

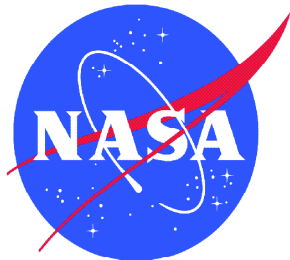
Adding Streaming Sparse Linear Matrix Algebra to StreamFEM

Tim Barth

NASA Ames Research Center

Moffett Field, California 94035-1000 USA

(Timothy.J.Barth@nasa.gov)



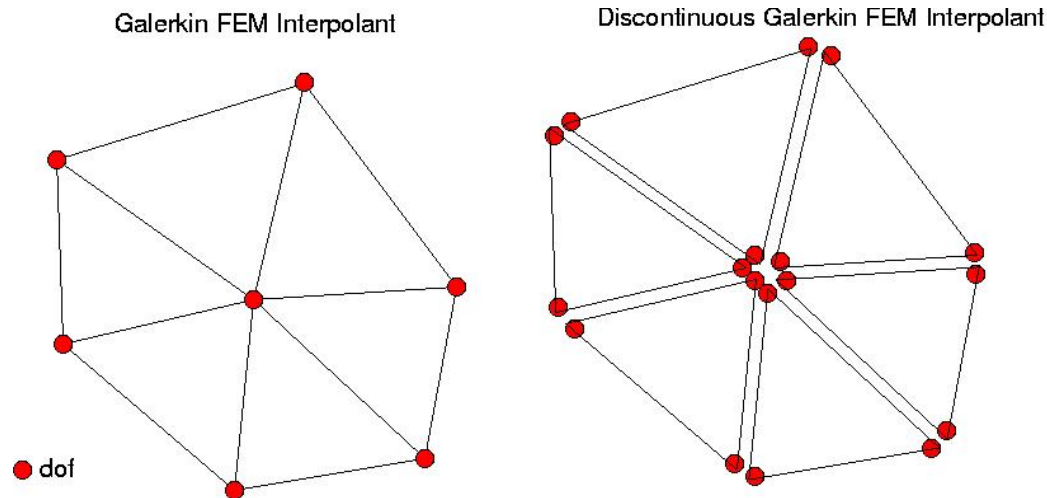
StreamFEM: $u_t + \operatorname{div}(\vec{f}) = 0$

Brook and ArrayC implementation of the **Discontinuous Galerkin Finite Element Method**:

Find $u \in V_h$ such that $\forall w \in V_h$

$$\sum_{\text{elements}} \left(\int_K w u_t d\vec{x} - \int_K \vec{f}(u) \cdot \nabla w d\vec{x} + \int_{\partial K} w_- h(n; u_-, u_+) \right) = 0$$

where $h(n; u_-, u_+)$ is a numerical flux function.

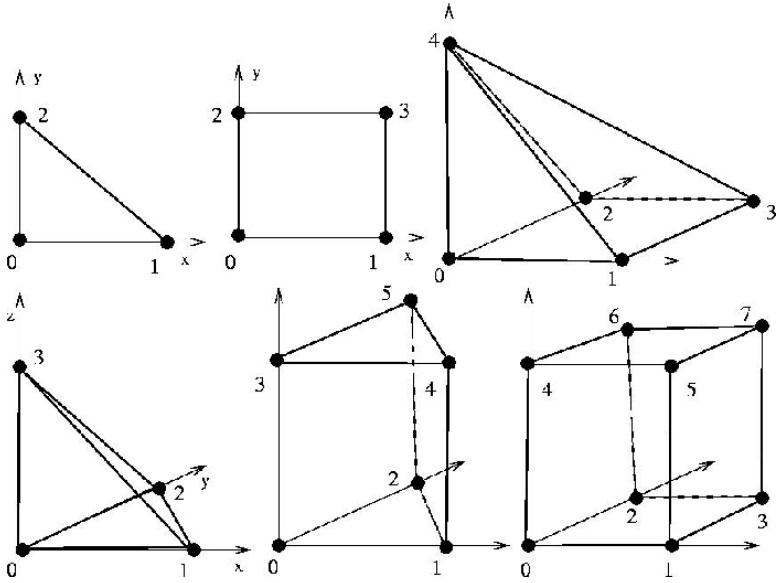


StreamFEM-3D

- User can change the arithmetic intensity
 - Choice of PDE equation set
 - * Euler equations (5 PDEs)
 - * Magnetohydrodynamics (MHD) equations (8 PDEs)
 - Choice of piecewise polynomial function space
 - * constant elements (1 dof)
 - * linear elements (4 dofs)
 - * quadratic elements (10 dofs)
 - * cubic elements (20 dofs)

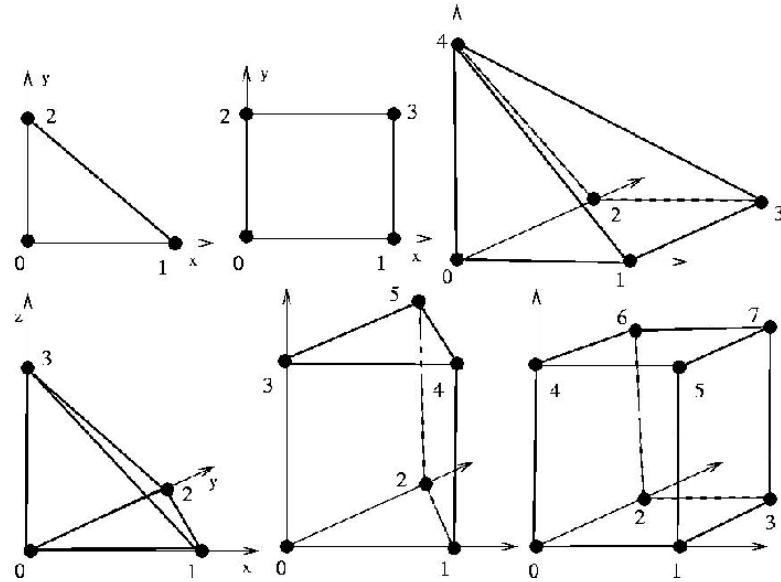
New StreamFEM Functionality

- Include most element types



New StreamFEM Functionality

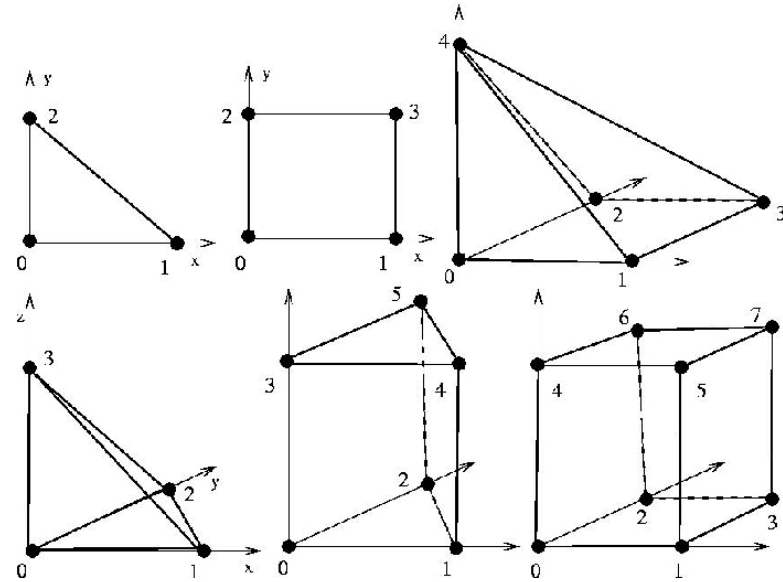
- Include most element types



- *Optional* mesh partitioning. Stream program on each mesh partition.

New StreamFEM Functionality

- Include most element types

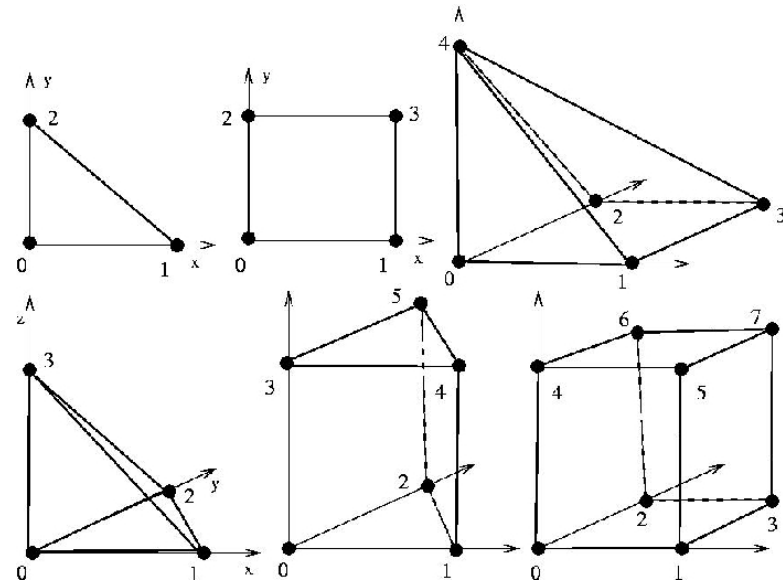


- *Optional* mesh partitioning. Stream program on each mesh partition.

Single mesh \rightarrow Multiple meshes (MPI) \rightarrow Multiple meshes (UPC)

New StreamFEM Functionality

- Include most element types



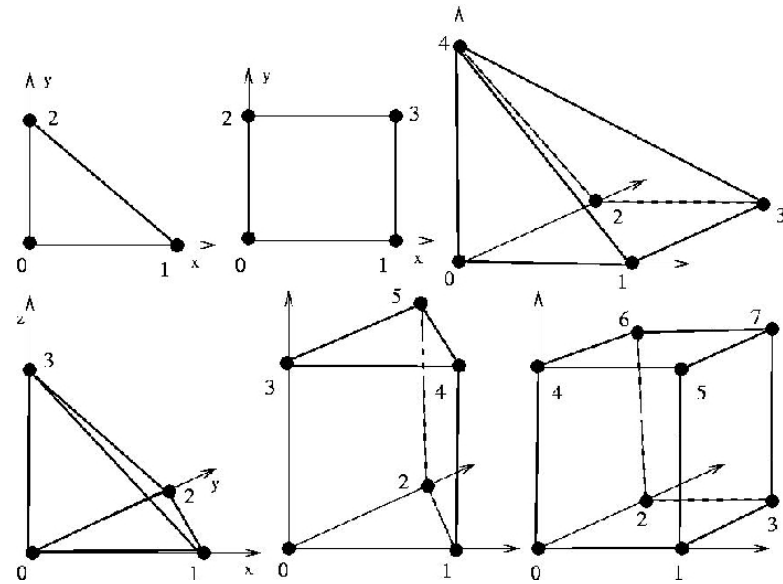
- *Optional* mesh partitioning. Stream program on each mesh partition.

Single mesh \rightarrow Multiple meshes (MPI) \rightarrow Multiple meshes (UPC)

- StreamFEM with mesh partitioning using PARMetis

New StreamFEM Functionality

- Include most element types



- *Optional* mesh partitioning. Stream program on each mesh partition.

Single mesh \rightarrow Multiple meshes (MPI) \rightarrow Multiple meshes (UPC)

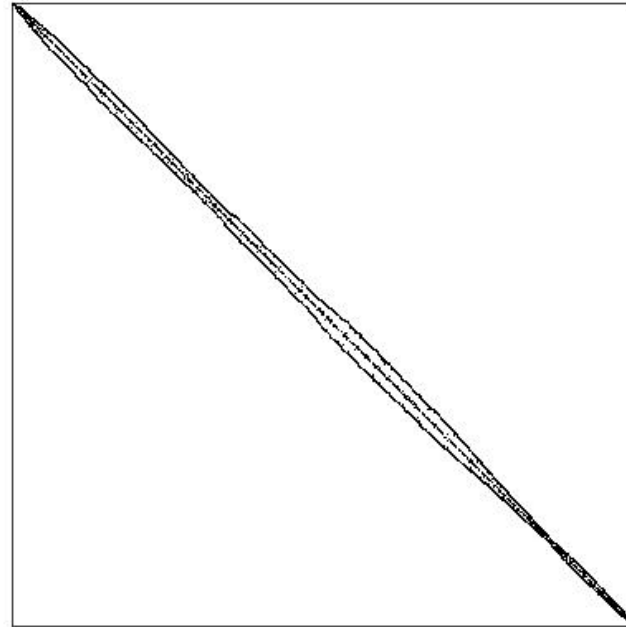
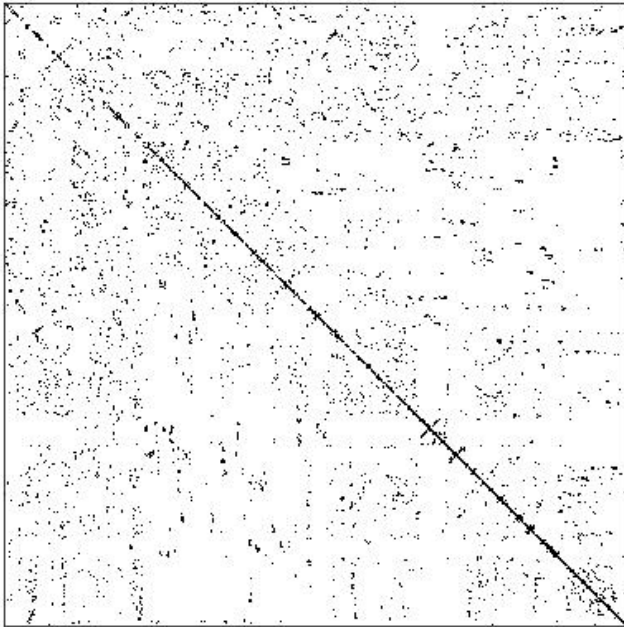
- **StreamFEM with mesh partitioning** using PARMetis
- Implements overlapping and nonoverlapping domain decomposition

Mesh Partitioning

Example $M + 1$ processors:

1. Read mesh containing N elements on control processor 0
2. Sequentially transfer N/M elements to each processor i for $i = 1, M$
3. Construct global adjacency matrix on all processors (PARMetis function)
4. Compute partitioning vector using PARMetis
5. Transfer elements to appropriate processors
6. Reorder elements in each submesh to improve memory access performance

Element Reordering



Block matrix nonzero pattern before (left) and after (right) element reordering

Sparse Linear Matrix Algebra in StreamFEM

- Replacement of StreamFEM explicit time stepping with implicit matrix solution
- None of the Merrimac applications so far address sparse matrix inversion (solution)

$$Ax - b = 0$$

- A large and sparse
- Standard direct matrix solution techniques (Gauss elimination, L-U factorization) are not suitable to stream computations

Sparse Linear Matrix Algebra

StreamFEM implements the Generalized Minimum Residual (GMRES) iterative method. GMRES has similarities with the conjugate gradient method but tailored to nonsymmetric matrices:

1. Matrix-Vector products required
2. Dot product scalar reductions required
3. Preconditioning usually needed for efficiency of GMRES

$$\text{Solve } P(Ax - b) = 0$$

- P should approximate A^{-1}
- P should be nonsingular and easily inverted (solved)

Streaming Matrix-Vector Products in StreamFEM

The sparse matrix A in StreamFEM has special structure

1. Matrix entries are dense $m \times m$ blocks where
 $m = \#\text{PDES} \times \#\text{polynomial dofs in each element}$
2. Number of off-diagonal blocks in each matrix row ≤ 3
(triangles) and ≤ 4 (tetrahedra)

Matrix-Vector Product in ArrayC

```
void spMatVec(__in double Amat[[nelements]][[nrow]][[mblock]][[mblock]],
             __in double xgather[[nelements]][[nrow]][[mblock]],
             __out double y[[nelements]][[mblock]]) {

    /* Fixed nrow number of (mblock x mblock) blocks per row */

#pragma res parallel
    doloop (int i = 0; i < nelements; i++) {
        doloop (int nr=0;nr<mblock;nr++) {
            y[[i]][[nr]] = 0.0;
            doloop (int nz = 0; nz < nrow; nz++) {
                doloop (int nc = 0; nc < mblock; nc++) {
                    y[[i]][[nr]] += Amat[[i]][[nz]][[nr]][[nc]]
                                   * xgather[[i]][[nz]][[nc]];
                }
            }
        }
    }
}
```

GMRES Preconditioning

$$P(Ax - b) = 0$$

Some standard preconditioners for nonsymmetric A

- $P^{-1} = \text{BlockDiag}(A)$. Block diagonal preconditioning.
 - Straightforward to stream but not very effective for general StreamFEM systems
- $P^{-1} = \text{ILU}(k)(A)$, Incomplete LU factorization with fill level k .
 - Sequential algorithm not amenable to stream computation

Candidate Streaming Preconditioners

$$P(Ax - b) = 0$$

Let $A = M + N$. Compute the preconditioner action $P^{-1}u$ by solving
 $Pv = u$

$$Mv^{n+1} + Nv^n = u, \quad n = 1, 2, 3, \dots$$

or with relaxation parameter

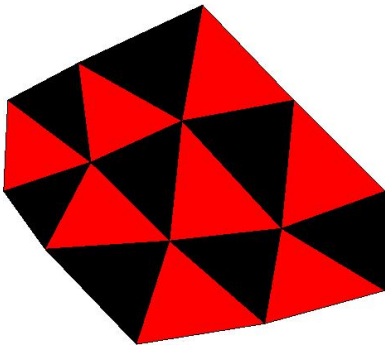
$$v^{n+1} = v^n + \omega M^{-1}(u - Av^n)$$

- M lower triangular. Gauss-Seidel iteration
 - Requires a stream implementation of a SweepLine algorithm (??)
 - Most straightforward on structured meshes
- M diagonal. Point Jacobi, $\text{SOR}(\omega)$, colored $\text{SOR}(\omega)$
 - Candidate of choice on vector architectures has historically been colored $\text{SOR}(\omega)$

Work in Progress: Colored SOR Preconditioning

Exploit the StreamFEM attribute that the number of nonzero blocks in each row is bounded by a small number (3 triangles, 4 tetrahedra)

- Color graph of cells in mesh. Doesn't need to be optimal wrt the number of colors.
- Precondition using multicolored variant of SOR.
 1. Randomize ordering of colors. Example (red, black)



$$v_{\text{red}}^{n+1} = v_{\text{red}}^n + \omega M_{\text{red}}^{-1}(u - Av_{\text{black}}^n)$$

$$v_{\text{black}}^{n+1} = v_{\text{black}}^{n+1} + \omega M_{\text{black}}^{-1}(u - Av_{\text{red}}^{n+1})$$

$$v_{\text{black}}^{n+2} = v_{\text{black}}^{n+1} + \omega M_{\text{black}}^{-1}(u - Av_{\text{red}}^{n+1})$$

$$v_{\text{red}}^{n+2} = v_{\text{red}}^{n+1} + \omega M_{\text{red}}^{-1}(u - Av_{\text{black}}^{n+2})$$

R-Stream 2.0 Issues

- Merrimac machine model?
- Dreaded “Loop tile size exceeds local memory”
- Gather/Scatter