Automatic PETSc Code Generation for Finite Differences

Zoe Leibowitz, Rhodri Nelson, Fabio Luporini, Mathias Louboutin, Matthew G. Knepley, Lawrence Mitchell, Gerard Gorman

Imperial College London

PETSc

DEVITO

1. Motivation and Goals

- Porting traditional solvers onto new hardware is difficult and time-intensive.
- An effective strategy is to raise the level of abstraction by using **domain-specific languages** (DSLs).
- **Devito** [1,2] is a DSL and compiler for the automated generation of optimised **finite differences** across several computer platforms, supporting explicit time marching schemes.
- Initially focused on seismic inversion problems, Devito is broadening its scope to tackle challenges in **Computational Fluid Dynamics** (CFD).
- Core Goal: Integrate **matrix-free** routines into Devito to automate the execution of PETSc's [3] iterative solvers, facilitating support for **implicit** kernels.

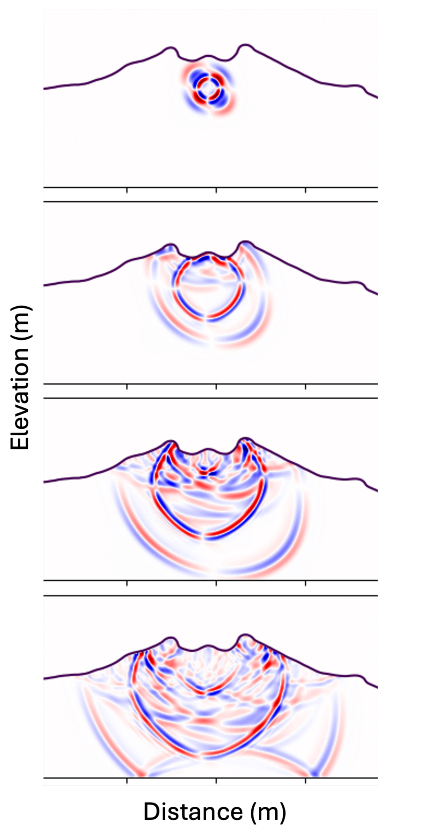

Figure 1: Anisotropic elastic wave propagation featuring immersed free-surface topography in Devito.

https://pixabay.com/photos/wind-energy-wind-turbines-windmills-7394705/

2. Devito - A Brief Introduction

Devito allows users to generate optimised solvers in a few lines of symbolic Python. Below is an example of how to build a simple 2D diffusion operator.

Step 1: Discretise computational domain via the `Grid` object.

```
# 1x1 grid with 11x11 nodes.
grid = Grid(shape=(11, 11), extent=(1., 1.))
```

Step 2: Encapsulate field data in `TimeFunction`.

```
# space_order specifies the discretisation order.
f = TimeFunction(name='f', grid=grid, space_order=2)
```

Step 3: Create symbolic expression of PDE using `Eq` object.

```
eqn = Eq(f.dt, 0.5 * f.laplace)
```

Step 4: Rearrange to represent valid state update using `solve` object.

```
update = Eq(f.forward, solve(eqn, f.forward))
```

Step 5: Use `Operator` object to generate low-level C kernel through a sequence of **compilation passes**.

```
op = Operator(update)
```

Step 6: Just-in-time (JIT) compile and execute.

```
op(t=timesteps, dt=dt)
```

Low-level C loop structure automatically generated:

```
for (int time = time_m, t0 = (time)%(2), t1 = ...)
{
  for (int x = x_m; x <= x_M; x += 1)
  {
    for (int y = y_m; y <= y_M; y += 1)
    {
     f[t1][x + 2][y + 2] = dt*(r2*f[t0][x + 2][y + 2]
     + (-1.0F)*(r0*f[t0][x + 2][y + 2] + ... ;
```

3.1. Proof of Concept - Setup

- We solve the **2D lid-driven cavity flow** problem. The Devito API is extended to efficiently solve for the pressure field at each time step with an **iterative solver**.
- The governing equations are the 2D incompressible Navier-Stokes equations in primitive variables. Two equations govern the velocity components u, v and one equation governs the pressure p:

$$\frac{\partial u}{\partial t} + u\frac{\partial u}{\partial x} + v\frac{\partial u}{\partial y} = -\frac{1}{\rho}\frac{\partial p}{\partial x} + \nu\left[\frac{\partial^2 u}{\partial x^2} + \frac{\partial^2 u}{\partial y^2}\right], \tag{2.1}$$

$$\frac{\partial v}{\partial t} + u\frac{\partial v}{\partial x} + v\frac{\partial v}{\partial y} = -\frac{1}{\rho}\frac{\partial p}{\partial y} + \nu\left[\frac{\partial^2 v}{\partial x^2} + \frac{\partial^2 v}{\partial y^2}\right], \tag{2.2}$$

$$\frac{\partial^2 p}{\partial x^2} + \frac{\partial^2 p}{\partial y^2} = \rho\left[\frac{\partial}{\partial t}\left(\frac{\partial u}{\partial x} + \frac{\partial v}{\partial y}\right) - \left(\frac{\partial u}{\partial x}\frac{\partial u}{\partial x} + 2\frac{\partial u}{\partial y}\frac{\partial v}{\partial x} + \frac{\partial v}{\partial y}\frac{\partial v}{\partial y}\right)\right], \tag{2.3}$$

  where ρ is the density and ν is the kinematic viscosity.
- The domain is given by $\Omega = (x, y) \in (0,1) \times (0,1)$, p=0 at $x = y = 0$, and the boundary conditions are as follows:

$$x = 0, 1 \quad 0 \le y \le 1 \quad u = v = \frac{\partial p}{\partial x} = 0, \qquad y = 0 \quad 0 \le x \le 1 \quad u = v = \frac{\partial p}{\partial y} = 0, \qquad y = 1 \quad 0 \le x \le 1 \quad u = 1, \quad v = \frac{\partial p}{\partial y} = 0.$$

3.2. Proof of Concept - API

As demonstrated in Section 2, the `Grid` object is used to setup the discretised domain. The fields u, v, and p are encapsulated within the DSL via `Function/TimeFunction` objects. Then, we symbolically express equations 2.1–2.3 as follows:

```
eq_u = Eq(u.dt + u*u.dx + v*u.dy, -1./rho * p.dxc + nu*(u.laplace))
eq_v = Eq(v.dt + u*v.dx + v*v.dy, -1./rho * p.dyc + nu*(v.laplace))
eq_p = Eq(p.laplace,rho*(1./dt*(u.dxc+v.dyc)-(u.dxc*u.dxc)+2*(u.dyc*v.dxc)+(v.dyc*v.dyc)))
```

These equations are then rearranged to denote a valid state update for each field. The velocities u and v are updated explicitly in time.

```
update_u = Eq(u.forward, solve(eq_u, u.forward))
update_v = Eq(v.forward, solve(eq_v, v.forward))
```

We employ a new API object, `PETScSolve`, to trigger the lowering to PETSc and iteratively solve for the pressure field p.

```
# The solver and preconditioner types are specifed using the solver_parameters argument.
update_p = PETScSolve(eq_p, p, solver_parameters={'ksp_type': 'gmres', 'pc_type': 'jacobi'})
```

Similarly to Section 2, a Devito `Operator` is created by passing in the update expressions. Following this, the code is JIT compiled and executed.

Note: Implementation details for boundary conditions are omitted for conciseness.

3.3. Proof of Concept - Low level code

Snippet of the generated C code solving the lid-driven cavity flow problem:

```
PetscCall(DMSetMatType(da,MATSHELL));
PetscCall(DMCreateMatrix(da,&A));
PetscCall(MatShellSetContext(A,ctx));
...
PetscCall(KSPCreate(PETSC_COMM_WORLD,&ksp));
PetscCall(KSPSetType(ksp,KSPGMRES));
...
PetscCall(MatShellSetOperation(A,MATOP_MULT,..matvec);
for (int time = time_m, t0 = (time)%(2), t1 = ...)
{ ...
  PetscCall(KSPSolve(ksp,b,p));
  for (int x = x_m; x <= x_M; x += 1)
  {
    for (int y = y_m; y <= y_M; y += 1)
    {
     u[t1][x + 2][y + 2] = dt*(nu*(u[t0][x + ...;
     v[t1][x + 2][y + 2] = dt*(nu*(v[t0][x + ...;
    }
  }
  ...
}
```

Snippet of the (matrix-vector) callback used in solving equation 2.3 (utilising a matrix-free method).

```
PetscErrorCode matvec(Mat A, Vec x, Vec y)
{
  ...
  PetscScalar** x_arr;
  PetscScalar** y_arr;
  struct MatContext * ctx;
  PetscCall(MatShellGetContext(A,&ctx));
  ...
  PetscCall(DMDAVecGetArrayRead(da,x_local,&x_arr));
  ...
  for (int x = ctx->x_m; x <= ctx->x_M; x += 1)
  {
    for (int y = ctx->y_m; y <= ctx->y_M; y += 1)
    {
      y_arr[x][y] = -2.0F*pow(ctx->h_x,-2)*x_arr[x][y]
      + pow(ctx->h_x,-2)*x_arr[x - 1][y] +
      pow(ctx->h_x,-2)*x_arr[x + 1][y] + ...;
    }
  }
  ...
}
```

3.4. Proof of Concept - Validation

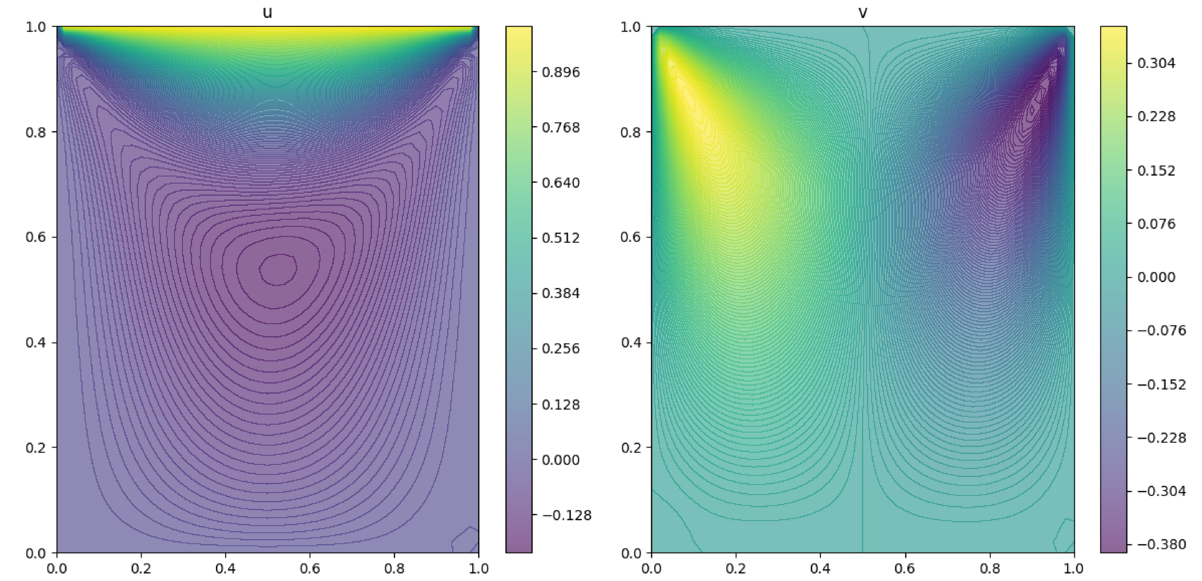

Figure 2: Contour plots of horizontal velocity u (left) and vertical velocity v (right).

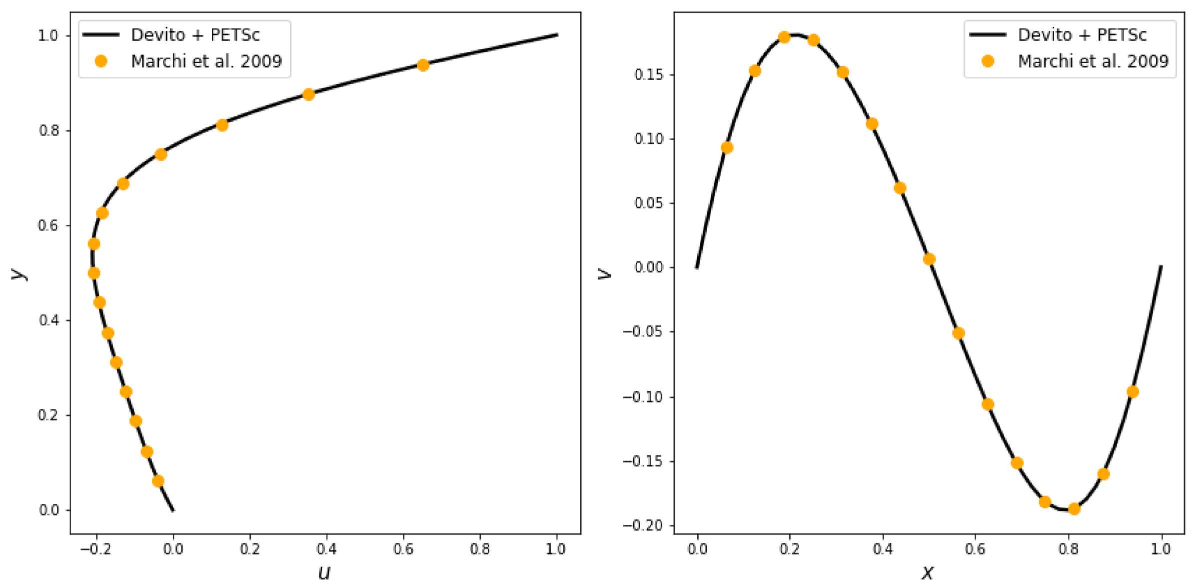

Figure 3: Validtion - Comparing Devito + PETSc solution with Marchi et al.(2009) [4]. u at x=0.5 (left), v at y=0.5 (right).

4. Future Direction

- Extend the application areas of Devito to CFD based problems such as simulating fluid flow in the context of wind turbines.
- Optimise the Devito compiler such that it can generate code that beats hand-written CFD code.
- Efficient solvers in the realm of CFD will involve the implementation of scalable non-linear solvers (via the SNES library [3]) and support for, e.g, multi-grid methods (using PCMG [3]).

https://pixabay.com/photos/wind-energy-wind-turbines-windmills-7394705/

5. References

[1] Louboutin, Mathias, et al. "Devito (v3. 1.0): an embedded domain-specific language for finite differences and geophysical exploration." Geoscientific Model Development 12.3 (2019): 1165-1187. (2018).
[2] Luporini, Fabio, et al. "Architecture and performance of Devito, a system for automated stencil computation." ACM Transactions on Mathematical Software (TOMS) 46.1 (2020): 1-28.
[3] PETSc Web Page. Satish Balay et al. 2023. [Online]. Available: https://petsc.org/
[4] Carlos Henrique Marchi, et al. The lid-driven square cavity flow: numerical solution with a 1024 x 1024 grid. Journal of the Brazilian Society of Mechanical Sciences and Engineering, 31:186–198, 2009.

Email Slack GitHub Website