Introduction

The goal of my intended project is to use simplified physics to simulate performance of an high-speed aircraft. An unguided hypersonic glide vehicle (HGV), will be used as the exemplar case to implement shape optimization. HGVs are vehicles that travel in excess of 5 times the speed of sound and due to their high speeds experience extreme temperatures. As a result, highly accurate simulations of the physics at these conditions requires solving the set of non-linear compressible Navier-Stokes (NS) equations with additional source terms for chemical reactions. Solving these set of equations requires discretizing a fluid domain and applying a stencil to reconstruct fluxes on the 3D grid. This incurs a huge computational cost and is cost prohibitive for most applications in design and optimization. Instead, simplified physics models are used for initial design work such as Newtonian aerodynamics (NA). Therefore, in this proposal i will seek to implement NA in shape optimization of HGVs to gain underlining performance metrics for a given vehicle design. The goal is to gain significant speed up to allow for rapid design of a parameter space for vehicle performance.

Description

The goal is to use a RL algorithm to preform guidance on an HGV such that $\gamma = 0$.

As part of this task the simulated environment will be offloaded on the GPU for speed-ups vs the serial code.

Step 1: Simulation

A geometry will be tessellated into several hundred-thousand triangles.

From which NA will be applied to compute the forces:

$N =$ number of triangles elements

procedure NA(α)

$i \leftarrow 1$

$\hat{v} = \{\cos(\alpha), sin(\alpha), 0\}$

Forces $=\{0,0,0\}$

while $i \leq N$ **do**

if $(\hat{v} \cdot \hat{n}_i) < 0$ **then**

$cp_i = 2\left(\hat{v} \cdot \hat{n}_i\right)^2$

else

$cp_i = 0$ ▷ Flow is hidden from this surface

end if

$P_i = (cp_i \cdot q) + P_\infty$

$F = F + (P_i \cdot A_i) \cdot (-\hat{n}_i)$ ▷ A_i computed from cross product

$i \leftarrow i + 1$

end while

$D = F(2)\sin(\alpha) + F(1)\cos(\alpha)$

$L = F(2)\cos(\alpha) - F(1)\sin(\alpha)$

end procedure

Once we have the forces in terms of lift (L) and drag (D), we can compute performance metrics such as max distance using ordinary differential equations (ODE's) for the equation of motions. The equation of motion for a lifting surface on a spherical earth is as follows:

$$\frac{\partial}{\partial t}\begin{bmatrix}\gamma\\ V\\ x\\ h\end{bmatrix} = \begin{bmatrix}\frac{1}{V}\left[\frac{L}{m} + g\cos(\gamma) - \frac{V^2}{R}\cos(\gamma)\right]\\ \frac{D}{m} + g\sin(\gamma)\\ -V\cos(\gamma)\\ -V\sin(\gamma)\end{bmatrix} \tag{1}$$

We then integrate the equation of motion listed above with a standard RK-4 method to arrive at a new state.

The goal is to record these trajectories under a given policy π then preform policy improvement

to select an α that will make $\dot{\gamma} = 0$.

GPU kernels

The following kerenel will need to be done to implement the following in CUDA C:

1. A kerenel to compute L,D from the NA algorithm.
2. A kerenel to reduce the L,D to a scalar value within the NA algorithm.
3. A kerenel to take the reduced L,D and preform RK-4 integrate.
4. A kerenel to take the new states and record to a file output.

Algorithmicaly this can shown as:

$S_0 = \{\gamma_0, v_0, x_0, h_0\}$

while s is feasible **do**

NA kernel

RK-4 kernel

I/O kernel

end while

Future goals would to solve the full PDE for the compressible NS equations on the GPU. However, this in my opinion is too ambitious for the given time. If you have an open source code we can work from that's numerically stable and has implicit time integration so that run times for a single trajectory aren't more than 48 hours then let me know!

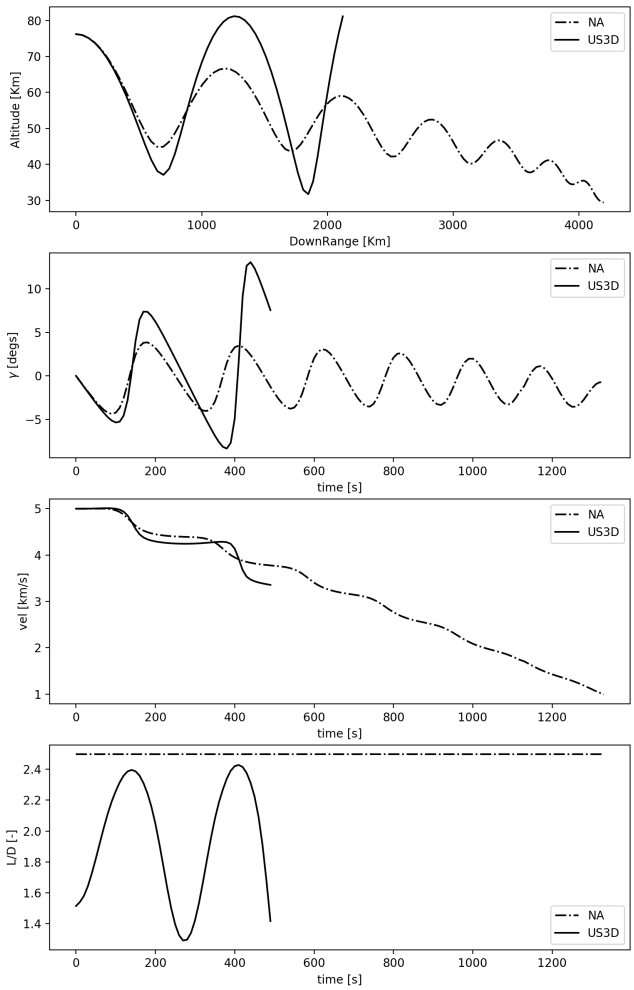
Resources and contact

I can provide explanations and examples. If you work in controls or want to help in any way reach out to the following:

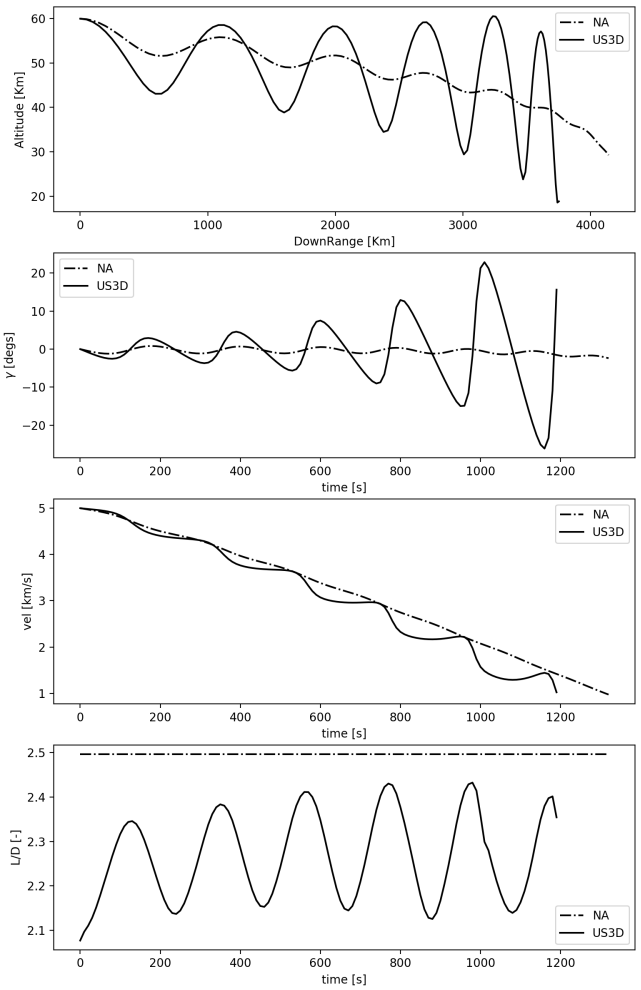
1. email: will7322@umn.edu

2. phone: 478-251-2267

Appendix



(a) vel = 5km/s, alt=76.2 km, α=20.05



(b) vel = 5km/s, alt=60.0 km, α=20.05