Title Slide

Good morning, everyone. Thank you for coming to my presentation. My name is Truman Everett Ellis, but you can call me Everett. I got my PhD in computational science engineering and mathematics from UT Austin early 2016 and I'm currently working on computational plasma physics at Sandia National Lab. But in this talk I wanted to briefly summarize my PhD work on Automating Fluid Simulations with Discontinuous Petrov-Galerkin Finite Elements.

Table of Contents

I'll begin by motivating some current bottlenecks with computational fluid dynamics then look at what the DPG method is and how it might address those.
From there I'll illustrate a simple DPG formulation for the convection-diffusion equation and then extrapolate to the compressible Navier-Stokes equations with a few example problems. Then we'll close with some conclusions.

Navier-Stokes Equations

Basically what I want to say on this slide is that Navier-Stokes is hard. Despite the decades of research that have been invested in simulating fluid flows, robust simulation is still a challenge.

- Numerical methods need to resolve a wide variety of flow features from shocks to boundary layers to turbulence.
- On top of that, most numerical methods have minimum mesh resolution requirements before convergence can be guaranteed. Meshes that are too coarse for the numerical method to converge on are termed to be in the pre-asymptotic regime. This means that in some sense, the mesh for the simulation to the phenomena on the right needs to anticipate the flow features before they are known.

Initial Mesh Design

This requirement can present additional challenges to the CFD practitioner designing a mesh. At the minimum level, the surface mesh needs to adequately represent the physical geometry of the problem under consideration. But the stability of the numerical methods presents additional requirements on the volume mesh surrounding the surface mesh. This often leads to a trial and error process for the CFD engineer.

DPG on Coarse Meshes

My goal with this research is to remove one of those constraints on computational mechanics practitioners. In contrast to most other methods out there, the DGP method does not suffer convergence issues on coarse meshes. We are able to start on the coarsest mesh possible which sufficiently resolves the geometry and adaptively refine towards a resolved solution. These are results from Jesse Chan's thesis on supersonic flow over a flat plate. This is not a remarkably difficult problem, but what is notable is that we were able to start the simulation on a mesh of only two elements, sit back, and let the method take over as it solved, refined, and converged to a resolved solution after 11 adaptive steps.

Lessons from Other Methods

Before I jump into the particulars of DPG, I wanted to briefly touch on some of the methods that influenced our work and the lessons we learned from them.

- The streamline Upwind Petrov-Galerkin method was really the first finite element method to successfully solve fluid problems. Put simply, SUPG was able to solve convection-diffusion type problems by adaptively upwinding the test functions based on the elements Peclet number. This introduced the idea that you could improve the stability of a finite element method by modifying your test space.
- Despite the similar name, DG methods only really have one significant influence on the DPG method. The realization that discontinuous basis functions or broken Sobolev spaces were fair game for finite element methods allowed the DPG method to progress from being an interesting trick to a practically computable method.
- Hybridized DG, or HDG is a newer variation of the DG method that seeks to address the main criticism of DG methods: that of proliferation of unknowns. For the lowest order DG method on a 3D hex mesh, the global solve will have 8 times as many unknowns as an equivalent continuous Galerkin method. HDG addresses this by adding new interface unknowns that couple elements together. Internal dofs no longer directly communicate with each other, and can be statically condensed out of the global solve, resulting in a global solve of comparable size to a continuous discretization. DPG likewise makes use of interface unknowns and static condensation.
- Least-Squares finite element methods attempt to turn any PDE into a least-squares solve, bypassing the LBB inf-sup stability conditions and producing a symmetric, positive definite global stiffness matrix. The insight is that finite element methods are most powerful in a Ritz type framework. DPG can be interpreted as a generalized least-squares method where instead of minimizing the residual in L2, we minimized it in a user-defined dual norm.
- Finally, space-time finite elements were proposed as a means to handle some of the disadvantages to attempting to time step a highly adapted mesh. Analysis shows that a unified treatment of space and time within one method can produce superior results for moving boundary problems, but producing a method which is stable both spatially and temporally has historically been a challenge.

Overview of DPG

Other Features

A few other interesting features of the DPG method.

The test space so far has not been defined, but if we were to use continuous test functions, then the auxiliary problem would turn into a global solve of similar complexity to the original problem. Introducing a broken test space localizes these solves making it embarrassingly parallel. The downside is that this introduces additional interface unknowns, which I will illustrate later with an example.

As a minimum residual method, the method always produces a hermitian positive definite stiffness matrix (SPD for real valued problems). We haven't really explored the implications for iterative solvers, but this seems like it would be a positive feature.

We can also evaluate the residual error without knowing the exact solution. This can be used to robustly drive adaptivity.

Space-Time DPG

But before I put some equations down, let's briefly explore why we might want to work in a space-time framework rather than a traditional finite difference based time stepper. Adaptivity has been an integral part of DPG from day one, and we typically get refined elements which are several orders smaller magnitude than other elements in the mesh. Classical time stepping techniques propagate the entire solution forward in lockstep at the pace of the most restrictive element. Implicit techniques allow you to take larger steps but for the sake of temporal accuracy, you may not want to. Now, there are techniques out there that do allow different elements to proceed at different time steps, such as asynchronous variational integrators, but for us, this is not an ideal solution. Bolting on a different temporal integrator to a DPG spatial integrator produces a kind of Frankenstein of properties. We know we have great stability in space, but where does that leave us temporally? If we instead decide to just treat time as another dimension to be discretized with a DPG method, then we get a unified treatment and preserve all of our nice stability and adaptivity properties. We get automatic local time stepping and a kind of parallel-in-time integration as we can solve an entire time slab at once, distributing different space-time elements within the slab to different processors. I mentioned that space-time presents a challenge for classical finite elements as equations may have different spatial and temporal characteristics, but DPG addresses this concern. Is your space-time formulation well-posed? Yes, then we are in business. The big complication is on the computational and implementation side. Your code now has to support higher dimensional

meshes or as we are implementing, a kind of tensor product of spatial and temporal elements.

Space-Time DPG for Convection-Diffusion

The heat equation is the simplest transient fluid problem we can think of. As we will see later, it actually serves as a decent model problem for something as complicated as compressible Navier-Stokes. Just like Navier-Stokes, it is parabolic in space-time.
I am going to derive one variational problem for this equation, but there are other choices we could make. In particular, early in DPG's history, we decided to focus on the ultra-weak first order formulation because we believed it presented the most flexibility, but increasingly we have been working with the original second order problem.
So as we decompose this into a system of first order equations, we get a constitutive law in space and conservation of mass. The key insight is that we can rewrite our conservation law as a space-time divergence operator.

Ultra-Weak Formulation

This is still in strong form, so let's multiply by test functions tau and v, and integrating by parts over each element. IBP of the stress equation introduces a trace unknown u hat.
If we were working with a conforming test space, the integration of this term against tau dot n over element boundaries would cancel, but the cost of a discontinuous test space is new interface unknowns.
Similarly, IBP of the conservation equation introduces flux t hat. The trace only exists on spatial boundaries because the stress equation is only integrated by parts over spatial dimensions. Note that the flux changes nature depending on whether it is on a temporal or spatial boundary or a mix of the two.

Space-Time Compressible Navier-Stokes

I'll get into some of our results applying the space-time methodology to compressible Navier-Stokes. I mentioned that space-time support in Camellia is still experimental, so the numerical results so far are limited to 1D space + time, but we are on the cusp of getting a tensor topology thing working for 2D and 3D computations. That is basically the last thing I need to do to graduate, actually. But here are the preliminary 1D results.

First Order System

One interesting thing about the DPG framework is that it takes stability out of the question. One long standing issue when it comes to simulating fluid flow is whether it's best to work with

primitive variables, conservation variables, or entropy variables.
Primitive variables have the benefit of being simple, intuitive and the least nonlinear formulation.
Conservation variables make time stepping much more straightforward.
And Entropy variables were introduced in the belief that a symmetric system was best.
I'm going to perform a simple comparison of these three formulations within our space-time DPG framework.

For the sake of simplicity, we assume Stokes hypothesis, ideal gas, and constant viscosity, but if we wanted to choose something more realistic for these, it would just make the linearization process a little messier. We start by splitting up our conservation equations into a first order system as we did with convection-diffusion, but here we jump straight into the space-time divergence form. Since the Navier-Stokes equations are incompletely parabolic, we only get 2 constitutive laws to the 3 conservation equations.

Compact Notation

We now introduce compact notation defining the conserved quantities, the inviscid fluxes, the viscous fluxes, viscous terms, and viscous relations.
At this point we could perform a change of variables to either conservation or entropy variables, but I'm going to hide those details in this talk.

Define Group Variables

We further compact our notation by introducing group variables. We can now write our Navier-Stokes problem in a way that we can easily draw analogies to convection-diffusion.
I'm going to skip the details, but we then linearize this and develop test norms.

Sod Problem

We start with everyone's favorite shock tube problem, the Sod problem. This is a spatially 1D problem from 0 to 1 with a final time of 0.2. So we introduce a starting space-time mesh with 4 quadratic elements where the y=0 axis corresponds to the initial conditions and y=0.2 is the final time. Now we are going to track the refinement process as the method automatically resolves the solution features. After 14 refinements, we've pretty much gotten our mesh down to the viscous scale along the shock.
I want to point out that we are now getting to the point where we can apply a multigrid solution strategy to the series of adaptive meshes that we generate here.