{n}_3$ be the vertices (nodes) of the quadrilateral.

1. For each $0 \leq i \leq 2$, put $\mathbf{v}_i \triangleq \mathbf{n}_i - \mathbf{p}$.
2. Compute the following quantities (volumes of tetrahedra):

$$u \triangleq \mathbf{v}_1 \cdot (\mathbf{r} \times \mathbf{v}_2) \qquad v \triangleq -\mathbf{v}_0 \cdot (\mathbf{r} \times \mathbf{v}_2) \qquad w \triangleq \mathbf{v}_1 \cdot (\mathbf{r} \times \mathbf{v}_0)$$

3. If $uv > 0$ and $uw > 0$, the line intersects the subtriangle formed by $\mathbf{n}_0$, $\mathbf{n}_1$, and $\mathbf{n}_2$. Exit.
4. Otherwise, put $\mathbf{v}_3 \triangleq \mathbf{n}_3 - \mathbf{p}$ and compute the following:

$$\hat{u} \triangleq \mathbf{v}_0 \cdot (\mathbf{r} \times \mathbf{v}_2) \qquad \hat{v} \triangleq -\mathbf{v}_3 \cdot (\mathbf{r} \times \mathbf{v}_2) \qquad \hat{w} \triangleq \mathbf{v}_3 \cdot (\mathbf{r} \times \mathbf{v}_0)$$

5. If $\hat{u}\hat{v} > 0$ and $\hat{u}\hat{w} > 0$, then the line intersects the subtriangle formed by $\mathbf{n}_2$, $\mathbf{n}_3$, and $\mathbf{n}_0$.

As with Algorithm 6, it is critical to include a post-check of whether $\boldsymbol{\nu} \cdot \mathbf{r} < 0$ to determine whether the face is encountered forward or backward in time along the ray's trajectory.

J. The Complete Algorithm

We are now ready to assemble the pieces forged in the preceding subsections of Section E into a clear and concise algorithm for traversing the tree and locating the closest face of intersection:

Algorithm 8 (Tree Traversal and Search)

Suppose a ray with direction $\mathbf{r}$ and origin $\mathbf{q} \in \mathcal{F}_o \subset \mathbb{B}$.

1. Execute Algorithm 5 to translate $\mathbf{p}$ outside $\mathbb{B}$.
2. Fix a tolerance τ. For each i, compute the entry time $t_0^{(i)}$ and exit time $t_1^{(i)}$:
    - Select the entry and exit locations via Eq. (61).
    - If $\left|r^{(i)}\right| > \tau$, use Eq. (60). Otherwise, set $t_0^{(i)} = -\infty$ and $t_1^{(i)} = +\infty$.
3. Determine the "negative direction components." Because components such that $\left|r^{(i)}\right| < \tau$ have essentially been set to zero by the infinite arithmetic module, we should set $s(i) = 1$ if $r^{(i)} > -\tau$ rather than the $r^{(i)} \geq 0$ of Eq. (66). With this change, compute a via Eq. (65).
4. Allocate space for a data container `collection`. Begin recursive traversal:

    (L_1) Compute the (absolute) entry time $t_0 = \max_i t_0^{(i)}$.

    (L_2) If this box is active: if $\mathbf{p}$ is contained in it or $t_0 > t_{pq}$, where t_{pq} is the time to the original origin per Algorithm 5, add this box's data to `collection` and terminate.

    (L_3) Otherwise, for each i, compute the midpoint time $t_m^{(i)}$:
    - If $t_0^{(i)}$ or $t_1^{(i)}$ is infinite (either both are or neither is), use Eq. (69).
    - Otherwise, use Eq. (64).

    (L_4) Using the relabeling scheme Eq. (67), determine which child boxes are pierced by the ray using whichever of Table 5 or Table 6 is appropriate for the dimension.

    (L_5) For each child box entered as determined above, execute Step 4 until termination.
5. Search `collection` to determine the closest face of intersection (if any): initialize $d_{\min} = 0$ and $\mathcal{F}_{\min} =$ `null`. Then `for` $\mathcal{F}_t \in$ `collection`,

    (L_1') Determine whether $\mathcal{F}_t$ is visible from $\mathcal{F}_o$. If not, go to the loop top.

    (L_2') Determine intersection with $\mathcal{F}_t$ via Algorithm 6 or Algorithm 7 (whichever proper).

    (L_3') If intersecting and $|\texttt{centroid}(\mathcal{F}_o) - \texttt{centroid}(\mathcal{F}_t)| < d_{\min}$ and $\mathbf{r} \cdot \boldsymbol{\nu}(\mathcal{F}_t) < 0$, set

$$
\begin{aligned}
d_{\min} &\hookleftarrow |\texttt{centroid}(\mathcal{F}_o) - \texttt{centroid}(\mathcal{F}_t)| \\
\mathcal{F}_{\min} &\hookleftarrow \mathcal{F}_t
\end{aligned}
$$

6. If $\mathcal{F}_{\min} \neq$ `null`, increment the counter for $\mathcal{F}_{\min}$ among the tallies of $\mathcal{F}_o$.

In Step (L_1'), a face $\mathcal{F}_t$ is visible from $\mathcal{F}_o$ if either

1. The line segment connecting the centroid of $\mathcal{F}_o$ to that of $\mathcal{F}_t$ is on the same side as the normal of $\mathcal{F}_t$.
2. The above is true instead for any line segment connecting a node of $\mathcal{F}_o$ to a node of $\mathcal{F}_t$.

Being on “the same side” is equivalent to the dot product between the two vectors being negative.

K. Performance

The code was tested on a variety of problem geometries, shown in Figure 8 below. The table below compares the performance of the tree-based search with that of a previous version the code which used exhaustive search.

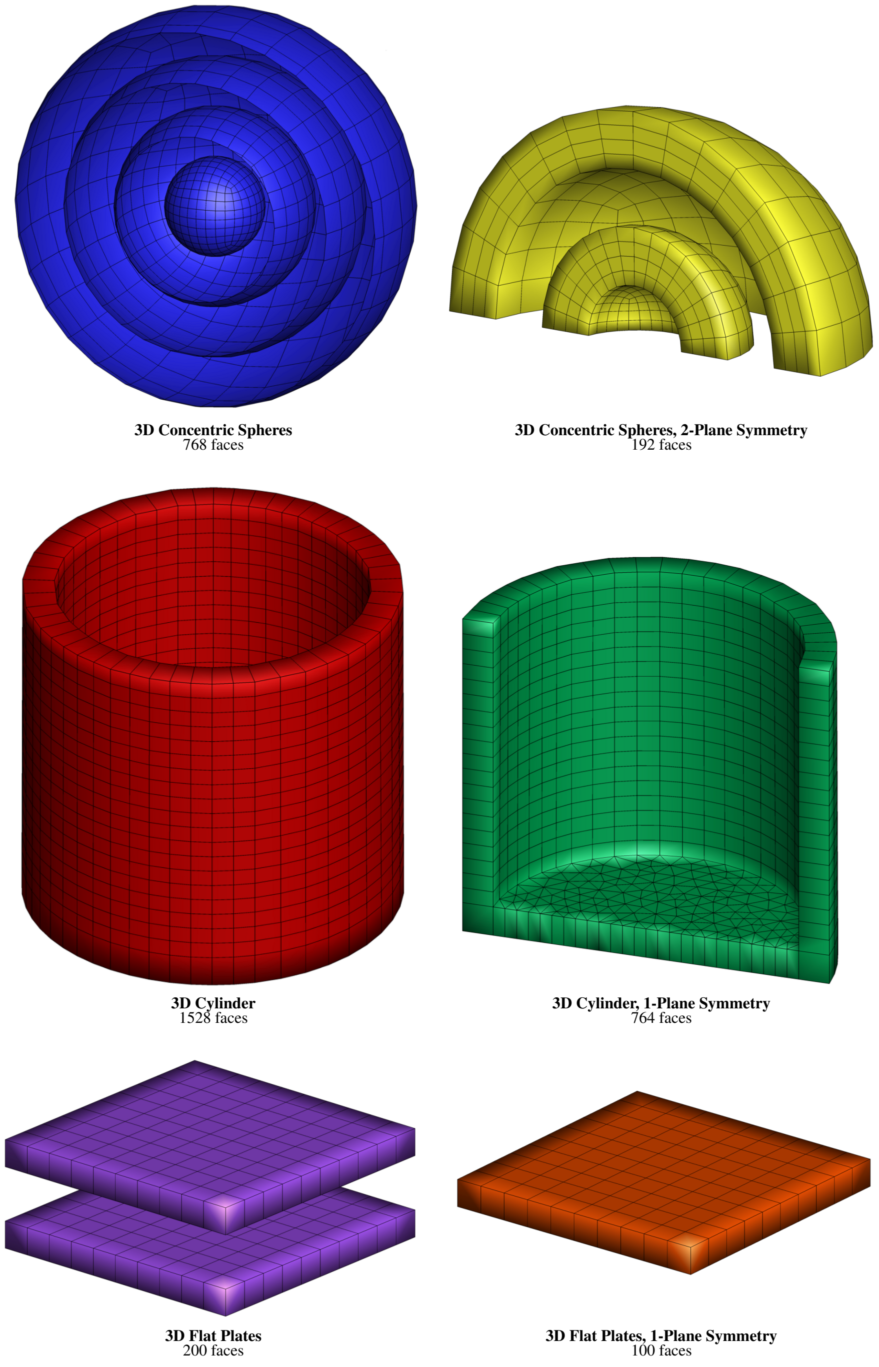


Figure 8. Rogues Gallery of Test Problems. These are geometries used to test the octree-based code. The 3D concentric spheres example has been sliced open to show the interior features.

The “3D Cylinder, Fine Mesh” has the same geometry as shown in the figure, just with many more grid points.

3D Concentric Spheres (5k Rays)
768 faces, Exhaustive Search: 47.69s

Bin Size	Run Time	Speed Up
30	9.72s	4.9x
20	7.99s	6.0x
10	5.97s	8.0x
5	4.44s	10.7x

2D Cavity (5k Rays)
216 faces, Exhaustive Search: 3.01s

Bin Size	Run Time	Speed Up
30	1.48s	2.0x
20	1.39s	2.2x
10	1.27s	2.4x
5	1.02s	3.0x

3D Spheres, 2-Plane Symmetry (5k Rays)
192 faces, Exhaustive Search: 20.52s

Bin Size	Run Time	Speed Up
30	2.59s	7.9x
20	2.15s	9.5x
10	1.86s	11.0x
5	1.51s	13.6x

3D Flat Plates (5k Rays)
200 faces, Exhaustive Search: 3.57s

Bin Size	Run Time	Speed Up
30	1.31s	2.8x
20	1.27s	2.8x
10	0.88s	4.1x
5	0.70s	5.1x

3D Cylinder (10k Rays)
1528 faces, Exhaustive Search: 307.05s

Bin Size	Run Time	Speed Up
30	41.41s	7.4x
20	21.83s	14.1x
10	17.95s	17.1x
8	14.07s	21.8x

3D Flat Plates, 1-Plane Symmetry (5k Rays)
100 faces, Exhaustive Search: 2.50s

Bin Size	Run Time	Speed Up
30	0.96s	3.1x
20	0.87s	3.8x
10	0.64s	4.8x
5	0.48s	5.2x

3D Cylinder, 1-Plane Symmetry (10k Rays)
764 faces, Exhaustive Search: 77.22s

Bin Size	Run Time	Speed Up
30	17.16s	4.5x
20	9.03s	8.5x
10	7.37s	10.5x
8	5.90s	13.1x

3D Cylinder, Fine Mesh (10k Rays)
6414 faces, Exhaustive Search: 9892.82s

Bin Size	Run Time	Speed Up
30	203.97s	48.5x
20	121.40s	81.5x
10	84.40s	117.2x
8	75.02s	131.9x

Table 7. Results of Tree-Based Viewfactor Code on Example Problems. Within a problem, the same set of seeds were used and the output among the different bin sizes was identical. A new set of seeds was used for each problem. The test hardware was a workstation with two 2.8 GHz Intel Xeon CPUs, each with 6 cores. `mpirun` with 12 processors was used.

As Table 7 shows, the tree-powered search performs spectacularly. To summarize:

- For small problems, speed-up is a modest 2-3x. Choosing smaller bin sizes results in higher speed up, although the absolute difference in time is small.
- For medium size problems, speed-up is more dramatic and the runtime decreases steadily as the bin size is decreased. Speed-up ranges from 5-10x.
- For large problems, the speed-up can only be described as “otherworldly,” exceeding 120x for small bin sizes. The tree reduces runtimes to the extent that problems previously requiring massively parallel resources can be run on desktop workstations in only a few minutes.

VI. Code Verification

Several problems are presented in this section to verify the implementation of the view factor and enclosure radiation routines in *CHAR*. Verification exercises demonstrate the correct implementation of the numerical algorithms used to solve the intended equations [10]. For this work, problems with simple geometries are used since analytical solutions exist to make comparisons against. First, simple flat plate problems will be used to verify the implementation of the two view factor algorithms supported in *CHAR*. Then, the implementation of the enclosure radiation methods described in previous sections are verified by solving a cylindrical enclosure problem.

A. Flat Plate Verification Problems

Classic flat plate problems are used for verification of view factors due to their simplicity, and the vast amount of work done in the open literature on these problems. This allows the authors (and the reader) to have greater confidence in the trends shown in this work by comparing to other published work. Figure 9 shows the geometry of the two types of flat plate problems considered in the following sections. Due to the non-deterministic nature of the monte carlo approach, each problem shown here was run 20 times to obtain an ensemble-averaged view factor. Furthermore, the errors shown for the monte carlo results are defined as the standard deviation divided by the mean of the 20 runs (in percent form). Results produced using the double area integral approach only had to be computed once, and the error metric is a traditional difference between the computation and the analytical value, normalized by the analytical value, and presented in percent form.

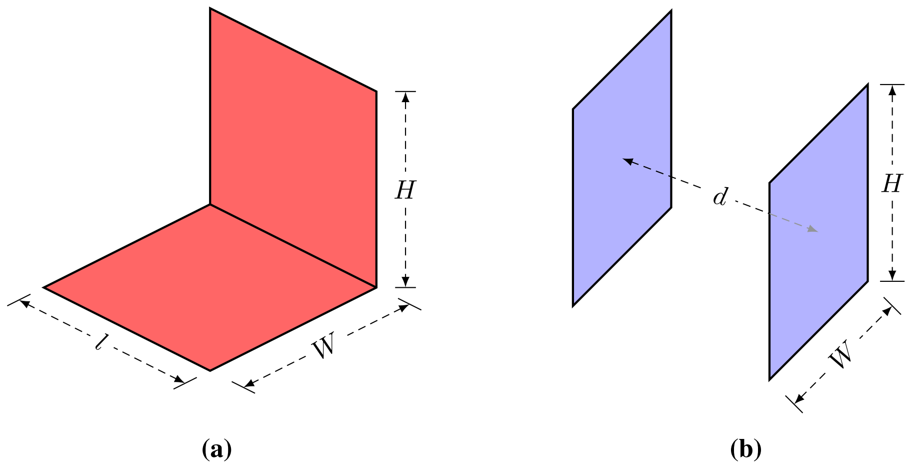


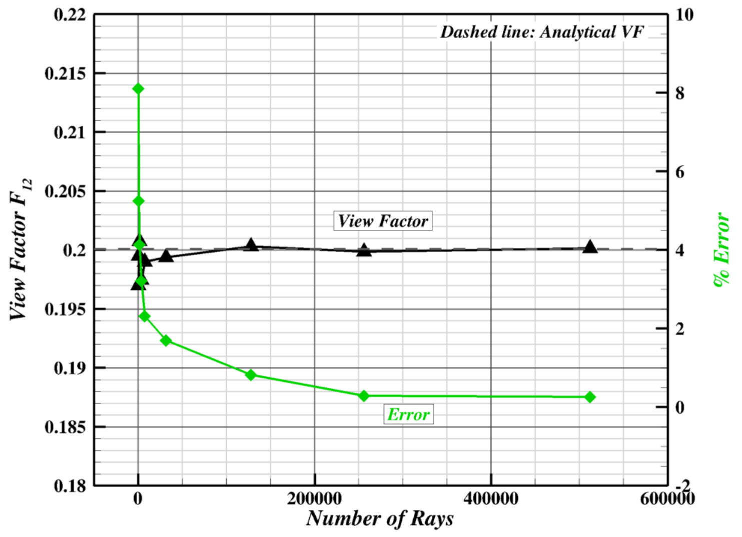
Figure 9. Geometry definitions for orthogonal plates (a) and parallel plates (b) test cases.

1. *Orthogonal Plates*

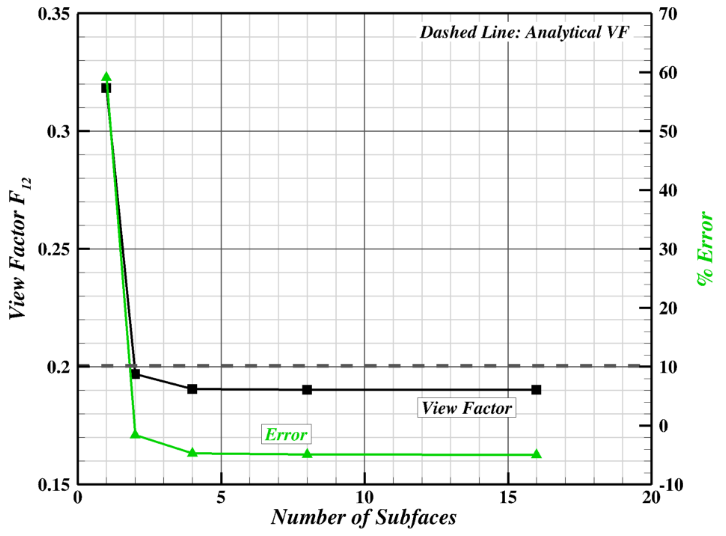
The first view factor verification problem consists of two flat plates, which share a common edge, and are orthogonal to each other as shown in Figure 9. The well-known analytical view factor for this configuration is given by [11],

$$F_{12} = \frac{1}{W\pi}\Bigg[W\tan^{-1}\frac{1}{W} + H\tan^{-1}\frac{1}{H} - \sqrt{H^2+W^2}\tan^{-1}\sqrt{\frac{1}{H^2+W^2}} + \frac{1}{4}\ln\left\{\frac{(1+W^2)(1+H^2)}{1+W^2+H^2}\left[\frac{W^2(1+W^2+H^2)}{(1+W^2)(W^2+H^2)}\right]^{W^2}\left[\frac{H^2(1+W^2+H^2)}{(1+H^2)(W^2+H^2)}\right]^{H^2}\right\}\Bigg] \tag{70}$$

where H and W are defined as $H = h/l$ and $W = w/l$. Both methods available in *CHAR*, double area integral and monte carlo method, were used to compute the view factors for this configuration. The number of subfaces and number of rays cast for each of the approaches was varied to highlight the accuracy and convergence of each approach to the analytical value. For this case, all dimensions were assumed to be equal, and therefore $H/l = W/l = 1$. The results are shown in Figure 10. The symbols in each figure represent the cases which were computed using *CHAR*, and the dashed line highlights the analytical view factor value of 0.200044 in each plot.



(a)



(b)

Figure 10. View factors computed for orthogonal plates problem using monte carlo approach (a) and double area integral method (b).

The view factor computations using the monte carlo approach show the familiar $\sqrt{N}$ decrease in the error a given by Eq. (43) for a given view factor and confidence level. After approximately 250,000 rays, the resulting view factor remains practically unchanged, with the smallest error of 0.25% resulting from casting 512,000 rays from each face. The double area integral results also show a trend in which the answer gets better as each face is subdivided into a larger number of subfaces. However, the view factors converge towards a value of 0.19013, which results in an error of approximately 5%. This larger error associated with the double area integral is expected for this particular problem. As explained in Section IV, the assumptions behind that approach hold best for small faces which are far apart. For this scenario, the subfaces closer to the common edge between the two plates have the largest contribution to this error.

2. *Parallel Plates*

In order to assess the implementation and performance of the view factor methods in *CHAR*, an additional configuration is considered in which two finite plates are parallel to each other (see Figure 9). The analytical view factor for this scenario can be computed as follows [11],

$$\begin{aligned} F_{12} = \frac{2}{\pi XY}\Bigg\{ \ln\left[\frac{\left(1+X^2\right)\left(1+Y^2\right)}{1+X^2+Y^2}\right]^{1/2} &+ X\sqrt{1+Y^2}\tan^{-1}\frac{X}{\sqrt{1+Y^2}} \\ &+ Y\sqrt{1+X^2}\tan^{-1}\frac{Y}{\sqrt{1+X^2}} - X\tan^{-1}X - Y\tan^{-1}Y \Bigg\} \end{aligned} \tag{71}$$

where $X = H/d$ and $Y = W/d$ according to Figure 9. Two different separation distances (d) are considered in this section. The results assuming a separation distance equal to the width (W) and height (H) of both plates are shown in Figure 11. The analytical view factor of 0.19982 is also shown in the plots. Similar to the orthogonal plates, the view factors for this particular case are better resolved by the monte carlo approach. Even with a relatively small number of rays, the result is more accurate than subdividing into many subfaces with the double area integral approach. The DAI view factor reaches a value of approximately 0.239 with many subfaces, whereas the error from the monte carlo approach reaches 0.24% for 512,000 rays cast. Additionally, the theoretical $\sqrt{N}$ error convergence is seen again for the monte carlo approach.

The large error seen in the previous two cases for the DAI approach highlights the types of errors which can result if the user is not careful in selecting an appropriate view factor approach for the problem at hand. To test a scenario which is better suited for computing view factors using DAI, a second parallel plates problem was solved but with increased distance between the plates. Letting the distance between the plates be 10 times the width of each plate, the analytical view factor value is 0.003162. The results obtained using both approaches in *CHAR* are shown in Figure 12. For this case, even without subdividing the faces into any further subfaces, the error is within 1%. Subdividing further, the DAI error is further reduced to values of 0.3%. This highlights the good performance of this approach when used in appropriate scenarios. On the other hand, the error using the MC approach is very high when only 500 rays are cast (90% error). Once again, the error decreases with the expected trend as more rays are used. Still, with 512,000 rays, the error is as high as 2.3%. This error could be reduced futher by continuing to shoot more rays from each face, but would result in a large increase in computational effort to resolve this view factor further. For this particular application, the view factor is better resolved using the DAI approach with relatively small computational expense.

In the end, the results shown for these simple flat plate problems verify the implementation of these two view factor methods in *CHAR*. The convergence and accuracy for each method on each test case considered are consistent with expectation and similar results seen in the literature [12][3]. The different cases considered also highlight the importance for the user to understand the applicability of each view factor approach in order to make an appropriate selection.

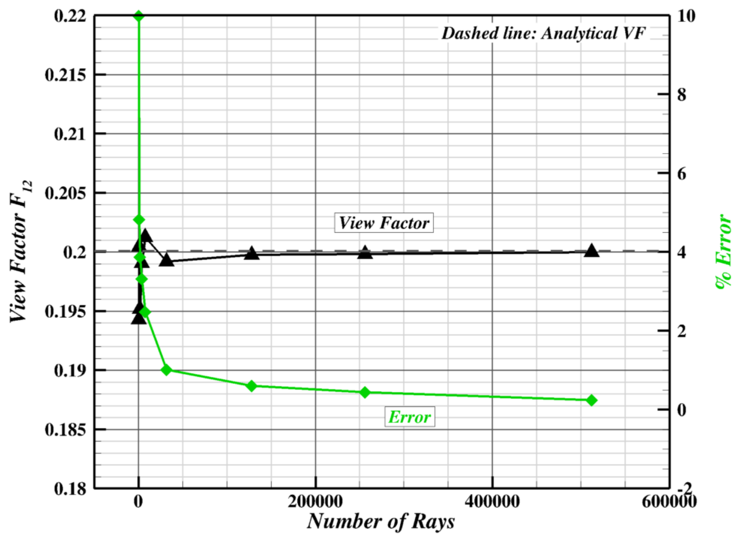
B. Cylindrical Enclosure Problem

The work by Blackwell, Dowding and Modest [13] presents a manufactured solution for the enclosure radiation in a cylindrical cavity by solving the integral form of the radiosity equation Eq. (12), given by

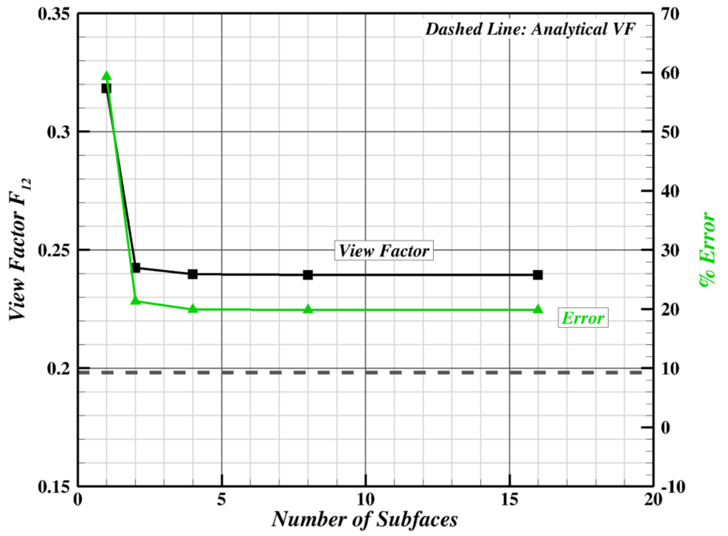
$$J_i(\mathbf{r}_i) = \epsilon_i \sigma T_i^4(\mathbf{r}_i) + (1-\epsilon_i)\sum_{j=1}^{N}\int_{A_j} J_j(\mathbf{r}_j)K(\mathbf{r}_i,\mathbf{r}_j)\mathrm{d}A_j \tag{72}$$

where K is defined as

$$K(\mathbf{r}_i,\mathbf{r}_j) = \frac{\mathrm{d}F_{ij}}{\mathrm{d}A_j} = \frac{\mathrm{d}F_{ji}}{\mathrm{d}A_i} \tag{73}$$

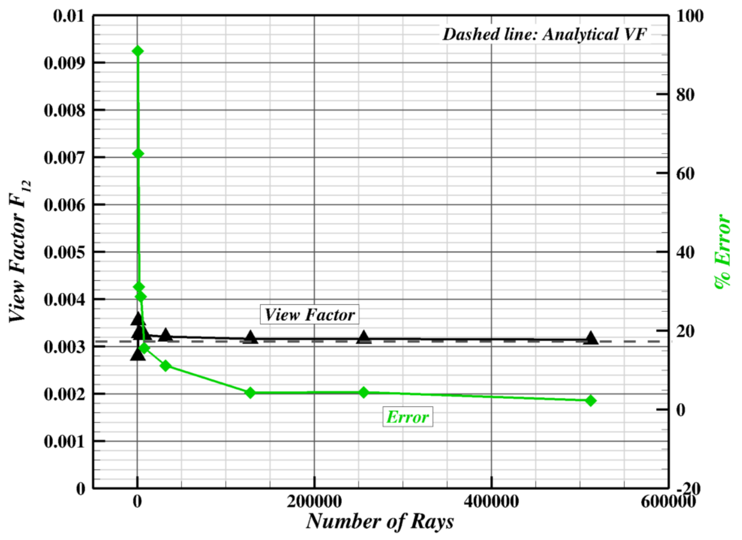


(a)

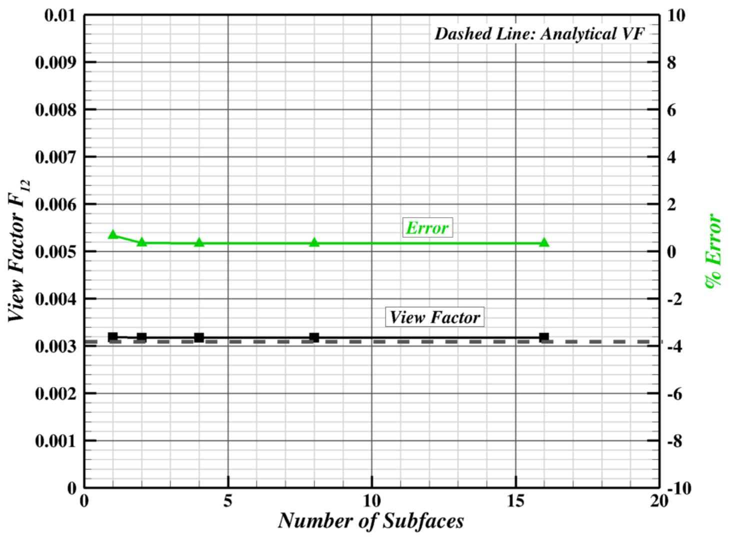


(b)

Figure 11. View factors computed for parallel plates problem ($X = Y = 1$) using monte carlo approach (a) and double area integral method (b).



(a)



(b)

Figure 12. View factors computed for parallel plates problem ($X = Y = 0.1$) using monte carlo approach (a) and double area integral method (b).

and the geometric view factor between the two infinitesimally small faces (i and j) that are a distance S apart is

$$\mathrm{d}A_i\mathrm{d}F_{ij} = \mathrm{d}A_j\mathrm{d}F_{ji} = \frac{\cos\theta_i \cos\theta_j}{\pi S^2} \tag{74}$$

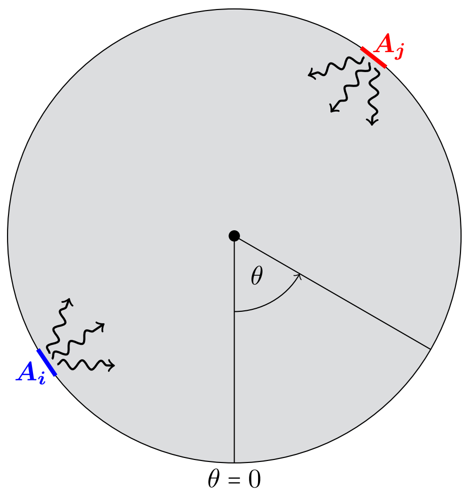


Figure 13. Angle definition for cylindrical cavity verification problem.

The details of the derivation are not presented here since they are described in great detail in [13]. However, the final expressions and steps used to carry out the verification of the enclosure radiation will be described. The analytical solution to this radiating cavity problem is used to verify the order of accuracy of the discrete radiosity equations implemented in *CHAR* (Eq. (23)). This problem was selected to carry out the present verification mainly because it allows the analyst to verify the enclosure radiation terms independent of any solid conduction in the problem. The geometry and angle definitions are shown in Figure 13. The angle (θ) is defined from the south pole of the cavity, and defined as positive as it travels counter-clockwise. Through the derivation in [13], the temperature at every point on the cavity is given by

$$T = T_o \left(\cos\left(\theta/2\right)\right)^{1/4} \tag{75}$$

Finally, the non-dimensional radiosity (J), irradiation (G), and flux (q) are simply functions of θ and are given by

$$J' = \frac{J}{E_o} = \frac{\sqrt{\epsilon}}{\sin\left(\pi\sqrt{\epsilon}/2\right)} \cos\left(\theta\sqrt{\epsilon}/2\right) \tag{76}$$

$$G' = \frac{G}{E_o} = \frac{\sqrt{\epsilon}}{1-\epsilon}\left[\frac{\cos\left(\theta\sqrt{\epsilon}/2\right)}{\sin\left(\pi\sqrt{\epsilon}/2\right)} - \sqrt{\epsilon}\cos(\theta/2)\right] \tag{77}$$

$$q' = \frac{q}{E_o} = \frac{\epsilon}{1-\epsilon}\left[\cos(\theta/2) - \sqrt{\epsilon}\frac{\cos\left(\theta\sqrt{\epsilon}/2\right)}{\sin\left(\pi\sqrt{\epsilon}/2\right)}\right] \tag{78}$$

and the blackbody emissive power at temperature T_o is $E_o = \sigma T_o^4$. The distributions are shown in Figure [ADD FIGURE] for half of the cylindrical cavity since these are symmetric across the $\theta = 0$ centerline. Because the temperature is largest near the bottom of the cavity, the radiosity from faces near this region are largest, and therefore results in positive fluxes out of these elements. On the other hand, the negative fluxes on the regions near the top of the cavity indicate that the irradiation (G) is larger than the radiosity (J) from these faces.

For the present verification study, three different grids of varying element densities were created, and are described in Table[ADD TABLE WITH DELs AND Ne]. As noted in [13], exact view factors for this configuration can be obtained using the crossed-strings method of Hottel [14]. However, since this approach is not implemented in *CHAR*, view factors were computed using the monte carlo approach with special considerations to the resulting error. By rearranging Eq. (43), the relationship between the number of rays casted and the resulting error for a given view factor and confidence interval is

$$N = \left(\frac{C}{E}\right)^2 \frac{1-F_{ij}}{F_{ij}} \tag{79}$$

Keeping the view factor error approximately constant for the different grid levels considered in this section, for a fixed confidence interval C, implies that the number of rays is directly proportional to $\frac{1-F_{ij}}{F_{ij}}$. Futhermore, for small values of F_{ij}, the right hand side behaves approximately as

$$N \approx \frac{1}{F_{ij}} \tag{80}$$

Since each refined grid level in this work is simply a factor of 2 finer than the previous grid level, Eq. (32) implies that by halving the size of two 1D faces (i and j), the resulting view factor $F_{ij}^{refined}$ will be approximately $1/4$ of the value previous to refining the grid. Combining this assumption with Eq. (80) then suggests to increase the number of rays cast by a factor of 4 each time the grid is refined by a factor of 2. This result was applied to this work in an attempt to keep the error resulting from the view factors constant, and thereby minimizing the influence of this error on the convergence of the enclosure radiation. For the coarse, medium and fine grid levels, the number of rays shot from each face when computing view factors were 50,000, 200,000, and 800,000, respectively.

The radiosity terms were solved using the radiosity matrix equation approach described in Section III. An emissivity of 0.9 was assumed, as well as $T_o = 100K$. For each grid level, the approach can be outlined as follows:

1. Compute the angle θ at the centroid of each face in the radiation domain
2. Use equation Eq. (75) to prescribe the temperature of each face
3. Solve the radiosity matrix equation in *CHAR* for the radiosities (J)
4. Compute the resulting flux at each face (i) according to the well-known expression [15]

$$q_i = \frac{\epsilon_i}{1-\epsilon_i}\left(\sigma T_i^4 - J_i\right)$$

5. Compute the analytical flux given by Eq. (78)
6. Compute the L_2 of the heat flux error

$$\text{Error} = \sum_{i=0}^{N}(q_i^{CHAR} - q_i^{analytical})^2$$

The results are shown in Figure 14 and compared to a 2nd order reference line. The results clearly appear to follow second order convergence. Even though the theoretical convergence of the enclosure radiation terms is not formally known, the authors in [13] present a reasonable argument as to why the discretization error is likely $\mathcal{O}(h^2)$ since the discrete form of the radiosity equations are derived in a manner somewhat consistent with the composite midpoint rule, which is $\mathcal{O}(h^2)$. In addition, the works in [13][16] have observed second order convergence in their implementations.

For the cylindrical geometry considered, it is also important to note the error introduced by the discretization of the shape. In this case, the circular outline is simply defined by straight-line segments according to the number of faces used in each grid level. As the number of faces are increased, the computational domain approaches the analytical circular geometry. Therefore, errors due to solving the problem on a slightly different geometry will also decrease as the grid levels are refined. However, the truncation error due to the changing geometry is $\mathcal{O}(h^5)$ ([13]), and therefore the error in the discretization of the enclosure radiation equations should be dominating in this study. In the end, the results shown herein verify the implementation of the discrete enclosure radiation equations in *CHAR*.

VII. Code-to-Code Comparison

One of the main motivations to implement some of the enclosure radiation capabilities within *CHAR* was the design of the cavity heating tiles experiment on Orion's EFT-1 mission. The cavity heating tiles were two tiles on the vehicle's backshell, each with a cylindrical cavity cut-out on the surface of the tile to mimic damage due to an MMOD impact. These cavities were designed with 1" diameters, and with depths of 1" and 1.4" [17]. Figure 15 shows the two cavity tiles installed on the Orion backshell before the EFT-1 mission. During design of these cavities, *CHAR* was used to analyze the thermal response of these tiles, which required many enclosure reradiation calculations. In these features, the fraction of heating to the tile due to energy being absorbed, emitted and reflected within the cavity walls is non-neglegible and must be accounted for. To instill confidence in *CHAR*'s ability to correctly model this

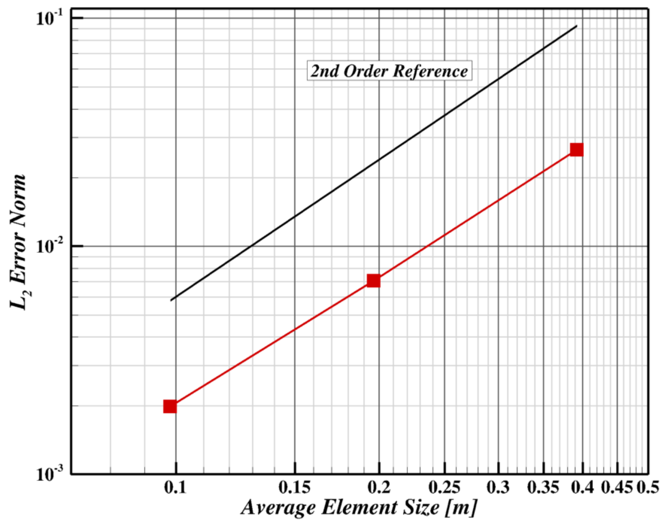


Figure 14. Grid convergence results for enclosure reradiation verification problem.

problem, a preliminary code-to-code comparison was carried out against the comercial software SINDA/FLUINT 5.6 Patch Level 9 [18], which has been heavily used in the past within NASA for tile thermal analysis. For this problem SINDA was used thorugh the Thermal Desktop (TD) interface, and radiation calculations were performed using the RadCAD module available within SINDA.

The problem was modeled individually by two different designers, each using his own best practices for the thermal software being run. The problem involved a realistic time- and space- dependent aeroheating boundary condition on a half-symmetry domain of the 1.4”-deep cavity tile, which was allowed to reradiate to a constant far-field temperature. The sidewalls of the domain, including the symmetry plane were modeled as adiabatic, and the (inner) backwall of the tile structure was also set as adiabatic. On the inner cavity surfaces, in addition to the specified heating distribution within the cavity, an enclosure reradiation boundary condition was set, with any view factor to space reradiating to a constant sink temperature.

Despite several attempts to import the finite element mesh created for the *CHAR* analysis into Thermal Desktop, the number of elements in this grid (~315,000) was too many to comfortably handle in Thermal Desktop. Therefore, a separate mesh was developed for the SINDA/TD analysis which consisted of 30,000 nodes. Both grids employed tighter element clustering near the surface of the tile and the cavity walls, tight clustering underneath the cavity floor, while keeping the surface elments on the cavity enclosure as large as possible to ensure convergence when computing radiation view factors. The gridded geometry included the tile, reaction cured glass coating (RCG) on the surface of the tile, and several substructure layers. Both, the *CHAR* and SINDA grids used for this analysis are shown in Figure 16.

A Monte Carlo ray tracing approach was used with both solvers to compute geometric view factors. For the SINDA model, 5 million rays were cast per node to achieve convergence, while the *CHAR* results for this problem were converged after shooting 1 million rays per element. Also, to closely mimic the numerics in SINDA/TD, *CHAR* was run assuming a trapezoidal integration rule and linear pressure interpolation in conductivity table lookups.

Comparisons were made at the ten different locations shown in Figure 17 and the comparisons between the two codes are presented in Figure 18. The temperatures presented have been normalized by the maximum temperature predicted by the *CHAR* simulation at each location. Considering differences in grids and the numerics of each code, the results at all ten locations show excellent agreement between the two solvers. Even though further work could isolate grid and numerics differences in this comparison, the close agreement between the two codes serves as additional evidence to verify the implementation of the enclosure reradiation boundary condition in *CHAR*.

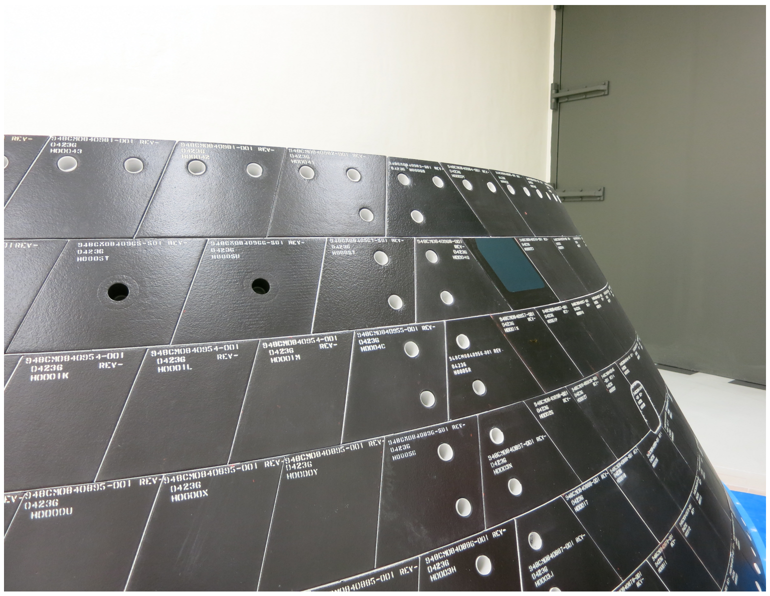
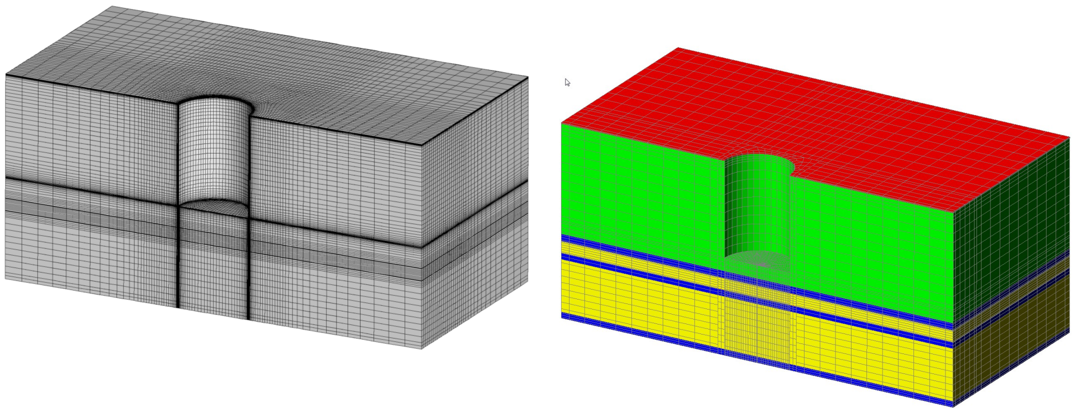


Figure 15. Orion EFT-1 cavity heating tiles. [17]



(a) (b)

Figure 16. Grids used for code-to-code cavity problem for (a) CHAR simulations and (b) TD/SINDA simulations.

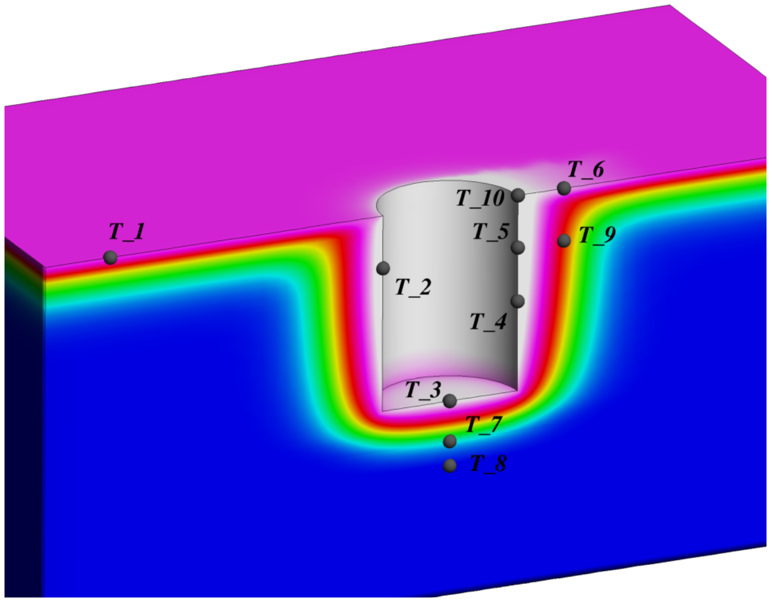


Figure 17. CHAR vs. SINDA/RADCAD comparison locations.

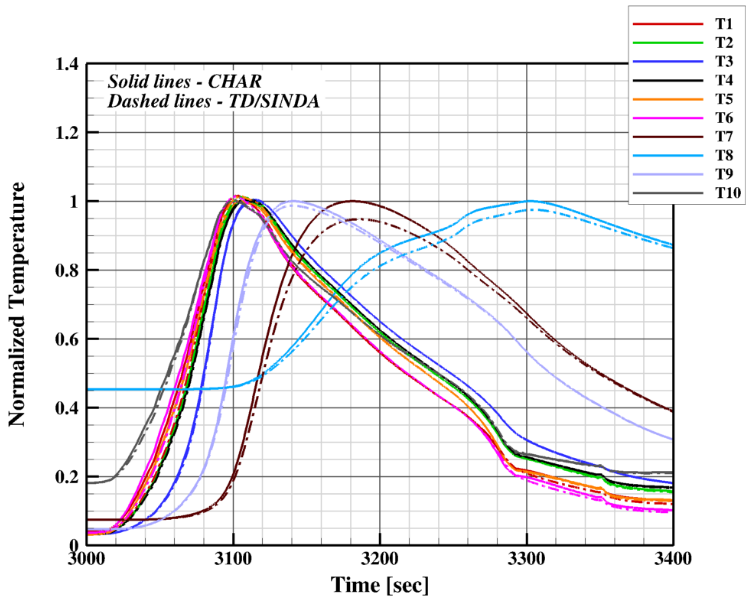


Figure 18. CHAR vs. SINDA/RADCAD comparison results.

VIII. Ablating Cavity Demonstration Case

This section presents a demonstration problem in which the enclosure radiation capabilities are applied to a more realistic thermal protection system (TPS) problem. The problem considered is a two dimensional cavity with a rectangular cross-section in an ablative TPS material. The ablative material used for this test case is TACOT [19]. A 1.5"-thick piece of TACOT with a 1.0"-deep and 0.25"-wide cavity is considered. A deep and narrow cavity was modeled in order to stress the enclosure radiation logic for this demonstration. The grid generated for this problem, shown in Figure 19, contains 8600 quadrilateral elements.

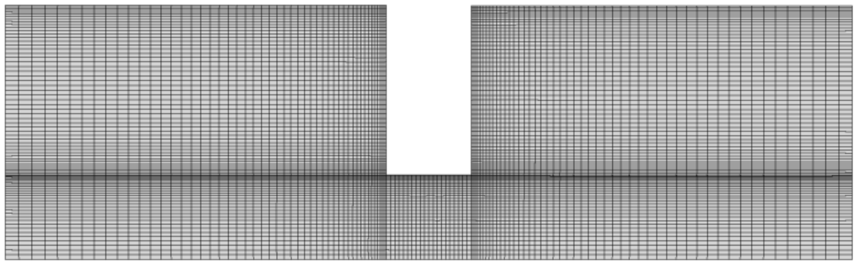


Figure 19. Mesh used for 2D ablating cavity problem.

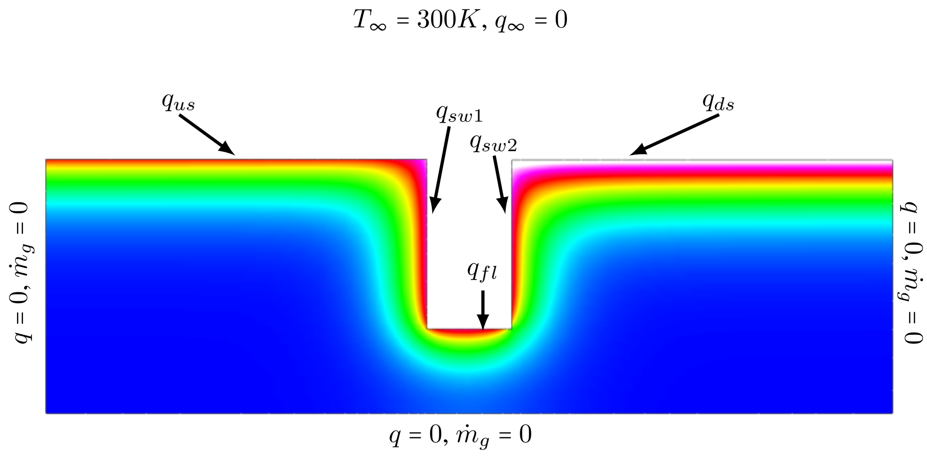


Figure 20. Nomenclature and distribution of heating conditions applied on the 2D cavity.

Figure 20 shows the different heating distributions applied around the geometry. Different heating levels were applied on the upstream (q_{us}) surface, the downstream (q_{ds}) surface, the upstream and downstream sidewalls (q_{sw1} and q_{sw2} respectively), and the cavity floor (q_{fl}). In addition, the bottom and outer sidewalls of the geometry were set as adiabatic and impermeable walls. The aeroheating was applied using a film coefficient and recovery enthalpy boundary condition, and the cold wall heating at the upstream location is shown in Figure 21. The environment at the other locations were simply scaled versions of the upstream heating. The same environment as the upstream (baseline) was used at the cavity floor and cavity sidewalls for this exercise. The only exception was the heating at the downstream surface (q_{ds}) which was scaled by a factor of 2.5 to exacerbate the heating and recession on the downstream edge of the cavity. These heating distributions were selected simply to stress the recession and mesh motion in this problem, and are not intended to be physcially accurate.

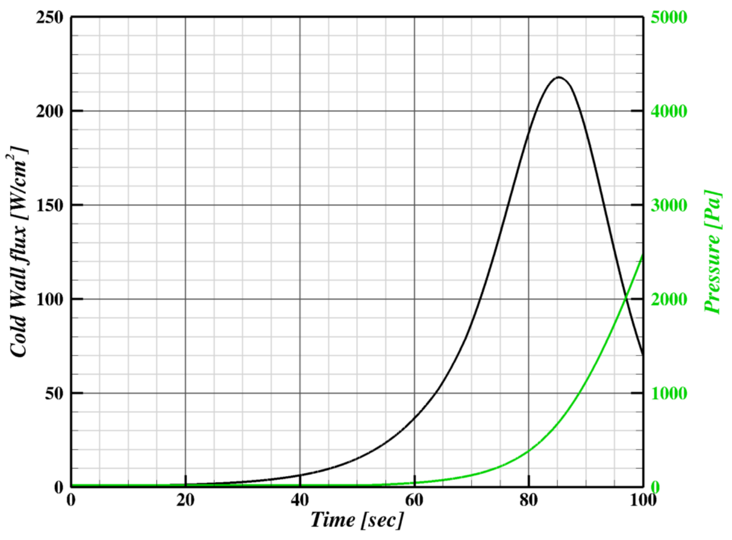
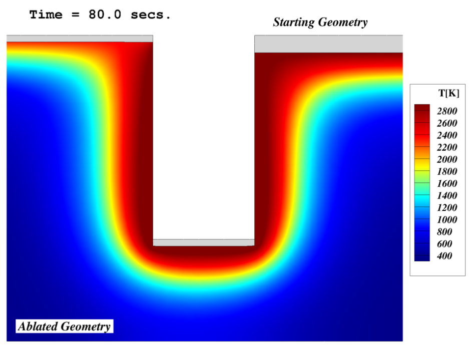


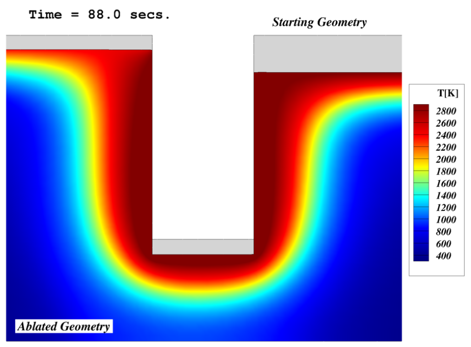
Figure 21. Baseline cold wall flux and pressure applied for the 2D cavity demonstration problem.

The problem was run for the 100-second heat pulse shown in Figure 21, and view factors were recomputed in *CHAR* every 5 time steps using the Monte Carlo ray casting approach. Since a varying time step with a maximum value of 0.2 was used, this resulted in view factors being recomputed every 1 second or less. Currently the user has the option to select the time step frequency at which to recompute view factors. On-going developments in *CHAR* will allow the user other alternatives to select when to recompute view factors.

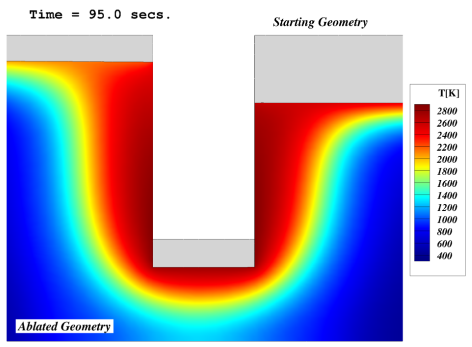
The resulting geometry and temperature contours at four different times are presented in Figure 22. The initial starting geometry is also shown in the background of each plot to better illustrate the amount of recession seen in this test case. As expected, the hottest temperatures are seen near the downstream corner of the cavity due to the environment assumptions applied. Most of the recession begins near the top of the heat pulse, and it continues decomposing and ablating until the end of the heat pulse. The aggressive environment applied inside of the cavity also resulted in the cavity floor ablating. The sidwalls were only allowed to slide in this problem to preserve the quality of the mesh as long as possible.



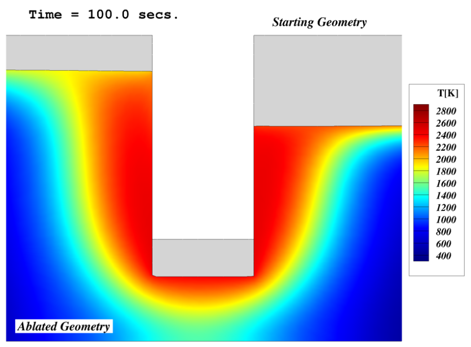
(a)



(b)



(c)



(d)

Figure 22. Temperature contours and changes in geometry due to ablation of 2D cavity.

Close examination of how each view factor is changing as the mesh moves is difficult to achieve since there is a view factor between every pair of faces within the cavity. Instead, to qualitatively assess the effect of recomputing view factors as the mesh deforms due to recession, the view factors to space from each face are presented at 3 different times in Figure 23. From these results, it is clear that the view factor to space changed significantly as the TPS ablated. The most significant change is seen near the top edges of the cavity, where the view factors to space initially are 0.5 at the outermost element. However, these values change to 0.73 and 00.27 by the end of the problem. The view to space at the upstream edge of the cavity increases as the downstream edge ablates, and similarly, the view to space at the downstream edge decreases significantly because the upstream edge does not recede as much. In the end, this problem highlights the importance of recomputing accurate view factors as a mesh deforms, which is typical in ablative TPS-type problems. Finally, this problem also demonstrates the capabilities in *CHAR* to account for radiation effects within enclosures for problems involving moving meshes.

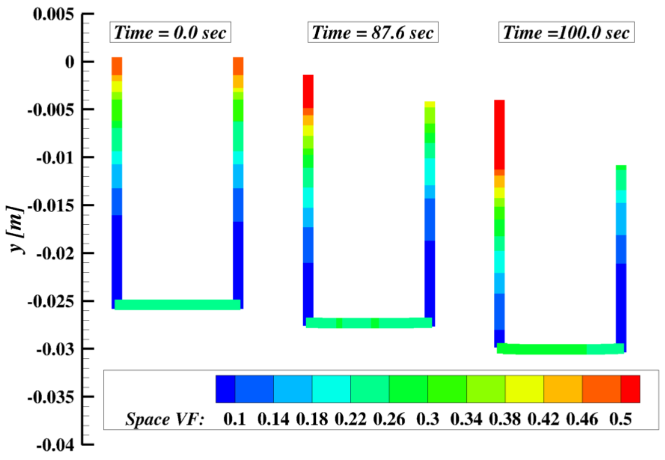


Figure 23. Changes in view factor to space at cavity wall locations due to changes in geometry.

IX. Summary and Conclusions

The enclosure radiation capabilities implemented to-date in *CHAR* have been described in detail. Verification of the enclosure radiation equations was accomplished using an infinite cylindrical cavity problem, and second order convergence is demonstrated. In addition, implementations of the double area integral and Monte Carlo approaches for computing geometric view factors were also verified by considering simple flat plate problems for which analytical values are well-known. Strengths and weaknesses for each view factor approach are described in order to highlight the important considerations when solving an enclosure radiation problem. The implementation of an octree structure to speed up the computation of view factors using the Monte Carlo approach is presented. Complete details on populating the tree structure and efficiently traversing the tree are provided. By considering several test problems, the tree-powered search results in speed-ups of up to 130 times as compared to a naive exhaustive search approach.

The modeling of surface-to-surface radiation exchange is also demonstrated for more realistic problems of interest to the broader TPS community. Results of a code-to-code comparison modeling one of the Orion Exploration Flight Test 1 cavity tile experiments show excellent agreement between *CHAR* and SINDA/Thermal Desktop. Furthermore, the code's ability to account for the radiation exchange in a rapidly changing geometry is demonstrated by considering an ablating 2D cavity. By leveraging the thermal modeling capabilities already available in *CHAR*, the implementation of the enclosure radiation boundary condition presented in this work enables users to solve real-world, production-level TPS problems involving complex geometries.

Acknowledgments

The authors would like to thank Dr. Ben Blackwell from Sandia National Laboratories for several helpful discussions on enclosure radiation modeling and verification problems. The contributions by Hee Jong Song (NASA JSC) in carrying out the thermal modeling in SINDA/Thermal Desktop for the cavity tile code-to-code comparisons are greatly appreciated. Finally, Dr. Benjamin Kirk (NASA JSC, Branch Chief) for his continued support in the development of *CHAR* and in the work presented herein.

References

[1] Amar, A. J., Oliver, A. B., Kirk, B. S., Salazar, G., and Droba, J., "Overview of the CHarring Ablator Response (CHAR) Code," 46th AIAA Thermophysics Conference, Submitted for publication.

[2] Balay, S., Buschelman, K., Eijkhout, V., Gropp, W. D., Kaushik, D., Knepley, M. G., McInnes, L. C., Smith, B. F., and Zhang, H., "PETSc Users Manual," Tech. Rep. ANL-95/11 - Revision 2.3.0, Argonne National Laboratory, April 2004.

[3] A.B, S., "FACET - A Radiation View Factor Computer Code for Axisymmetric, 2D Planar and 3D Geometries with Shadowing," Tech. Rep. UCID-19887, Lawrence Livermore National Laboratory, August 1983.

[4] Osada, R., Funkhouser, T., Chazelle, B., and Dobkin, D., "Shape Distributions," ACM Transactions on Graphics, Vol. 21, 2002, pp. 807–832.

[5] Katzgraber, H. G., "Random Numbers in Scientific Computing: An Introduction," http://arxiv.org/abs/1005.4117v1, 2010, [Online; posted 22-May-2010; accessed 2014].

[6] Baranoski, G., Rokne, J. G., and Xu, G., "Applying the exponential Chebyshev inequality to the nondeterministic computation of form factors," Journal of Quantitative Spectroscopy & Radiative Transfer, Vol. 69, 2001, pp. 447–467.

[7] Daun, K. J., Morton, D. P., and Howell, J. R., "Smoothing Monte Carlo Exchange Factors through Constrained Maximum Likelihood Estimation," 2005.

[8] Revelles, J., Ureña, C., and Lastra, M., "An Efficient Parametric Algorithm for Octree Traversal," Preprint.

[9] Ericson, C., Real-Time Collision Detection, The Morgan Kauffmann Series in Interactive 3-D Technology, Elsevier, 2005.

[10] Trucano, T. G., Pilch, M., and Oberkampf, W. L., "On the Role of Code Comparisons in Verification and Validation," Tech. Rep. SAND2003-2752, Sandia National Laboratories, August 2003.

[11] Howell, J. R., "A Catalog of Radiation Heat Transfer Configuration Factors," http://www.thermalradiation.net, 2015, [Online; posted 2015; accessed 30-March-2016].

[12] Mazumder, S. and Ravishankar, M., "General Procedure for Calculation of Diffuse View Factors Between Arbitrary Planar Polygons," International Journal of Heat and Mass Transfer, Vol. 55, 2012, pp. 7330–7335.

[13] Blackwell, B., Dowding, K., and Modest, M., "Cylindrical Geometry Verification Problem for Enclosure Radiation," Journal of Thermophysics and Heat Transfer, Vol. 23, 2009, pp. 711–715.

[14] Hottel, H. C., "Radiant Heat Transmission," Heat Transmission, edited by W. McAdams, chap. 4, McGraw-Hill, 3rd ed., 1954.

[15] Howell, J. R. and Siegel, R., "Thermal Radiation Heat Transfer," Tech. Rep. NASA SP-164, National Aeronautics and Space Administration, July 1969.

[16] Silva III, H. and Carnes, B., "Fully Two-Dimensional Verification Problem for Coupled Heat Conduction and Enclosure Radiation," Journal of Thermophysics and Heat Transfer, 2015, [Online; posted 30-Dec-2015; accessed 01-April-2016].

[17] Kremer, K., "Heat Protecting Back Shell Tiles Installed on NASA's Orion EFT-1 Spacecraft Set for Dec. 2014 Launch," http://www.universetoday.com/114064/heat-protecting-back-shell-tiles-installed-on-nasas-orion-eft-1-spacecraft-set-for-dec-2014-launch/, August 2014, [Online; posted 30-August-2014].

[18] "Sinda/Fluint, General Purpose Thermal/Fluid Analyzer, User's Manual," April 2013.

[19] Lachaud, J., Martin, A., Cozmuta, I., and Laub, B., "Ablation test-case series 1," 4th AFOSR/SNL/NASA Ablation Workshop, 2010.