Progress towards development of discontinuous Galerkin finite-element methods for compressible flows using Charm++

Aditya K. Pandare[†], Jozsef Bakosi[*], Hong Luo[†]

[†]Department of Mechanical and Aerospace Engineering,
North Carolina State University.

[*] Computer, Computational and Statistical Sciences,
Los Alamos National Laboratory.

16th Annual Charm++ Workshop,
11-12 April 2018

Overview

- Platform
- Physical system
- Numerical approximation
- Considerations for parallel computation
- Some examples
- Future plan

Code platform: Quinoa

- Written in modern C++11 and native Charm++
- Fully asynchronous parallel programming
- Extensive unit and regression tested

Capabilities:

- Continuous Galerkin solver for compressible fluid flow
- Fully unstructured tetrahedral mesh support
- Stochastic differential equation particle solver
- Dynamic load balancing capabilities using over-decomposition

The Physical System: from continuum equations to code

- Transport phenomena represented by system of (non-linear) PDEs

$$\frac{\partial \mathbf{U}}{\partial t} + \frac{\partial \mathbf{F}_k}{\partial x_k} = \mathbf{S}$$

- Multiply with a test-function and integrate over the entire domain:

$$\int_\Omega \frac{\partial \mathbf{U}}{\partial t} W_j \mathrm{d}\Omega + \int_\Gamma \mathbf{F}_k(U) n_k W_j \mathrm{d}\Gamma - \int_\Omega \mathbf{F}_k(U) \frac{\partial W_j}{\partial x_k} \mathrm{d}\Omega = \int_\Omega \mathbf{S}(U) W_j \mathrm{d}\Omega$$

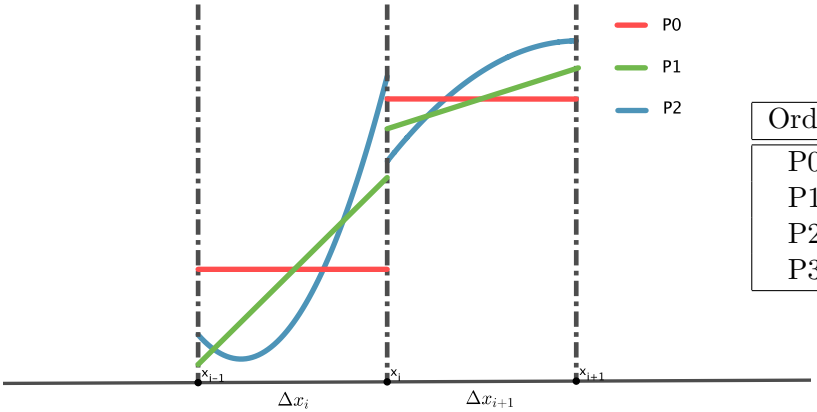
- Choose a triangulation $\Omega_e \in \Omega$ with $\Gamma_e = \partial\Omega_e$, and then approximate the solution as:

$$U(\boldsymbol{x}) \approx U_h(\boldsymbol{x}) = \sum_i u_i B_i(\boldsymbol{x}) \quad \forall\, \boldsymbol{x} \in \Omega_e$$

B_i is a *compact* basis for the continuous function U on element Ω_e.

High-order DG

Solution polynomial: $\mathbf{U}_h(x) = \sum_i^{ndof} u_i B_i(x)$



Order	B_i
P0	$[1]$
P1	$[1, \tilde{x}]$
P2	$[1, \tilde{x}, \tilde{x}^2]$
P3	$[1, \tilde{x}, \tilde{x}^2, \tilde{x}^3]$

Discrete (weak) form of integral conservation laws

- The above weak form can be written per element using this approximation as:

$$\left(\int_{\Omega_e} B_i W_j \mathrm{d}\Omega\right) \frac{\partial \mathbf{u}_i}{\partial t} + \int_{\Gamma_e} \mathbf{F}_k(U_h) n_k W_j \mathrm{d}\Gamma - \int_{\Omega_e} \mathbf{F}_k(U_h) \frac{\partial W_j}{\partial x_k} \mathrm{d}\Omega = \int_{\Omega_e} \mathbf{S}(U_h) W_j \mathrm{d}\Omega$$

$$\mathbf{M}_{ij} \frac{\partial \mathbf{u}_i}{\partial t} = \mathbf{R}_j$$

  For Galerkin-type finite-element methods, $W_i = B_i$. This is the equation to be solved.

- Special case: First order approximation for U (Finite volume) : $B_1 = 1$

$$\Omega_e \frac{\partial \mathbf{u}_h}{\partial t} + \mathbf{F}_k(u_h) n_k \Gamma_e = \mathbf{S}(u_h) \Omega_e$$

Examples of PDE systems

- Scalar advection:

$$\mathbf{U} = \phi, \quad \mathbf{F} = \boldsymbol{a}\phi$$

- Compressible Euler equations of gas dynamics:

$$\mathbf{U} = \begin{bmatrix} \rho \\ \rho u_i \\ \rho E \end{bmatrix}, \quad \mathbf{F} = \begin{bmatrix} \rho \boldsymbol{u} \\ \rho u_i \boldsymbol{u} + p \\ (\rho E + p)\boldsymbol{u} \end{bmatrix}$$

Time integration

Explicit forward Euler discretization:

$$\mathbf{M}_{ij} \frac{\boldsymbol{u}_i^{n+1} - \boldsymbol{u}_i^n}{\Delta t} = \mathbf{R}_j(\boldsymbol{u}_h^n)$$

- Ensure time-step Δt is within CFL restrictions
- Compute volume and surface integrals for $\mathbf{R}_j \rightarrow$ most time-consuming
- Update vector of solution DOFs $\boldsymbol{u}_h$

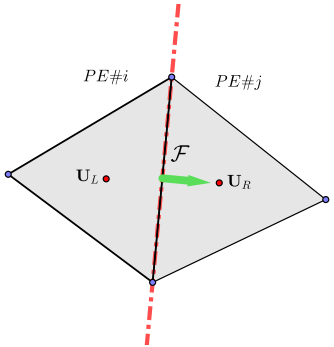
Time-stepping is done without using any global reductions, only using SDAG.

Riemann solvers

- Solution discontinuous at cell boundaries Γ_e
- How to uniquely define integrals $\int_{\Gamma_e} \mathbf{F}_k n_k \mathrm{d}\Gamma_e$?
  $\rightarrow$ Riemann solver.
- $\int_{\Gamma_e} \mathbf{F}_k n_k \, \mathrm{d}\Gamma_e \approx \int_{\Gamma_e} \mathcal{F}_k(u_L, u_R) \, n_k \, \mathrm{d}\Gamma_e$
- Design based on physics:
  - Upwind
  - Downwind
  - Central-difference

Parallel communication for DG methods

- Riemann solver requires left and right states
- Requires communication at processor boundaries



Communicated information

1) "Outer" element-id corresponding to face-id
2) Geometry information of this element
3) Solution vector of this element
4) $\rightarrow$ Communication map depends on the derived data

Derived data structures for unstructured meshes

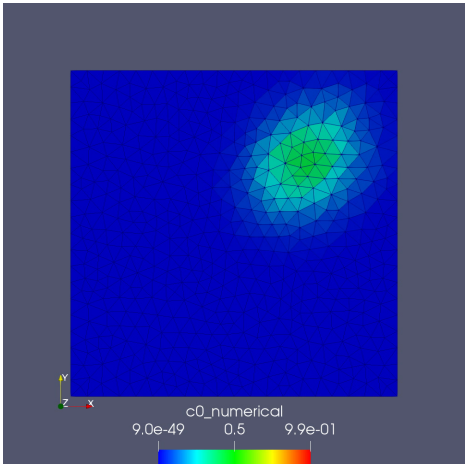
- Elements surrounding elements
- Elements surrounding faces
- Vertices of faces

Additionally, data structures required for parallel communication:

- Partition face-ids and corresponding “inner” element-id associated to chare-ids
- Mapping for local “outer” element-id to remote element-id

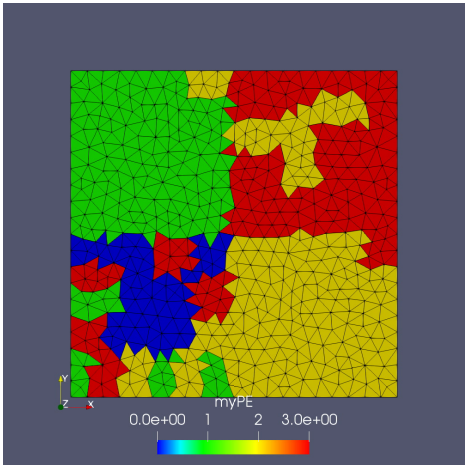
Preliminary results

Linear advection of a Gaussian hump (on 3D Tetrahedral mesh)



Preliminary results for load balancing

- Perform excess computations (fake) on some chares
- Sanity-check of the chare-migration and load-balancing

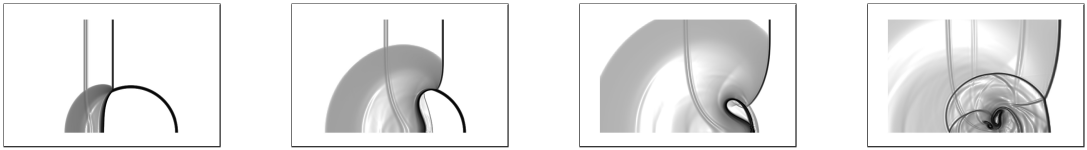


Roadmap for DG

1) Extension of DG(P0) to coupled systems of PDEs
2) High-order discontinuous Galerkin discretization
3) p-refinement

p-refinement

- Order of solution polynomial raised/lowered based on errors/solution moments
- Initial mesh-based distribution on PEs now unbalanced
- Example† of unbalanced computational load:



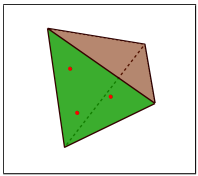
→ Charm++ load balancers

† From a separate research: Pandare A. Luo H., 2018, "A robust and efficient finite volume method for compressible viscous two-phase flows", *In press*.

High-order DG

- Volume and surface integrals: $\int_{\Gamma_e} \mathbf{F}_k(U_h) n_k B_j \mathrm{d}\Gamma - \int_{\Omega_e} \mathbf{F}_k(U_h) \frac{\partial B_j}{\partial x_k} \mathrm{d}\Omega$
- Approximate both integrals using Gauss quadrature

Example of surface integral over (green) face:



$$\rightarrow \qquad \int_{\Gamma_e} \mathbf{G}(U_h) \mathrm{d}\Gamma = \sum_{igp} \left(w_i \mathbf{G}(U_i) \right) \Gamma_e$$

As order of solution polynomial increases:

- Number of integration points for an accurate numerical approximation increases.
- Computational effort per-element increases.

Causes of load-imbalance

p-adaptation leads to the following:

- More number of unknowns to solve for, and larger problem size
- Gaussian quadrature requires more integration points at high-order zones

Other possible sources:

- Expensive limiting strategies to preserve monotonicity for highly nonlinear problems.

Summary

- Ongoing development of a high-order p-adaptive discontinuous Galerkin method for fluid dynamics
- Dynamic load balancing using over-decomposition and migration using Charm++
- Production-style coding

Thank you!