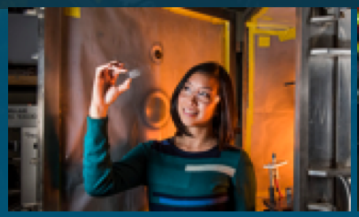
Guest Lecture Stanford ME469: An Example of an Unstructured Discretization



PRESENTED BY

Stefan P. Domino

Computational Thermal and Fluid Mechanics

Sandia National Laboratories SAND2018-4536 PE

Sandia National Laboratories is a multimission laboratory managed and operated by National Technology & Engineering Solutions of Sandia, LLC, a wholly owned subsidiary of Honeywell International Inc., for the U.S. Department of Energy's National Nuclear Security Administration under contract DE-NA0003525.

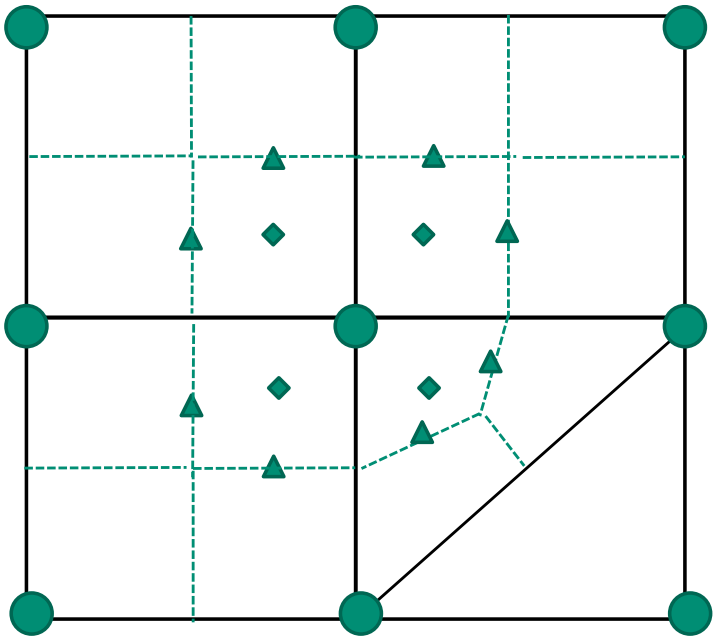
An Example of an Unstructured Discretization: Outline

- Review of CVFEM
- Sample Diffusion Kernel, CVFEM
- Review of EBVC
- Sample Diffusion Kernel, EBVC
- Conclusions

The Hybrid Control-Volume Finite Element Method (CVFEM)

- A combination between the edge-based vertex-centered and FEM is the method known as Control Volume Finite Element ()
- A dual mesh is constructed to obtain flux and volume quadrature locations
- As with FEM, a basis is defined:

$$T(x_k) \approx \sum_{i=1}^{npe} N_i(x_k) T_i \qquad \frac{\partial T(x_k)}{\partial x_j} \approx \sum_{i=1}^{npe} \frac{\partial N_i(x_k)}{\partial x_j} T_i$$



Dual-volume definition

- Integration-by-parts over test function w:

$$\int w\rho C_p \frac{\partial T}{\partial t} dV + \int \frac{\partial w}{\partial x_j} \lambda \frac{\partial T}{\partial x_j} dV - \int w\, \lambda \frac{\partial T}{\partial x_j} n_j dS = 0$$

- However, define a test function, w, as a piece-wise constant function (Heavyside) to be 1 inside the dual volume and 0 outside. Gradient is a Dirac-delta function:

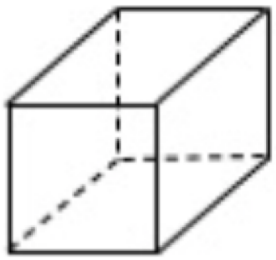
$$\frac{\partial w}{\partial x_j} = -n_j \delta(x_j - xIP_j)$$

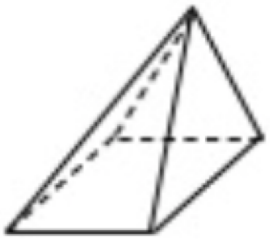
- Leading to:

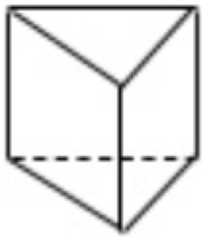
$$\int \rho C_p \frac{\partial T}{\partial t} dV - \int \lambda \frac{\partial T}{\partial x_j} n_j\, dS = 0$$

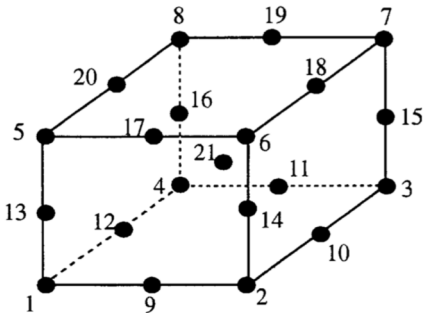
Examples of Various Topologies

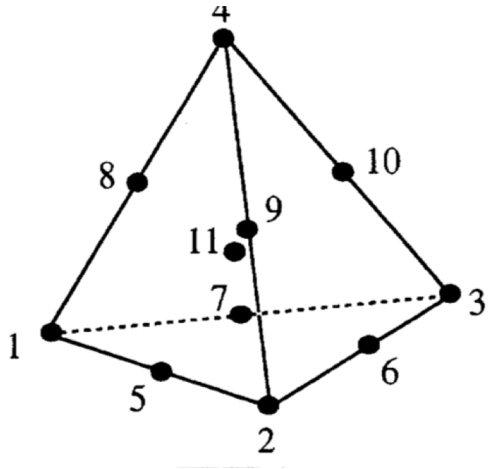
Hex8 | Tet4 | Pyramid5 | Wedge6 | Arbitrary

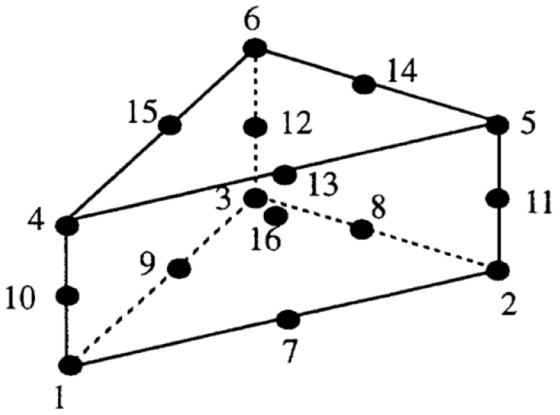












Higher-order promoted elements (Hex27, Tet10, Wedge16, Hex64, etc.)

Code Design: Managing a Hybrid Mesh

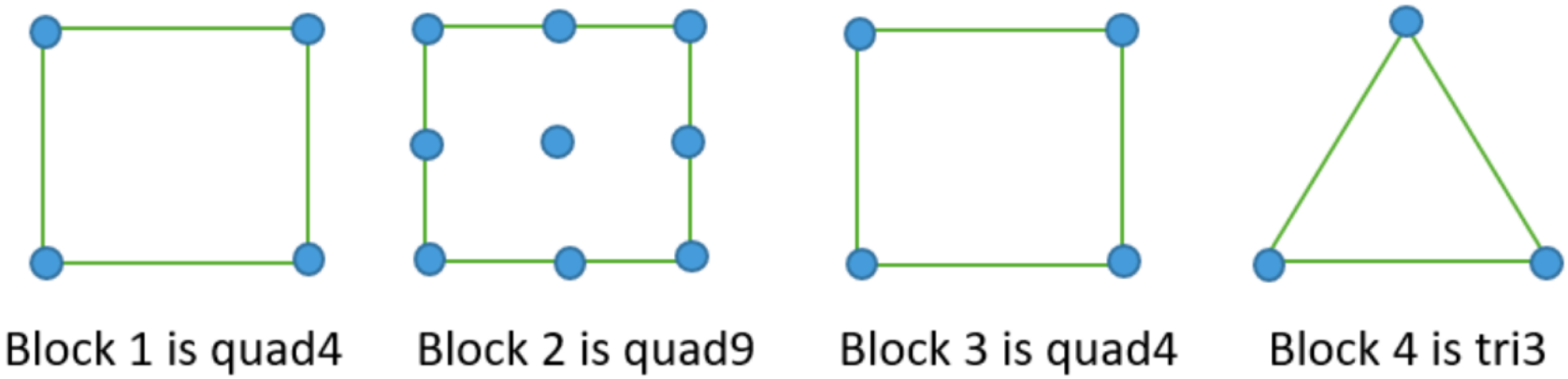


Figure 4: Heterogeneous topologies example.

Attributes:

- Three unique topologies: quad_4, quad9, tri3
- Iterate a set of “parts” that map to the homogeneous element blocks
- Abstract out the integration rule, i.e., AlgTraits::nodesPerElement_, etc.
- Create one algorithm per element topology type (avoids resizing)

Sample Nalu ElemKernel: Construction

```
template<typename AlgTraits>
MomentumNSOElemKernel<AlgTraits>::MomentumNSOElemKernel(
  ElemDataRequests& dataPreReqs)
{
  // define master element rule for this kernel
  MasterElement *meSCS
    = sierra::nalu::MasterElementRepo::get_surface_master_element(AlgTraits::topo_);

  // add ME rule
  dataPreReqs.add_cvfem_surface_me(meSCS);

  // add fields to gather
  dataPreReqs.add_coordinates_field(*coordinates_, AlgTraits::nDim_, CURRENT_COORDINATES);
  dataPreReqs.add_gathered_nodal_field(*velocityNp1_, AlgTraits::nDim_);

  // add ME calls
  dataPreReqs.add_master_element_call(SCS_GIJ, CURRENT_COORDINATES);
}
```

Listing 4: Attributes of a kernel; part A the constructor.

Constructor defines the fields to gather and the element operations required, e.g., dndx, area, etc

Sample Nalu ElemKernel: Execution

Design: Elements are “served” to the ElemKernel

```
template<typename AlgTraits>
void
MomentumNSOElemKernel<AlgTraits>::execute(
  SharedMemView<DoubleType**>& lhs,
  SharedMemView<DoubleType *>& rhs,
  ScratchViews<DoubleType>& scratchViews)
{
  SharedMemView<DoubleType**>& v_uNp1
    = scratchViews.get_scratch_view_2D(*velocityNp1_);
  SharedMemView<DoubleType***>& v_gijUpper
    = scratchViews.get_me_views(CURRENT_COORDINATES).gijUpper;

  for ( int ip = 0; ip < AlgTraits::numScsIp_; ++ip ) {

    // determine scs values of interest
    for ( int ic = 0; ic < AlgTraits::nodesPerElement_; ++ic ) {

    // assemble each component
    for ( int k = 0; k < AlgTraits::nDim_; ++k ) {

      // determine scs values of interest
      for ( int ic = 0; ic < AlgTraits::nodesPerElement_; ++ic ) {

        // save off velocityUnp1 for component k
        const DoubleType& ukNp1 = v_uNp1(ic,k);

        // denominator for nu as well as terms for "upwind" nu
        for ( int i = 0; i < AlgTraits::nDim_; ++i ) {
          for ( int j = 0; j < AlgTraits::nDim_; ++j ) {
            gUpperMagGradQ += constant*v_gijUpper(ip,i,j);
          }
        }
      }
    }
  }
}
```

Listing 5: Attributes of a kernel; part B the body.

Thread-local scratch arrays using
A Kokkos SharedMemView

Templated

SIMD

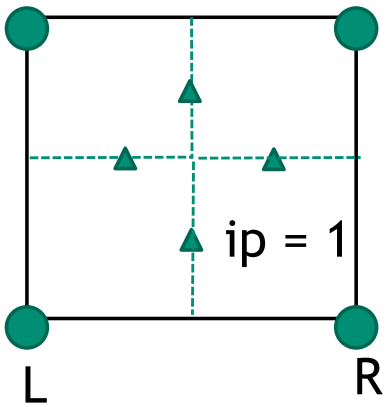
MD-array rather than
error-prone
pointer arithmetic

A CVFEM Diffusion Kernel

```
// start the assembly
for ( int ip = 0; ip < AlgTraits::numScsIp_; ++ip ) {

  // left and right nodes for this ip
  const int il = lrscv_[2*ip];
  const int ir = lrscv_[2*ip+1];

  // compute ip property
  DoubleType diffFluxCoeffIp = 0.0;
  for ( int ic = 0; ic < AlgTraits::nodesPerElement_; ++ic ) {
    const DoubleType r = v_shape_function_(ip,ic);
    diffFluxCoeffIp += r*v_diffFluxCoeff(ic);
  }

  // assemble to rhs and lhs
  DoubleType qDiff = 0.0;
  for ( int ic = 0; ic < AlgTraits::nodesPerElement_; ++ic ) {
    DoubleType lhsfacDiff = 0.0;
    for ( int j = 0; j < AlgTraits::nDim_; ++j ) {
      lhsfacDiff += -diffFluxCoeffIp*v_dndx(ip,ic,j)*v_scs_areav(ip,j);
    }
    qDiff += lhsfacDiff*v_scalarQ(ic);

    // lhs; il then ir
    lhs(il,ic) += lhsfacDiff;
    lhs(ir,ic) -= lhsfacDiff;
  }

  // rhs; il then ir
  rhs[il] -= qDiff;
  rhs[ir] += qDiff;
}
}
```

$$\int q_j n_j dS = -\int \lambda \frac{\partial \phi}{\partial x_j} n_j dS$$

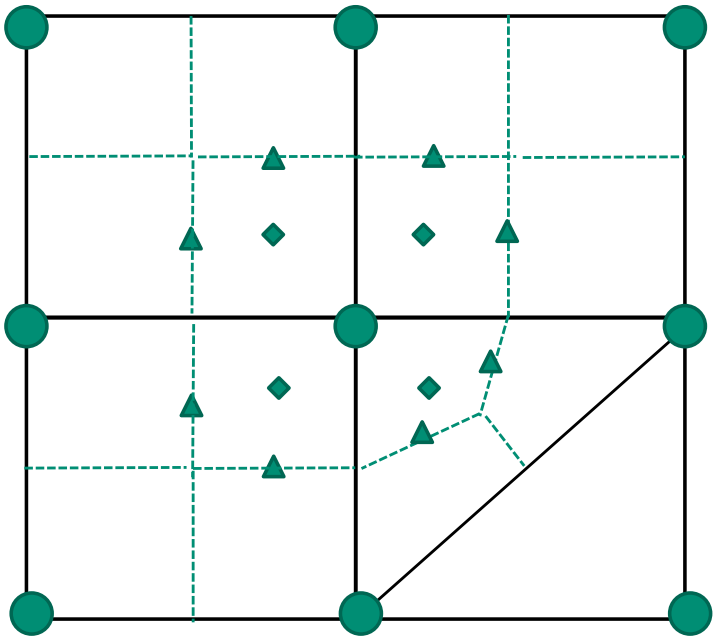
$$\lambda_{ip} = \sum_n N_{n,ip} \lambda_n$$



$$-\sum_{ip} \lambda_{ip} \sum_n \frac{\partial N_{n,ip}}{\partial x_j} \phi_n n_j dS$$

Equal Order Interpolation Edge-Based Vertex-Centered (EBVC) Finite Volume

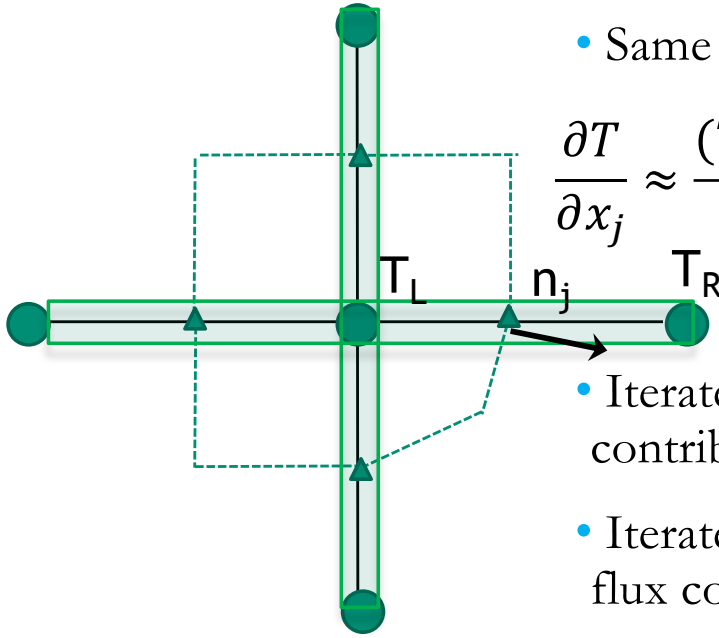
- All primitives are collocated at the vertices of the elements with equal-order interpolation
- A dual mesh is constructed to obtain flux and volume quadrature locations
- Classic two-state, "L" and "R" approach provides spatially second-order accuracy



Dual-volume definition

▲ Surface quadrature point (area summed to edge)

◆ Volume quadrature point (sub-vol summed to node)



Edge-based stencil

- Same grad-op as CC:

$$\frac{\partial T}{\partial x_j} \approx \frac{(T_R - T_L)n_j}{\|\Delta x_j\|} + O(\Delta x)$$

- Iterate nodes for volume contributions
- Iterate edges for surface flux contribution

An EBVC Algorithm

- Regardless of topology (restricted to low-order), edges are unique

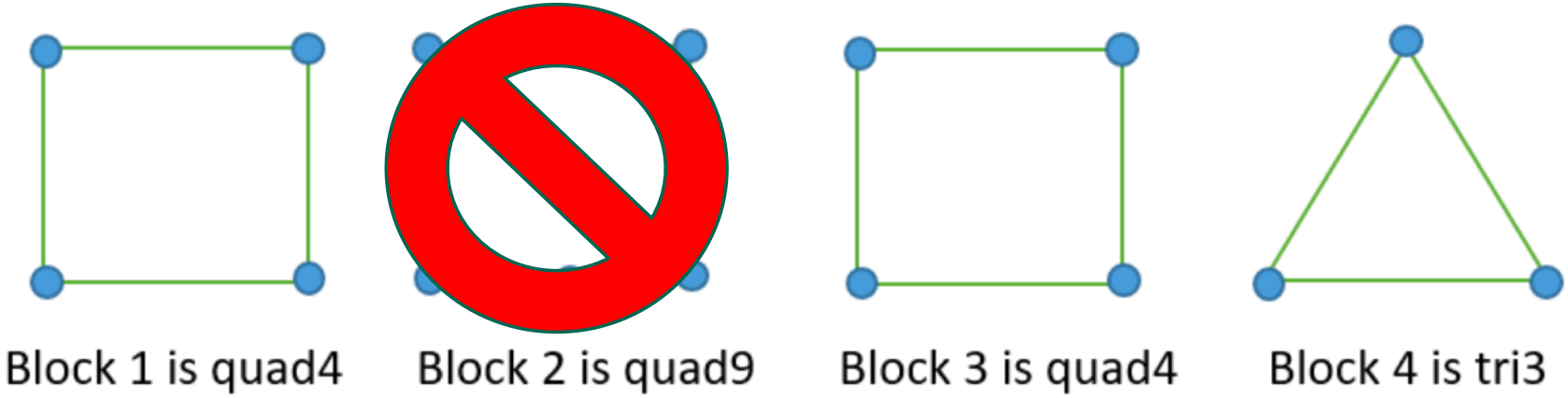


Figure 4: Heterogeneous topologies example.

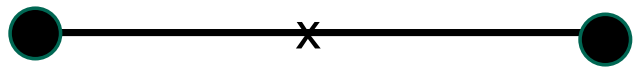
- Therefore, the topology over which we loop is the edge2 with "L" and "R" nodes, respectively

Edge-Based Sample Code

```
for ( stk::mesh::Bucket::size_type k = 0 ; k < length ; ++k ) {

      // get edge
      stk::mesh::Entity edge = b[k];

      // left and right nodes
      stk::mesh::Entity nodeL = edge.edge_node_rels[0];
      stk::mesh::Entity nodeR = edge.edge_node_rels[1];

      const double viscIp = 0.5*(diffFluxCoeffL + diffFluxCoeffR);
       const double qNp1L = *stk::mesh::field_data( scalarQNp1, nodeL );
      const double qNp1R = *stk::mesh::field_data( scalarQNp1, nodeR );

      double lhsfac = -viscIp*asq*inv_axdx;
      double diffFlux = lhsfac*(qNp1R - qNp1L) + nonOrth;

      // first left
      p_lhs[0] = -lhsfac;
      p_lhs[1] = +lhsfac;
      p_rhs[0] = -diffFlux;

      // now right
      p_lhs[2] = +lhsfac;
      p_lhs[3] = -lhsfac;
      p_rhs[1] = diffFlux;
}
```

$$\int q_j n_j dS = -\int \lambda \frac{\partial \phi}{\partial x_j} n_j dS$$

$$\frac{\partial \phi}{\partial x_j} = G_j \phi + [(\phi_R - \phi_L) - G_k \phi \Delta x_k] \frac{A_j}{A_l \Delta x_l}$$



An Example of an Unstructured Discretization: Conclusions

- Decide on the topology over which one must loop to assemble the PDE discrete contributions
- Provide a code interface that supports this topology loop with minimal resizing
- Solve in residual form: $A\Delta x = -Res$