Diffusion Discretization Schemes in Augustus: A New Hexahedral Symmetric Support Operator Method

Michael L. Hall
Jim E. Morel
Transport Methods Group
Unstructured Mesh Radiation Transport Team
Los Alamos National Laboratory
Email: **hall@lanl.gov**

X-Division Work In Progress Presentation
7 / 29 / 98

Available on-line at
http://www.lanl.gov/Augustus/

Outline

- Augustus Background
  - Equation Set
  - Mesh Description
  - Method Overview
  - References
- Hexahedral Support Operator Method Derivation
  - Overview
  - Main Derivation
  - Symmetry and Positive Definiteness
  - Comparison of MH and SO
- Support Operator Method Properties
- Matrix Solution
  - Second Order Demonstration
  - Solution Time Comparison
    * Marshak Wave Problem
    * Two Material Problem
  - Solution Scalability
- Conclusions

Augustus: Diffusion (P_1) Equation Set

$$\alpha\frac{\partial\Phi}{\partial t} - \overrightarrow{\nabla}\cdot D\overrightarrow{\nabla}\Phi + \overrightarrow{\nabla}\cdot\overrightarrow{J} + \sigma\Phi = S$$

Which can be written

$$\alpha\frac{\partial\Phi}{\partial t} + \overrightarrow{\nabla}\cdot\overrightarrow{F} + \sigma\Phi = S$$

$$\overrightarrow{F} = -D\overrightarrow{\nabla}\Phi + \overrightarrow{J}$$

Where

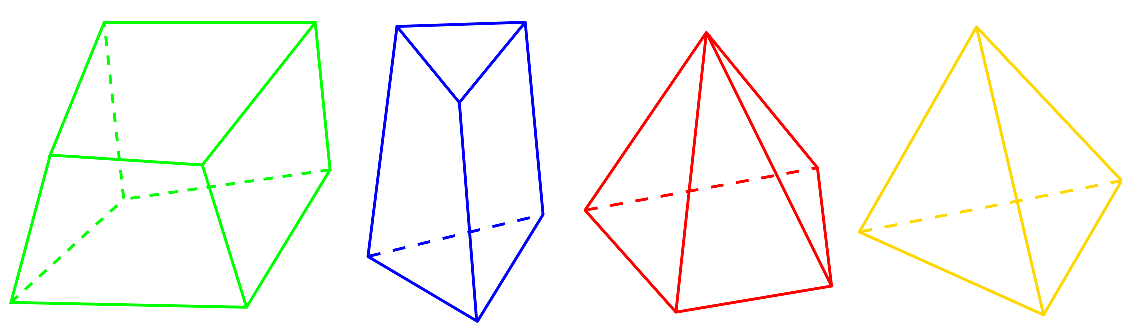
Φ	$=$	Intensity
$\overrightarrow{F}$	$=$	Flux
D	$=$	Diffusion Coefficient
α	$=$	Time Derivative Coefficient
σ	$=$	Removal Coefficient
S	$=$	Intensity Source Term
$\overrightarrow{J}$	$=$	Flux Source Term

Augustus Mesh Description

Multi-Dimensional Mesh:

Dimension	Geometries	Type of Elements
1-D	spherical, cylindrical or cartesian	line segments
2-D	cylindrical or cartesian	quadrilaterals or triangles
3-D	cartesian	hexahedra or degenerate hexahedra (tetrahedra, prisms, pyramids)

all with an unstructured (arbitrarily connected) format.



This presentation will assume a 3-D mesh.

Augustus Method Overview

- Spatial Discretization
  - Morel-Hall asymmetric diffusion discretization
  - Support Operator symmetric diffusion discretization
- Temporal Discretization
  - Backwards Euler implicit discretization
- Matrix Solution
  - Krylov Subspace Iterative Methods
    * JTpack: GMRES, BCGS, TFQMR, CG
    * Preconditioners:
      · JTpack: Jacobi, SSOR, ILU
      · Low-order version of Morel-Hall or Support Operator discretization that is a smaller, symmetric system and is solved by CG with SSOR (from JTpack)
  - Incomplete Direct Method - UMFPACK
- Augustus is used as the diffusion kernel for the Spartan SP$_N$ package and the Magnum MHD package

Diffusion Discretization References

- Morel-Hall Asymmetric Method
  - Described in

    Michael L. Hall, and Jim E. Morel. A Second-Order Cell-Centered Diffusion Differencing Scheme for Unstructured Hexahedral Lagrangian Meshes. In *Proceedings of the 1996 Nuclear Explosives Code Developers Conference (NECDC), UCRL-MI-124790*, pages 359–375, San Diego, CA, October 21–25 1996. LA-UR-97-8.

    which is an extension of

    J. E. Morel, J. E. Dendy, Jr., Michael L. Hall, and Stephen W. White. A Cell-Centered Lagrangian-Mesh Diffusion Differencing Scheme. Journal of Computational Physics, 103(2):286-299, December 1992.

    to 3-D unstructured meshes, with an alternate derivation.
- Support Operator Symmetric Method:
  - Extension of the method described in

    Mikhail Shashkov and Stanly Steinberg. Solving Diffusion Equations with Rough Coefficients in Rough Grids. Journal of Computational Physics, 129:383-405, 1996.

    to 3-D unstructured meshes, with an alternate derivation.

Support Operator Method Derivation: Outline

The Support Operator Method for Diffusion on Hexahedra:

- Represent the diffusion term ($\overrightarrow{\nabla} \cdot D \overrightarrow{\nabla} \Phi$) as the divergence ($\overrightarrow{\nabla} \cdot$) of a gradient ($\overrightarrow{\nabla}$)
- Explicitly define one of the operators (in this case, the divergence operator)
- Define the remaining operator (in this case, the gradient operator) as the discrete adjoint of the first operator
- The previous step is accomplished by discretizing a portion of a vector identity

In other words, the first operator is set up explicitly, and the second operator is defined in terms of the first operator's definition.

Support Operator Method Derivation

Starting with a vector identity,

$$\overrightarrow{\nabla} \cdot \left(\phi \overrightarrow{W} \right) = \phi \, \overrightarrow{\nabla} \cdot \overrightarrow{W} + \overrightarrow{W} \cdot \overrightarrow{\nabla} \phi \, ,$$

where ϕ is the scalar variable to be diffused and $\overrightarrow{W}$ is an arbitrary vector, integrate over a cell volume:

$$\int_c \overrightarrow{\nabla} \cdot \left(\phi \overrightarrow{W} \right) \, dV = \int_c \phi \, \overrightarrow{\nabla} \cdot \overrightarrow{W} \, dV + \int_c \overrightarrow{W} \cdot \overrightarrow{\nabla} \phi \, dV \, .$$

Each colored term in the equation above will be treated separately.

Aside: note that, if inner products for scalars and vectors are defined by

$$\left\langle a, b \right\rangle = \int_c ab \, dV \text{ and } \left\langle \overrightarrow{A}, \overrightarrow{B} \right\rangle = \int_c \overrightarrow{A} \cdot \overrightarrow{B} \, dV \, ,$$

and if $\phi = 0$ on the boundary, such that the Green term vanishes, then this equation becomes the definition of an adjoint,

$$\left\langle - \overrightarrow{\nabla} \cdot \overrightarrow{W}, \phi \right\rangle = \left\langle \overrightarrow{W}, \overrightarrow{\nabla} \phi \right\rangle \, ,$$

which shows that the divergence is the negative adjoint of the gradient.

Support Operator Method Derivation

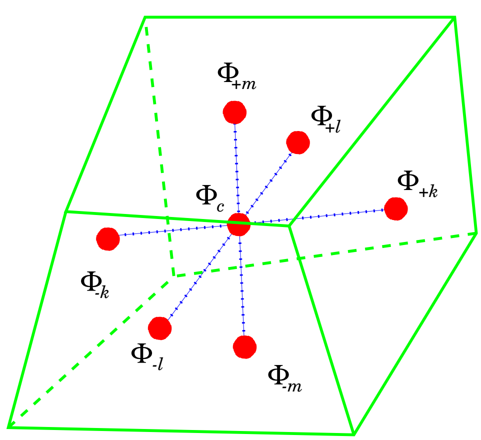
The Green term can be transformed via Gauss's Theorem into a surface integral,

$$\int_c \overrightarrow{\nabla} \cdot \left(\phi \overrightarrow{W} \right) \, dV = \oint_S \left(\phi \overrightarrow{W} \right) \cdot \overrightarrow{dA} \, .$$

This is discretized into values defined on each face of the hexahedral cell,

$$\oint_S \left(\phi \overrightarrow{W} \right) \cdot \overrightarrow{dA} \approx \sum_f \phi_f \overrightarrow{W_f} \cdot \overrightarrow{A_f} \, .$$

The summation over faces ($\sum_f$) includes six faces ($+\boldsymbol{k}$, $-\boldsymbol{k}$, $+\boldsymbol{l}$, $-\boldsymbol{l}$, $+\boldsymbol{m}$, $-\boldsymbol{m}$), shown here for the intensity variable ϕ:



Support Operator Method Derivation

The Red term is approximated by first assuming that ϕ is constant over the cell (at the center value), and then performing a discretization similar to the previous one for the Green term:

$$\begin{aligned}
\int_c \phi \overrightarrow{\nabla} \cdot \overrightarrow{W}\, dV &\approx \phi_c \int_c \overrightarrow{\nabla} \cdot \overrightarrow{W}\, dV \,, \\
&= \phi_c \oint_S \overrightarrow{W} \cdot \overrightarrow{dA} \,, \\
&\approx \phi_c \sum_f \overrightarrow{W_f} \cdot \overrightarrow{A_f} \,.
\end{aligned}$$

Support Operator Method Derivation

Turning to the final Blue term, insert the definition of the flux*,

$$\overrightarrow{F} = -D\,\overrightarrow{\nabla}\,\phi\ ,$$

to get

$$\int_c \overrightarrow{W} \cdot \overrightarrow{\nabla}\,\phi\, dV = -\int_c D^{-1}\,\overrightarrow{W} \cdot \overrightarrow{F}\, dV\ .$$

Note that by defining the flux in terms of the remainder of the equation, the gradient is being defined in terms of the divergence.

The Blue term is discretized by evaluating the integrand at each of the cell nodes (octants in 3-D) and summing:

$$-\int_c D^{-1}\,\overrightarrow{W} \cdot \overrightarrow{F}\, dV \approx -\sum_n D_n^{-1}\overrightarrow{W_n} \cdot \overrightarrow{F_n} V_n\ .$$

---

*the $\overrightarrow{J}$ term, which is necessary for a P_1 solver, is omitted here and is treated explicitly in the overall diffusion equation

Support Operator Method Derivation

Combining all of the discretized terms of the colored equation and changing to a linear algebra representation gives

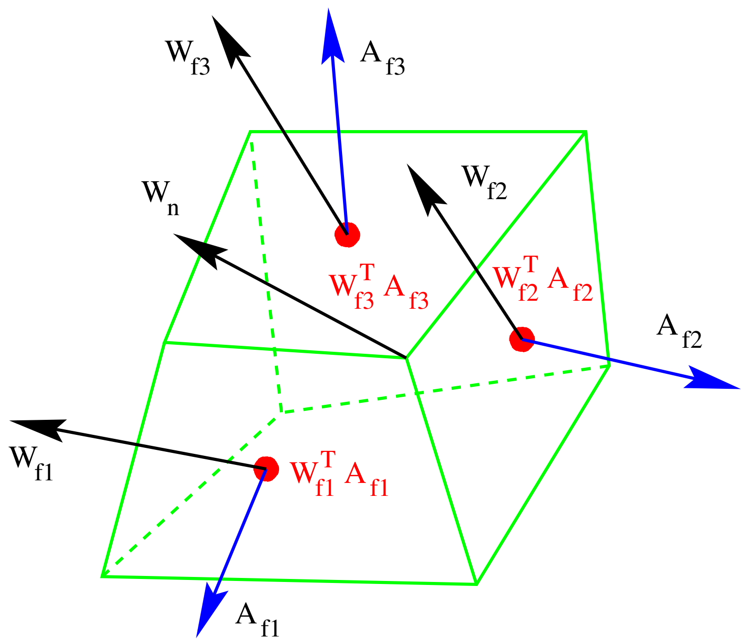
$$\sum_f \phi_f \mathbf{W}_f^{\mathrm{T}} \mathbf{A}_f = \phi_c \sum_f \mathbf{W}_f^{\mathrm{T}} \mathbf{A}_f - \sum_n D_n^{-1} \mathbf{W}_n^{\mathrm{T}} \mathbf{F}_n V_n \ .$$

Rearranging terms gives

$$\sum_n D_n^{-1} \mathbf{W}_n^{\mathrm{T}} \mathbf{F}_n V_n = \sum_f \left(\phi_c - \phi_f \right) \mathbf{W}_f^{\mathrm{T}} \mathbf{A}_f \ .$$

Note that the right hand side is a sum over the six faces, but the left hand side is a sum over the eight nodes.

Support Operator Method Derivation



In order to express the node-centered vectors, $\mathbf{W}_n$ and $\mathbf{F}_n$, in terms of their face-centered counterparts, define

$$\mathbf{J}_n^{\mathrm{T}}\mathbf{W}_n \equiv \begin{bmatrix} \mathbf{W}_{f1}^{\mathrm{T}}\mathbf{A}_{f1} \\ \mathbf{W}_{f2}^{\mathrm{T}}\mathbf{A}_{f2} \\ \mathbf{W}_{f3}^{\mathrm{T}}\mathbf{A}_{f3} \end{bmatrix} ,$$

where $f1$, $f2$, and $f3$ are the faces adjacent to node n and the Jacobian matrix is the square matrix given by

$$\mathbf{J}_n = \begin{bmatrix} \mathbf{A}_{f1} & \mathbf{A}_{f2} & \mathbf{A}_{f3} \end{bmatrix} .$$

Support Operator Method Derivation

Using this definition for the node-centered vectors $\mathbf{W}_n$ and $\mathbf{F}_n$ and performing some algebraic manipulations results in

$$\sum_n D_n^{-1} V_n \begin{bmatrix} \mathbf{W}_{f1}^{\mathrm{T}} \mathbf{A}_{f1} \\ \mathbf{W}_{f2}^{\mathrm{T}} \mathbf{A}_{f2} \\ \mathbf{W}_{f3}^{\mathrm{T}} \mathbf{A}_{f3} \end{bmatrix}^{\mathrm{T}} \mathbf{J}_n^{-1} \mathbf{J}_n^{-\mathrm{T}} \begin{bmatrix} \mathbf{F}_{f1}^{\mathrm{T}} \mathbf{A}_{f1} \\ \mathbf{F}_{f2}^{\mathrm{T}} \mathbf{A}_{f2} \\ \mathbf{F}_{f3}^{\mathrm{T}} \mathbf{A}_{f3} \end{bmatrix} = \widetilde{\mathbf{W}}^{\mathrm{T}} \tilde{\mathbf{\Phi}} \, .$$

where the sum over faces has been written as a dot product of $\widetilde{\mathbf{W}}$ and $\tilde{\mathbf{\Phi}}$, which are defined by

$$\widetilde{\mathbf{W}} = \begin{bmatrix} \mathbf{W}_1^{\mathrm{T}} \mathbf{A}_1 \\ \mathbf{W}_2^{\mathrm{T}} \mathbf{A}_2 \\ \vdots \\ \mathbf{W}_{N_{lf}}^{\mathrm{T}} \mathbf{A}_{N_{lf}} \end{bmatrix} , \quad \tilde{\mathbf{\Phi}} = \begin{bmatrix} (\phi_c - \phi_1) \\ (\phi_c - \phi_2) \\ \vdots \\ \left(\phi_c - \phi_{N_{lf}}\right) \end{bmatrix} .$$

N_{lf} is the total number of local faces, which is equal to 6 in 3-D.

Support Operator Method Derivation

To convert the short vectors involving the faces adjacent to a particular node into sparse long vectors involving all of the faces of the cell, define permutation matrices for each node, $\mathbf{P}_n$, such that

$$\begin{bmatrix} \mathbf{W}_{f1}^{\mathrm{T}}\mathbf{A}_{f1} \\ \mathbf{W}_{f2}^{\mathrm{T}}\mathbf{A}_{f2} \\ \mathbf{W}_{f3}^{\mathrm{T}}\mathbf{A}_{f3} \end{bmatrix} = \mathbf{P}_n \widetilde{\mathbf{W}} \ ,$$

where, for example,

$$\mathbf{P}_n = \begin{bmatrix} 0 & 0 & 1 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 1 & 0 \\ 0 & 1 & 0 & 0 & 0 & 0 \end{bmatrix} \qquad \begin{array}{l} \text{if } f1\,(n) = 3, \\ f2\,(n) = 5, \\ \text{and } f3\,(n) = 2. \end{array}$$

Note that $\mathbf{P}_n$ is rectangular, with a size of $N_d \times N_{lf}$ (3×6 for 3-D, 2×4 for 2-D, 1×2 for 1-D).

Support Operator Method Derivation

Using the permutation matrices, and defining $\widetilde{\mathbf{F}}$ in a fashion similar to $\widetilde{\mathbf{W}}$ ($\widetilde{\mathbf{F}}$ is a vector of $\mathbf{F}_f^{\mathrm{T}}\mathbf{A}_f$ for each cell face), gives

$$\sum_n D_n^{-1} V_n \widetilde{\mathbf{W}}^{\mathrm{T}} \mathbf{P}_n^{\mathrm{T}} \mathbf{J}_n^{-1} \mathbf{J}_n^{-\mathrm{T}} \mathbf{P}_n \widetilde{\mathbf{F}} = \widetilde{\mathbf{W}}^{\mathrm{T}} \widetilde{\mathbf{\Phi}} \ ,$$

or

$$\widetilde{\mathbf{W}}^{\mathrm{T}} \left[\sum_n D_n^{-1} V_n \mathbf{P}_n^{\mathrm{T}} \mathbf{J}_n^{-1} \mathbf{J}_n^{-\mathrm{T}} \mathbf{P}_n \right] \widetilde{\mathbf{F}} = \widetilde{\mathbf{W}}^{\mathrm{T}} \widetilde{\mathbf{\Phi}} \ ,$$

or

$$\widetilde{\mathbf{W}}^{\mathrm{T}} \mathbf{S} \widetilde{\mathbf{F}} = \widetilde{\mathbf{W}}^{\mathrm{T}} \widetilde{\mathbf{\Phi}} \ ,$$

where

$$\mathbf{S} = \sum_n D_n^{-1} V_n \mathbf{P}_n^{\mathrm{T}} \mathbf{J}_n^{-1} \mathbf{J}_n^{-\mathrm{T}} \mathbf{P}_n \ .$$

The original vector $\overrightarrow{W}$ (on which $\mathbf{W}_f$ and $\widetilde{\mathbf{W}}$ are based) was an arbitrary vector. It can now be eliminated from the equation to give

$$\mathbf{S} \widetilde{\mathbf{F}} = \widetilde{\mathbf{\Phi}} \ ,$$

which can easily be inverted to give the fluxes (dotted into the areas) in terms of the ϕ-differences, $\widetilde{\mathbf{F}} = \mathbf{S}^{-1} \widetilde{\mathbf{\Phi}}$. This is exactly the form needed for discretization of the diffusion term within Augustus.

Support Operator Method Derivation: SPD Proof

The matrix $\mathbf{S}$ is symmetric, since

$$
\begin{aligned}
\mathbf{S}^{\mathrm{T}} &= \left[\sum_n D_n^{-1} V_n \mathbf{P}_n^{\mathrm{T}} \mathbf{J}_n^{-1} \mathbf{J}_n^{-\mathrm{T}} \mathbf{P}_n\right]^{\mathrm{T}} \\
&= \sum_n D_n^{-1} V_n \left[\mathbf{P}_n^{\mathrm{T}} \mathbf{J}_n^{-1} \mathbf{J}_n^{-\mathrm{T}} \mathbf{P}_n\right]^{\mathrm{T}} \\
&= \sum_n D_n^{-1} V_n \left[\mathbf{J}_n^{-\mathrm{T}} \mathbf{P}_n\right]^{\mathrm{T}} \left[\mathbf{P}_n^{\mathrm{T}} \mathbf{J}_n^{-1}\right]^{\mathrm{T}} \\
&= \sum_n D_n^{-1} V_n \mathbf{P}_n{}^{\mathrm{T}} \mathbf{J}_n^{-1} \mathbf{J}_n^{-\mathrm{T}} \mathbf{P}_n \\
&= \mathbf{S}
\end{aligned}
$$

The matrix $\mathbf{S}$ is positive definite, since

$$
\begin{aligned}
\mathbf{x}^{\mathrm{T}} \mathbf{S} \mathbf{x} &= \sum_n D_n^{-1} V_n \mathbf{x}^{\mathrm{T}} \mathbf{P}_n^{\mathrm{T}} \mathbf{J}_n^{-1} \mathbf{J}_n^{-\mathrm{T}} \mathbf{P}_n \mathbf{x} \\
&= \sum_n D_n^{-1} V_n \left[\mathbf{J}_n^{-\mathrm{T}} \mathbf{P}_n \mathbf{x}\right]^{\mathrm{T}} \left[\mathbf{J}_n^{-\mathrm{T}} \mathbf{P}_n \mathbf{x}\right] \\
&= \sum_n D_n^{-1} V_n \left\| \mathbf{J}_n^{-\mathrm{T}} \mathbf{P}_n \mathbf{x} \right\|_2 \\
&> 0 \qquad \text{if } D_n^{-1} V_n > 0 \text{ and } \mathbf{J}_n^{-\mathrm{T}} \mathbf{P}_n \mathbf{x} \neq 0
\end{aligned}
$$

If $\mathbf{S}$ is SPD, then $\mathbf{S}^{-1}$ is also symmetric positive definite.

Comparison to Morel-Hall Asymmetric Method

For an orthogonal grid, the flux out of a face can be defined simply as

$$\mathbf{F}_f^{\mathrm{T}}\mathbf{A}_f = -D_f \frac{\left(\phi_f - \phi_c\right)}{\left|\mathbf{r}_f - \mathbf{r}_c\right|} A_f \ .$$

But for a skewed grid, this is incorrect.

The Support Operator Method corrects the left hand side of the equation, defining each ϕ difference in terms of all the face fluxes:

$$\left[\sum_n D_n^{-1} V_n \mathbf{P}_n^{\mathrm{T}} \mathbf{J}_n^{-1} \mathbf{J}_n^{-\mathrm{T}} \mathbf{P}_n\right] \widetilde{\mathbf{F}} = \widetilde{\boldsymbol{\Phi}} \ .$$

The Morel-Hall Asymmetric Method corrects the right hand side of the equation, defining each face flux in terms of all of the ϕ differences:

$$\mathbf{F}_f^{\mathrm{T}}\mathbf{A}_f = -D_f \left[\mathbf{J}^{-\mathrm{T}} \mathbf{P}_f \widetilde{\boldsymbol{\Phi}}\right]^{\mathrm{T}} \mathbf{A}_f \ .$$

Support Operator Method Properties

- It is conservative.
- Material discontinuities are treated rigorously.
- It generates a symmetric positive definite matrix.
- It is second-order accurate.
- It has both cell-centered and face-centered unknowns.
- It has a local stencil.
- It reduces to the standard differencing scheme if the mesh is orthogonal.
- It is not exact for linear functions.

The Morel-Hall asymmetric method does not share the properties specified in Blue above.

Second-Order Demonstration

Support Operator Method:

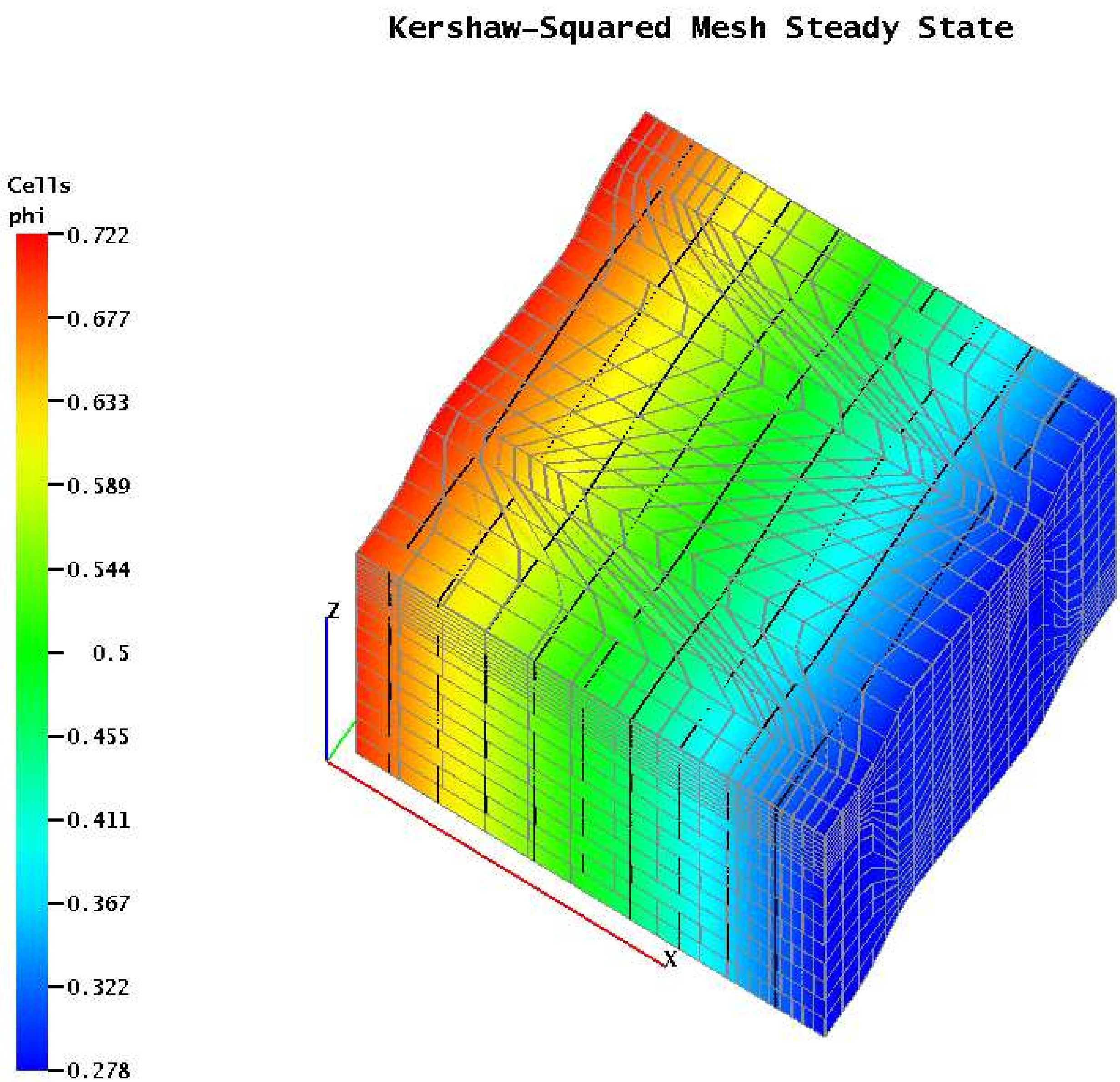
Problem Size (cells)	$\frac{\|\Phi_{\text{exact}}-\Phi\|_2}{\|\Phi_{\text{exact}}\|_2}$	Error Ratio
$2 \times 2 \times 2$	7.4950×10^{-2}	
$4 \times 4 \times 4$	2.4163×10^{-2}	3.10
$8 \times 8 \times 8$	5.5245×10^{-3}	4.37
$16 \times 16 \times 16$	1.5467×10^{-3}	3.57
$32 \times 32 \times 32$	3.6797×10^{-4}	4.20
$64 \times 64 \times 64$	9.6113×10^{-5}	3.82

Morel-Hall Asymmetric Method:

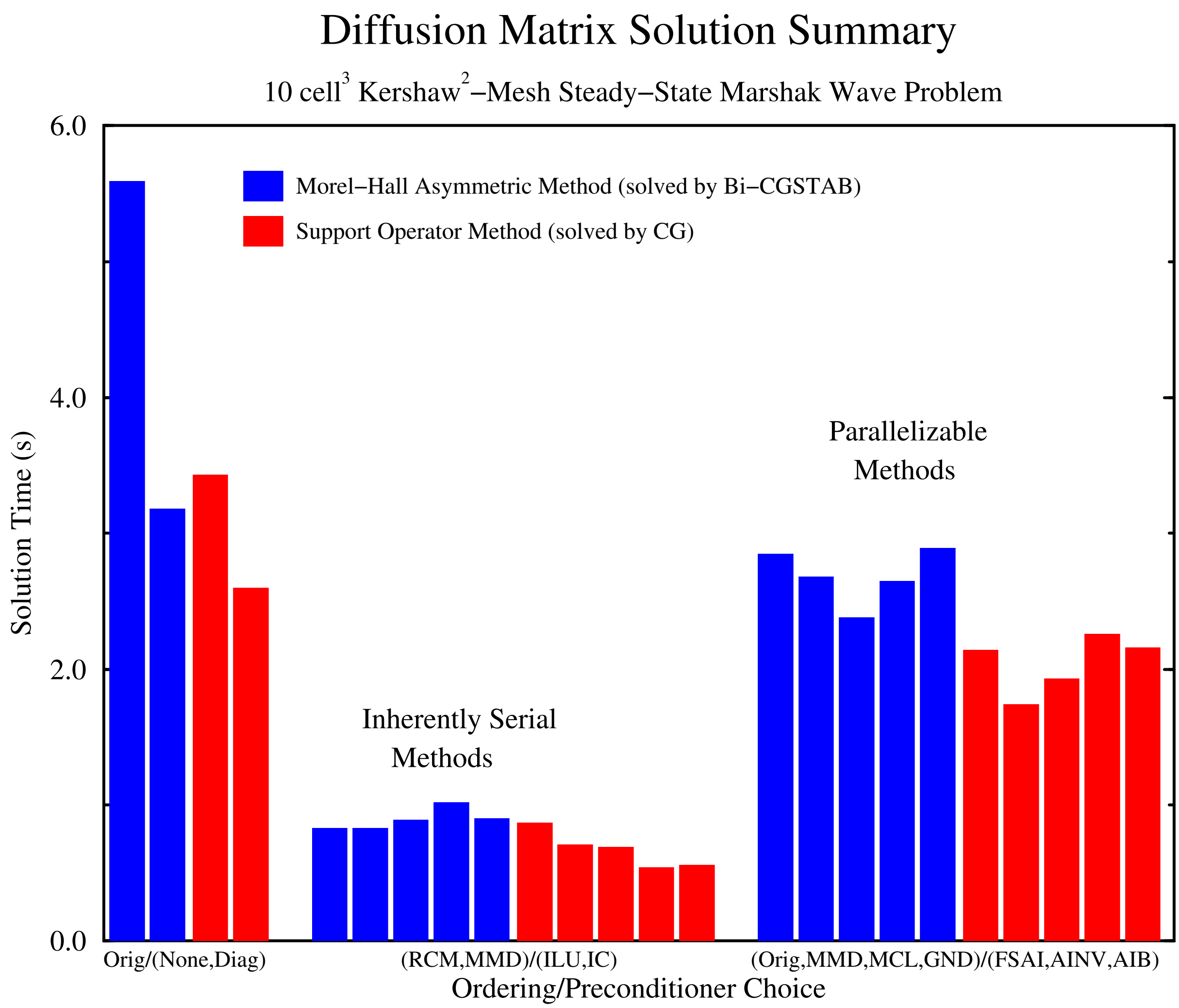
Problem Size (cells)	$\frac{\|\Phi_{\text{exact}}-\Phi\|_2}{\|\Phi_{\text{exact}}\|_2}$	Error Ratio
$2 \times 2 \times 2$	7.4350×10^{-2}	
$4 \times 4 \times 4$	2.4044×10^{-2}	3.09
$8 \times 8 \times 8$	5.4575×10^{-3}	4.41
$16 \times 16 \times 16$	1.5256×10^{-3}	3.58
$32 \times 32 \times 32$	3.6960×10^{-4}	4.12
$64 \times 64 \times 64$	9.5032×10^{-5}	3.88

Two-material problem, ratio = 10, GMRES/CG, Low-Order Preconditioner, $\epsilon = 10^{-10}$, $\epsilon_{pre} = 10^{-9}$

Kershaw-Squared Mesh



Matrix Solution Time Comparison



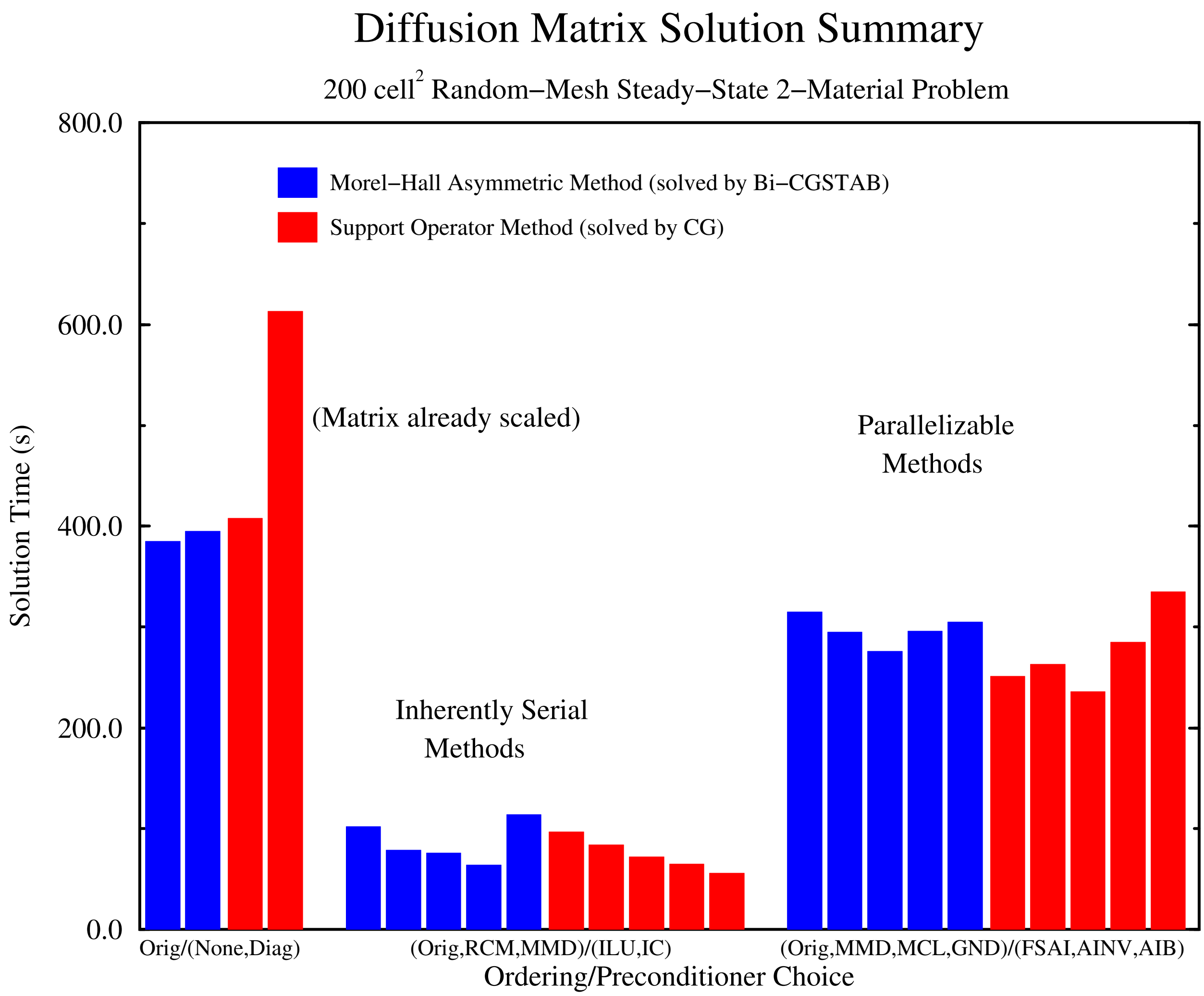
- This is a summary of extensive calculations that were done by LANL CIC-19: Michele Benzi, Mike Delong, et al. Only the five best times for each category are shown.

- All above runs were done on a Sun Ultra 2, solved to the same tolerance.

- Matrix set-up time is NOT included.

- An AMG run on this problem on one node of an SGI Origin 2000 took 4 seconds on the MH discretization and failed on the SO discretization.

Matrix Solution Time Comparison



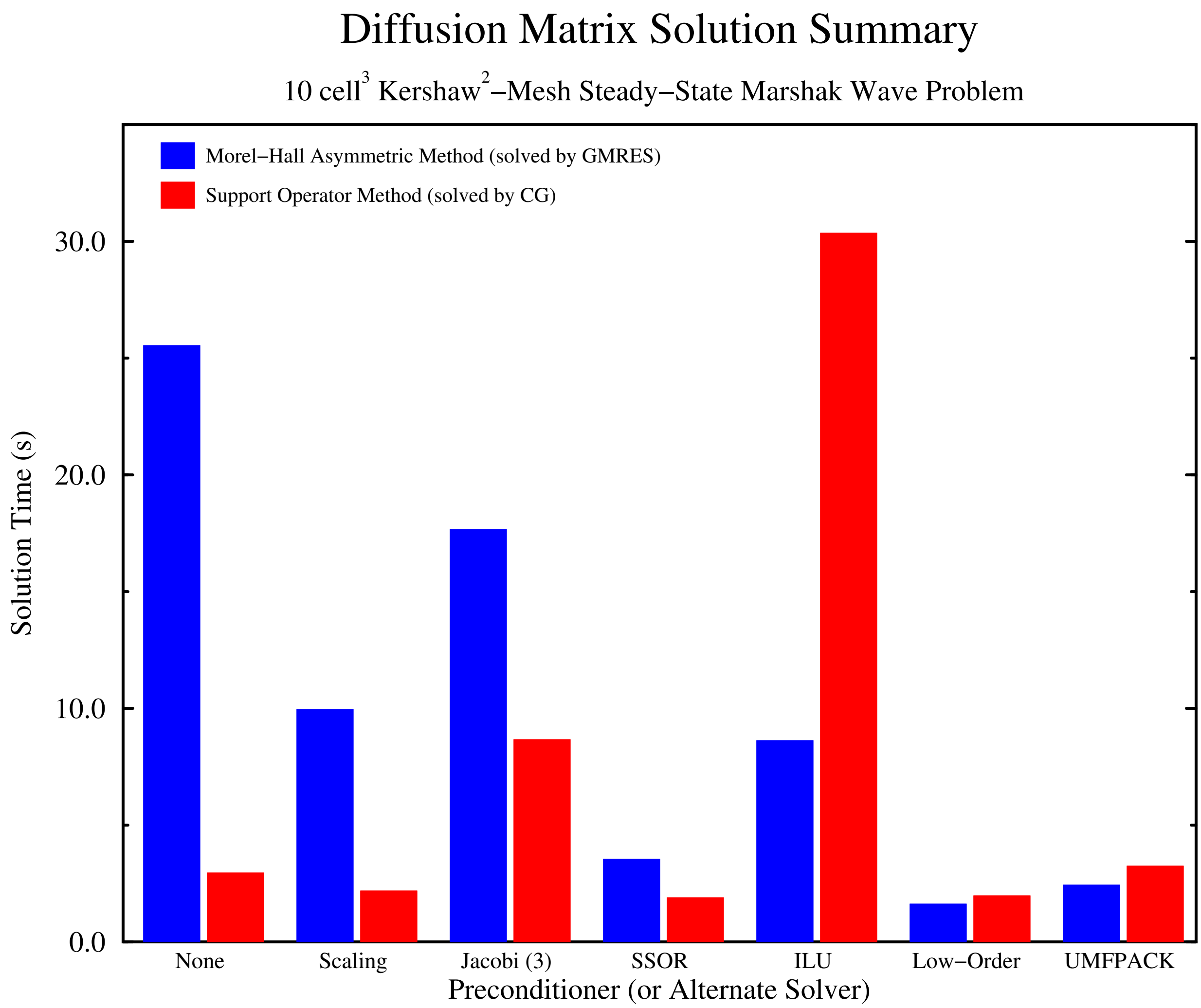
- This is a summary of extensive calculations that were done by LANL CIC-19: Michele Benzi, Mike Delong, et al. Only the five best times for each category are shown.

- All above runs were done on a Sun Ultra 2, solved to the same tolerance.

- Matrix set-up time is NOT included.

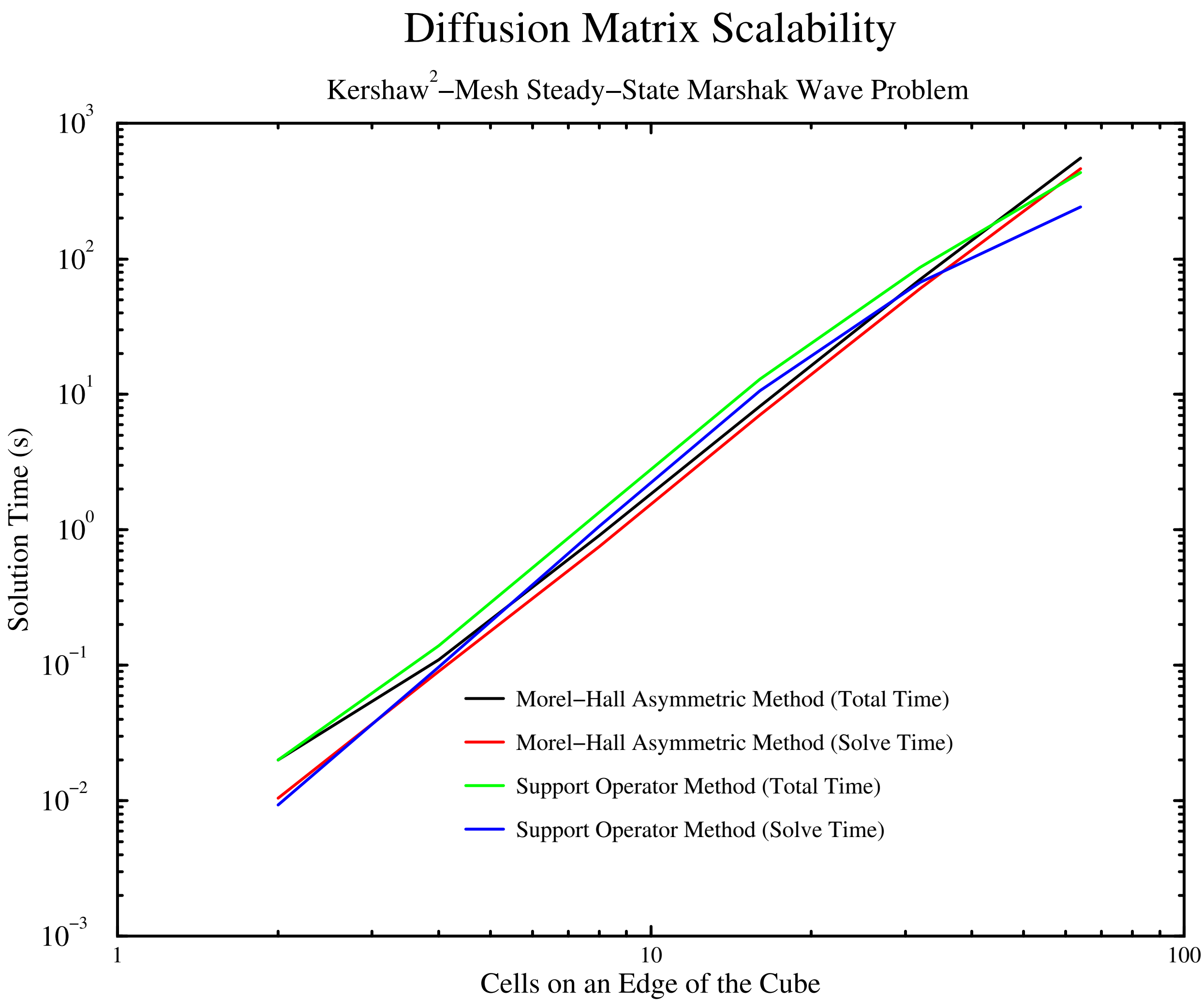
- An AMG run on this problem on one node of an SGI Origin 2000 took 124 seconds on the MH discretization and 995 seconds on the SO discretization.

Matrix Solution Time Comparison



- These results were generated from the Augustus code itself, using JTpack and UMFPACK.
- All above runs were done on a Sun Ultra 1/170. All of the Krylov solves had a tolerance of 10^{-7}, but the UMFPACK solve was accurate to machine precision, about 10^{-12}.
- Matrix set-up time IS included.

Matrix Solution Scalability



- These results were generated from the Augustus code itself, using JTpack, on a Sun Ultra 1/170, with a tolerance of 10^{-7}, using the low-order preconditioner and CG or GMRES.

- Set-Up Time, Solve Time and Total Time scale according to (edge cells)^3, which is linear in total number of cells, for both methods.

- Matrix set-up time is ∼16% for MH and ∼21% for SO.

- Ratio of MH to SO is: Total Time - 70%, Solve Time - 75%.

- The preceding statements are for mid-range – values are less accurate at the extremes.

Conclusions

- The Support Operator Methodology has been extended to 3-D Unstructured Hexahedral Meshes.

- For standard preconditioners, solution times for SO are slightly better than MH.

- For the specialized low-order preconditioners, solution times for SO are slightly worse than MH.

- Vanilla AMG works much better on MH than SO.

- Both methods provide second-order accurate solutions.

Future Work:

Parallel versions of Augustus and Spartan (a multi-group SP_N package) are being developed.