

Performance-Portable Implicit Scale-Resolving Compressible Flow Using libCEED

SIAM CSE 2023

James Wright, Jed Brown, Kenneth Jansen, Leila Ghaffari

February 27, 2023

Ann and H.J. Smead Department of Aerospace Engineering Sciences



Smead Aerospace
UNIVERSITY OF COLORADO **BOULDER**

1. libCEED Overview
2. Compressible Fluid Equations in libCEED
3. Viscous Outflow Boundary Conditions
4. Efficient Implicit Timestepping



libCEED Overview



What is libCEED?

- C library for element-based discretizations
 - Bindings available for Fortran, Rust, Python, and Julia



What is libCEED?

- C library for element-based discretizations
 - Bindings available for Fortran, Rust, Python, and Julia
- Designed for matrix-free operator evaluation



What is libCEED?

- C library for element-based discretizations
 - Bindings available for Fortran, Rust, Python, and Julia
- Designed for matrix-free operator evaluation
- Portable to different hardware via computational backends
 - Code that runs on CPU also runs on GPU without changes
 - Computational backend selectable at runtime, using runtime compilation



What is libCEED?

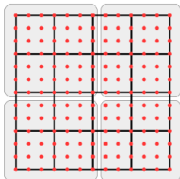
- C library for element-based discretizations
 - Bindings available for Fortran, Rust, Python, and Julia
- Designed for matrix-free operator evaluation
- Portable to different hardware via computational backends
 - Code that runs on CPU also runs on GPU without changes
 - Computational backend selectable at runtime, using runtime compilation
- Geared toward high-order finite element discretizations



Finite Element Operator Decomposition

$$A = \mathcal{P}^T \mathcal{E}^T B^T D B \mathcal{E} \mathcal{P}$$

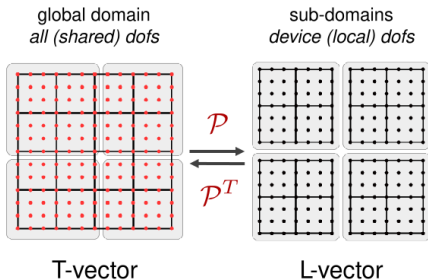
global domain
all (shared) dofs



T-vector

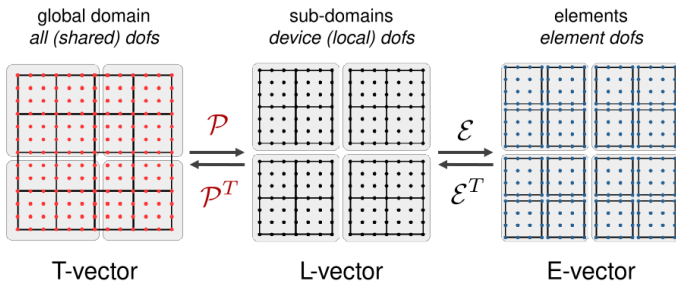
Finite Element Operator Decomposition

$$A = \mathcal{P}^T \mathcal{E}^T B^T D B \mathcal{E} \mathcal{P}$$



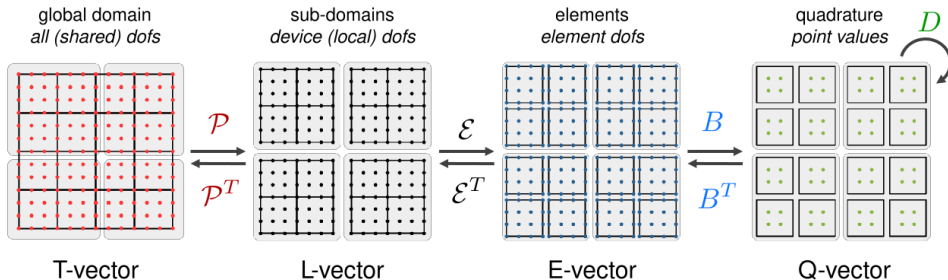
Finite Element Operator Decomposition

$$A = \mathcal{P}^T \mathcal{E}^T B^T D B \mathcal{E} \mathcal{P}$$



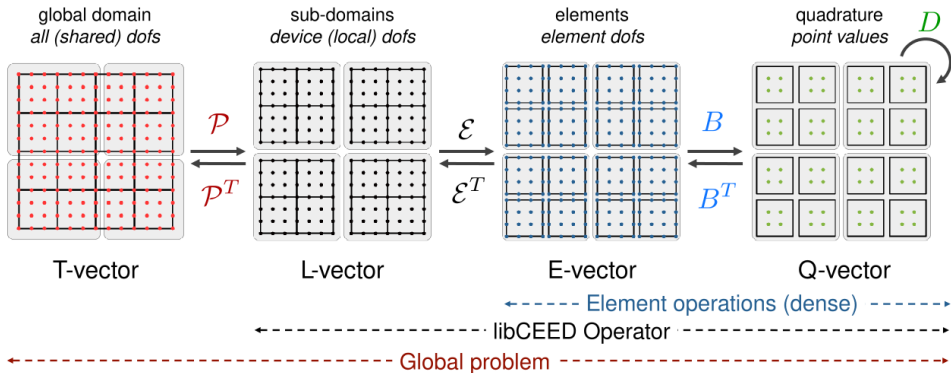
Finite Element Operator Decomposition

$$A = \mathcal{P}^T \mathcal{E}^T B^T D B \mathcal{E} \mathcal{P}$$



Finite Element Operator Decomposition

$$A = \mathcal{P}^T \mathcal{E}^T B^T D B \mathcal{E} \mathcal{P}$$



Compressible Fluid Equations in libCEED



Compressible Navier-Stokes

$$\mathbf{A}_0 \mathbf{Y}_{,t} + \mathbf{F}_{i,i}(\mathbf{Y}) - S(\mathbf{Y}) = 0$$

for

$$\underbrace{\mathbf{A}_0 \begin{bmatrix} p \\ u_i \\ T \end{bmatrix}}_{\mathbf{Y}} = \begin{bmatrix} \rho \\ \rho u_i \\ \rho e \end{bmatrix}, \quad \mathbf{F}_i(\mathbf{Y}) = \underbrace{\begin{pmatrix} \rho u_i \\ \rho u_i u_j + p \delta_{ij} \\ (\rho e + p) u_i \end{pmatrix}}_{\mathbf{F}_i^{\text{adv}}} + \underbrace{\begin{pmatrix} 0 \\ -\sigma_{ij} \\ -\rho u_j \sigma_{ij} - k T_{,i} \end{pmatrix}}_{\mathbf{F}_i^{\text{diff}}}, \quad S(\mathbf{Y}) = - \begin{pmatrix} 0 \\ \rho \mathbf{g} \\ 0 \end{pmatrix}$$



Compressible Navier-Stokes for Continuous-Galerkin FEM

Find $\mathbf{Y} \in \mathcal{S}^h$, $\forall \mathbf{v} \in \mathcal{V}^h$

$$\int_{\Omega} \mathbf{v} \cdot (\mathbf{A}_0 \mathbf{Y}_{,t} - \mathbf{S}(\mathbf{Y})) \, d\Omega + \int_{\Omega} \mathbf{v} \cdot \mathbf{F}_{i,i}(\mathbf{Y}) \, d\Omega$$



Compressible Navier-Stokes for Continuous-Galerkin FEM

Find $\mathbf{Y} \in \mathcal{S}^h$, $\forall \mathbf{v} \in \mathcal{V}^h$

$$\int_{\Omega} \mathbf{v} \cdot (\mathbf{A}_0 \mathbf{Y}_{,t} - \mathbf{S}(\mathbf{Y})) \, d\Omega - \int_{\Omega} \mathbf{v}_{,i} \cdot \mathbf{F}_i(\mathbf{Y}) \, d\Omega + \int_{\partial\Omega} \mathbf{v} \cdot \mathbf{F}_i(\mathbf{Y}) \cdot \hat{\mathbf{n}}_i \, d\partial\Omega$$



Compressible Navier-Stokes for Continuous-Galerkin FEM

Find $\mathbf{Y} \in \mathcal{S}^h$, $\forall \mathbf{v} \in \mathcal{V}^h$

$$\begin{aligned} \int_{\Omega} \mathbf{v} \cdot (\mathbf{A}_0 \mathbf{Y}_{,t} - \mathbf{S}(\mathbf{Y})) \, d\Omega - \int_{\Omega} \mathbf{v}_{,i} \cdot \mathbf{F}_i(\mathbf{Y}) \, d\Omega + \int_{\partial\Omega} \mathbf{v} \cdot \mathbf{F}_i(\mathbf{Y}) \cdot \hat{\mathbf{n}}_i \, d\partial\Omega \\ + \underbrace{\int_{\Omega} \mathcal{L}(\mathbf{v}) (\mathbf{A}_0 \mathbf{Y}_{,t} + \mathbf{F}_{i,i}(\mathbf{Y}) - \mathbf{S}(\mathbf{Y})) \, d\Omega}_{\text{SUPG}} = 0 \end{aligned}$$



Compressible Navier-Stokes for Continuous-Galerkin FEM

Find $\mathbf{Y} \in \mathcal{S}^h$, $\forall \mathbf{v} \in \mathcal{V}^h$

$$\begin{aligned} \int_{\Omega} \mathbf{v} \cdot (\mathbf{A}_0 \mathbf{Y}_{,t} - \mathbf{S}(\mathbf{Y})) \, d\Omega - \int_{\Omega} \mathbf{v}_{,i} \cdot \mathbf{F}_i(\mathbf{Y}) \, d\Omega + \int_{\partial\Omega} \mathbf{v} \cdot \mathbf{F}_i(\mathbf{Y}) \cdot \hat{\mathbf{n}}_i \, d\partial\Omega \\ + \underbrace{\int_{\Omega} \mathcal{L}(\mathbf{v}) (\mathbf{A}_0 \mathbf{Y}_{,t} + \mathbf{F}_{i,i}(\mathbf{Y}) - \mathbf{S}(\mathbf{Y})) \, d\Omega}_{\text{SUPG}} = 0 \end{aligned}$$

Simplify into residual form:

$$\mathcal{G}(\mathbf{Y}_{,t}, \mathbf{Y}) = 0$$



Compressible Navier-Stokes for Continuous-Galerkin FEM

Find $\mathbf{Y} \in \mathcal{S}^h$, $\forall \mathbf{v} \in \mathcal{V}^h$

$$\begin{aligned} \int_{\Omega} \mathbf{v} \cdot (\mathbf{A}_0 \mathbf{Y}_{,t} - \mathbf{S}(\mathbf{Y})) \, d\Omega - \int_{\Omega} \mathbf{v}_{,i} \cdot \mathbf{F}_i(\mathbf{Y}) \, d\Omega + \int_{\partial\Omega} \mathbf{v} \cdot \mathbf{F}_i(\mathbf{Y}) \cdot \hat{\mathbf{n}}_i \, d\partial\Omega \\ + \underbrace{\int_{\Omega} \mathcal{L}(\mathbf{v}) (\mathbf{A}_0 \mathbf{Y}_{,t} + \mathbf{F}_{i,i}(\mathbf{Y}) - \mathbf{S}(\mathbf{Y})) \, d\Omega}_{\text{SUPG}} = 0 \end{aligned}$$

Simplify into residual form:

$$\begin{aligned} \mathcal{G}(\mathbf{Y}_{,t}, \mathbf{Y}) &= 0 \\ \Rightarrow \quad \mathcal{P}^T \mathcal{E}^T \mathbf{B}^T \mathbf{G} \mathbf{B} \mathcal{E} \mathcal{P} \begin{bmatrix} \mathbf{Y}_{,t} \\ \mathbf{Y} \end{bmatrix} &= 0 \end{aligned}$$



Viscous Outflow Boundary Conditions

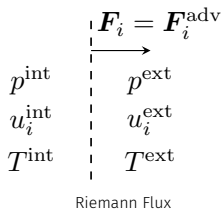


Viscous Outflow Boundary Conditions



Viscous Outflow Boundary Conditions

- Riemann Flux BCs everywhere is ideal

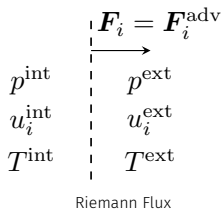


The diagram illustrates a Riemann flux boundary condition at a cell interface. A vertical dashed line represents the interface. To the left of the interface, the internal state variables are listed: p^{int} , u_i^{int} , and T^{int} . To the right, the external state variables are listed: p^{ext} , u_i^{ext} , and T^{ext} . A horizontal arrow points from the internal state to the external state, with the label $\mathbf{F}_i = \mathbf{F}_i^{\text{adv}}$ above it. Below the dashed line, the text "Riemann Flux" is written.



Viscous Outflow Boundary Conditions

- Riemann Flux BCs everywhere is ideal
- Velocity not known *a priori* \rightarrow traditional Riemann BCs not applicable

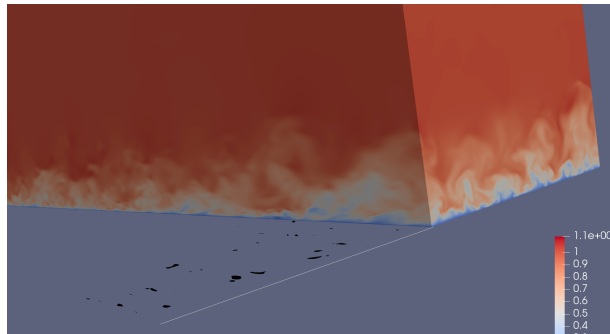


Viscous Outflow Boundary Conditions

- Riemann Flux BCs everywhere is ideal
- Velocity not known *a priori* \rightarrow traditional Riemann BCs not applicable
- Weak-pressure prescription ill-posed for recirculation

$$\begin{array}{c|c} p^{\text{int}} & \mathbf{F}_i = \mathbf{F}_i^{\text{adv}} \\ u_i^{\text{int}} & p^{\text{ext}} \\ T^{\text{int}} & u_i^{\text{ext}} \\ & T^{\text{ext}} \end{array}$$

Riemann Flux

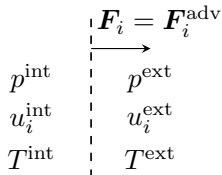


Viscous Outflow Boundary Conditions

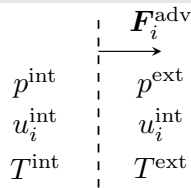
- Riemann Flux BCs everywhere is ideal
- Velocity not known *a priori* \rightarrow traditional Riemann BCs not applicable
- Weak-pressure prescription ill-posed for recirculation

Solution

- Calculate $\mathbf{F}_i^{\text{adv}}$ via Riemann solver, setting $u_i^{\text{ext}} = u_i^{\text{int}}$



Riemann Flux



Riemann Outflow Boundary Condition Flux

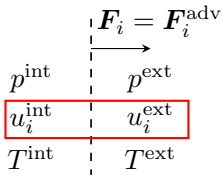


Viscous Outflow Boundary Conditions

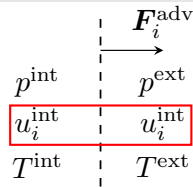
- Riemann Flux BCs everywhere is ideal
- Velocity not known *a priori* \rightarrow traditional Riemann BCs not applicable
- Weak-pressure prescription ill-posed for recirculation

Solution

- Calculate $\mathbf{F}_i^{\text{adv}}$ via Riemann solver, setting $u_i^{\text{ext}} = u_i^{\text{int}}$



Riemann Flux



Riemann Outflow Boundary Condition Flux

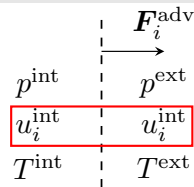
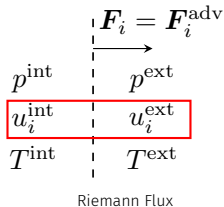


Viscous Outflow Boundary Conditions

- Riemann Flux BCs everywhere is ideal
- Velocity not known *a priori* \rightarrow traditional Riemann BCs not applicable
- Weak-pressure prescription ill-posed for recirculation

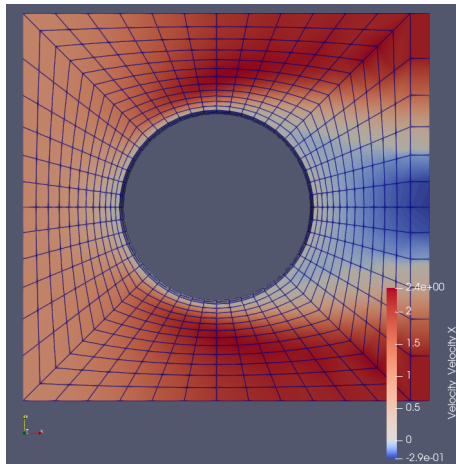
Solution

- Calculate $\mathbf{F}_i^{\text{adv}}$ via Riemann solver, setting $u_i^{\text{ext}} = u_i^{\text{int}}$
- Calculate $\mathbf{F}_i^{\text{diff}}$ from solution, then $\mathbf{F}_i = \mathbf{F}_i^{\text{adv}} + \mathbf{F}_i^{\text{diff}}$



Riemann Outflow Boundary Condition Flux

Viscous Outflow Boundary Conditions Demonstration



- Cylinder in cross flow torture test
- Stable for significant outflow recirculation
- Only p and T set at boundary



Efficient Implicit Timestepping



Implicit Timestepping

Implicit timestepping requires solving:

$$\frac{d\mathcal{G}(\mathbf{Y}_t, \mathbf{Y})}{d\mathbf{Y}} \Delta \mathbf{Y} = -\mathcal{G}(\mathbf{Y}_t, \mathbf{Y})$$



Implicit Timestepping

Implicit timestepping requires solving:

$$\frac{d\mathcal{G}(\mathbf{Y}_t, \mathbf{Y})}{d\mathbf{Y}} \Delta \mathbf{Y} = -\mathcal{G}(\mathbf{Y}_t, \mathbf{Y})$$

- System too large for direct solve \rightarrow iterative solve



Implicit Timestepping

Implicit timestepping requires solving:

$$\frac{d\mathcal{G}(\mathbf{Y}_t, \mathbf{Y})}{d\mathbf{Y}} \Delta \mathbf{Y} = -\mathcal{G}(\mathbf{Y}_t, \mathbf{Y})$$

- System too large for direct solve \rightarrow iterative solve
- Krylov iterative solvers used most commonly



Implicit Timestepping

Implicit timestepping requires solving:

$$\frac{d\mathcal{G}(\mathbf{Y}_t, \mathbf{Y})}{d\mathbf{Y}} \Delta \mathbf{Y} = -\mathcal{G}(\mathbf{Y}_t, \mathbf{Y})$$

- System too large for direct solve \rightarrow iterative solve
- Krylov iterative solvers used most commonly
- Krylov solvers form solution basis from $\text{span} \left\{ \left[\frac{d\mathcal{G}(\mathbf{Y}_t, \mathbf{Y})}{d\mathbf{Y}} \right]^n \Delta \mathbf{Y} \right\}_{n=0}$



Implicit Timestepping

Implicit timestepping requires solving:

$$\frac{d\mathcal{G}(\mathbf{Y}_t, \mathbf{Y})}{d\mathbf{Y}} \Delta \mathbf{Y} = -\mathcal{G}(\mathbf{Y}_t, \mathbf{Y})$$

- System too large for direct solve \rightarrow iterative solve
- Krylov iterative solvers used most commonly
- Krylov solvers form solution basis from $\text{span} \left\{ \left[\frac{d\mathcal{G}(\mathbf{Y}_t, \mathbf{Y})}{d\mathbf{Y}} \right]^n \Delta \mathbf{Y} \right\}_{n=0}$

Bottom Line

Cost of $\frac{d\mathcal{G}(\mathbf{Y}_t, \mathbf{Y})}{d\mathbf{Y}} \Delta \mathbf{Y}$ dominates implicit timestepping cost



Jacobian Matrix-Vector Multiply Options

How to compute $\frac{d\mathcal{G}(\mathbf{Y}_t, \mathbf{Y})}{d\mathbf{Y}} \Delta \mathbf{Y}$?



Jacobian Matrix-Vector Multiply Options

How to compute $\frac{d\mathcal{G}(\mathbf{Y}_t, \mathbf{Y})}{d\mathbf{Y}} \Delta \mathbf{Y}$?

- Store $\frac{d\mathcal{G}}{d\mathbf{Y}}$ directly (sparse matrix representation)
 - **Pros:** Opens up preconditioning options
 - **Cons:** Is large, expensive to store



Jacobian Matrix-Vector Multiply Options

How to compute $\frac{d\mathcal{G}(\mathbf{Y}_t, \mathbf{Y})}{d\mathbf{Y}} \Delta \mathbf{Y}$?

- Store $\frac{d\mathcal{G}}{d\mathbf{Y}}$ directly (sparse matrix representation)
 - **Pros:** Opens up preconditioning options
 - **Cons:** Is large, expensive to store
- Finite difference matrix-free approximation:

$$\frac{d\mathcal{G}(\mathbf{Y}_t, \mathbf{Y})}{d\mathbf{Y}} \Delta \mathbf{Y} \approx \frac{\mathcal{G}(\mathbf{Y}_t, \mathbf{Y} + \epsilon \Delta \mathbf{Y}) - \mathcal{G}(\mathbf{Y}_t, \mathbf{Y})}{\epsilon}$$

- **Pros:** Just need a residual evaluation, cheap (in programming and computation)
- **Cons:** Accuracy limited to $\sqrt{\epsilon_{\text{machine}}}$, preconditioning require partial assembly



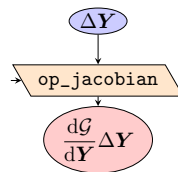
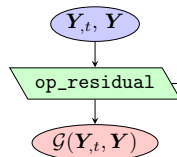
Exact Matrix-Free Jacobian via CeedOperator

$$\begin{aligned}\frac{d\mathcal{G}}{d\mathbf{Y}}\Delta\mathbf{Y} &= \frac{d}{d\mathbf{Y}} \left[\overbrace{\mathcal{P}^T \mathcal{E}^T B^T G B \mathcal{E} \mathcal{P}}^{\mathcal{G}(\mathbf{Y},t,\mathbf{Y})} \right] \Delta\mathbf{Y} \\ &= \left[\mathcal{P}^T \mathcal{E}^T B^T \frac{dG}{d\mathbf{Y}} B \mathcal{E} \mathcal{P} \right] \Delta\mathbf{Y}\end{aligned}$$



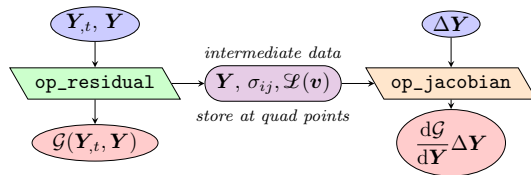
Exact Matrix-Free Jacobian via CeedOperator

$$\begin{aligned}\frac{d\mathcal{G}}{d\mathbf{Y}}\Delta\mathbf{Y} &= \frac{d}{d\mathbf{Y}} \left[\overbrace{\mathcal{P}^T \mathcal{E}^T B^T G B \mathcal{E} \mathcal{P}}^{\mathcal{G}(\mathbf{Y},t,\mathbf{Y})} \right] \Delta\mathbf{Y} \\ &= \left[\mathcal{P}^T \mathcal{E}^T B^T \frac{dG}{d\mathbf{Y}} B \mathcal{E} \mathcal{P} \right] \Delta\mathbf{Y}\end{aligned}$$



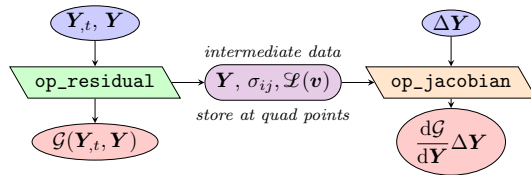
Exact Matrix-Free Jacobian via CeedOperator

$$\begin{aligned}\frac{d\mathcal{G}}{d\mathbf{Y}}\Delta\mathbf{Y} &= \frac{d}{d\mathbf{Y}} \left[\overbrace{\mathcal{P}^T \mathcal{E}^T B^T G B \mathcal{E} \mathcal{P}}^{\mathcal{G}(\mathbf{Y}_t, \mathbf{Y})} \right] \Delta\mathbf{Y} \\ &= \left[\mathcal{P}^T \mathcal{E}^T B^T \frac{dG}{d\mathbf{Y}} B \mathcal{E} \mathcal{P} \right] \Delta\mathbf{Y}\end{aligned}$$



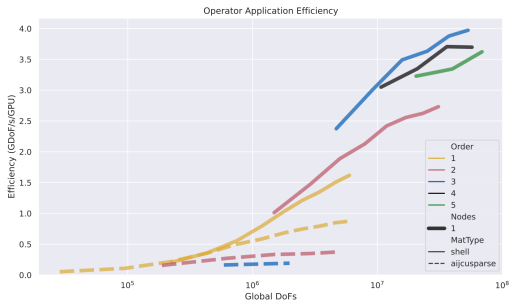
Exact Matrix-Free Jacobian via CeedOperator

$$\begin{aligned}\frac{d\mathcal{G}}{d\mathbf{Y}}\Delta\mathbf{Y} &= \frac{d}{d\mathbf{Y}} \overbrace{\left[\mathcal{P}^T \mathcal{E}^T \mathcal{B}^T \mathcal{G} \mathcal{B} \mathcal{E} \mathcal{P} \right]}^{\mathcal{G}(\mathbf{Y}_t, \mathbf{Y})} \Delta\mathbf{Y} \\ &= \left[\mathcal{P}^T \mathcal{E}^T \mathcal{B}^T \frac{d\mathcal{G}}{d\mathbf{Y}} \mathcal{B} \mathcal{E} \mathcal{P} \right] \Delta\mathbf{Y}\end{aligned}$$



- **Pros:** Exact Jacobian matrix-vector product (potentially faster convergence)
- **Cons:** Preconditioning requires partial assembly

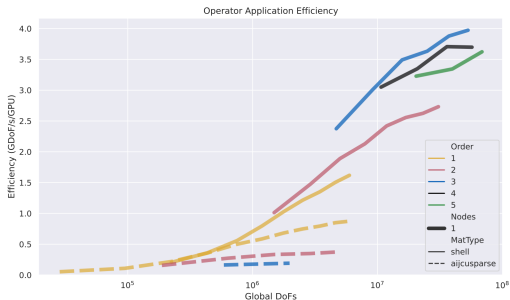
Performance Exact Matrix-Free Jacobian via CeedOperator



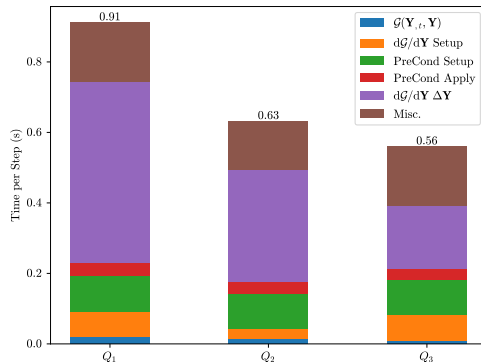
Matrix-Free Application Comparison (Reproduced from Brown *et al.* 2022).
Not from fluids code, but representative of libCEED matrix-free



Performance Exact Matrix-Free Jacobian via CeedOperator



Matrix-Free Application Comparison (Reproduced from Brown *et al.* 2022).
Not from fluids code, but representative of libCEED matrix-free



Fluids performance, DoFs fixed.
Run on ALCF's Polaris on two nodes (8 \times NVIDIA A100)



This work was supported by.... Add in sponsor support (ALCF, FastMATH DOE, ECP, etc)

