Numerical study of the airflow over a high-altitude pseudo-satellite wing

Adjoint Implementation

Carlo Brunelli

January 17, 2024

RMA Royal Military Academy Brussels

VUB VRIJE UNIVERSITEIT BRUSSEL

von KARMAN INSTITUTE FOR FLUID DYNAMICS

Sections

- Implementation
- Test Cases
- Considerations

Royal Military Academy

Continuous Adjoint

1 Implementation

- Continuous Adjoint: equations available
- Steady (an unsteady version is also implemented, is just faster for validation)
- Stabilized: SUPG stabilization - same order element interpolation

Royal Military Academy

Procedure

Define the objective function I to minimize (eg: Drag coefficient)

```
function compute_drag(uh,ph,nΓ,dΓ,params)
    CD, _ = compute_airfoil_coefficients(uh,ph,nΓ,dΓ,params)
    return CD, CD
end

function compute_airfoil_forces(uh,ph,nΓ,dΓ,params::Dict{Symbol,Any})
    @unpack tagname,ν = params
    IForce = ∫(-ph ⋅ nΓ + ν* transpose(∇(uh)) ⋅ nΓ)dΓ
    D,L = sum(IForce)
    return D,L
end
```

Procedure

1 Implementation

Solve the primal Flow

```
function primal_steady_SUPG(params::Dict{Symbol,Any})
@unpack ν, dt, dΩ, D, Ω, θ,uh = params
h = h_param(Ω, D)
updatekey(params, :h,h)
a((u, p), (v, q)) = ∫(ν * ∇(v) ⊙ ∇(u) + ∇(p) ⊙ v + q * (∇ ⋅ u))dΩ + ∫(v ⊙ (conv ∘ (uh, ∇(u))))dΩ
Rm(u, p) = conv ∘ (uh, ∇(u)) + ∇(p)
astab((u, p), (v, q)) = ∫( τsu(uh, h,ν,dt) ⋅ (uh ⋅ ∇(v) + ∇(q)) ⊙ Rm(u, p))dΩ
+ ∫( τb(uh, h,ν,dt) ⋅ (∇ ⋅ v) ⊙ (∇ ⋅ u))dΩ
res_prim((u, p), (v, q)) = a((u, p), (v, q)) + astab((u, p), (v, q))
return res_prim
end
```

Solve the adjoint Flow

```
function adjoint_steady_SUPG(params::Dict{Symbol,Any})
  @unpack ν, dt, dΩ, D, Ω, θ,uh = params
  h = h_param(Ω, D)
  updatekey(params,:h,h)
  Rmadj(u, p) = -uh ⋅ ∇(u) - ∇(u) ⋅ uh - ∇(p)
  Rcadj(u) = -1 .* (∇ ⋅ (u))
  a_adj((u, p), (v, q)) = ∫(ν * ∇(v) ⊙ ∇(u) - q * (∇ ⋅ u))dΩ + ∫(v ⊙ Rmadj(u, p))dΩ
  a_adj_stab((u, p), (v, q)) = ∫( τsu(uh, h,ν,dt) ⋅ (-uh ⋅ ∇(v) - ∇(v) ⋅ uh - ∇(q))
              ⊙ Rmadj(u, p))dΩ + -1 * ∫( τb(uh, h,ν,dt) ⋅ (∇ ⋅ v) ⊙ Rcadj(u))dΩ
  res_adj((u, p), (v, q)) = a_adj((u, p), (v, q)) + a_adj_stab((u, p), (v, q))
  return res_adj

end
```

Parametrization

1 Implementation

The airfoil is parametrized using Class Shape Transformations (CST), to obtain easily arifoil-like shapes. We start from from a classic NACA0012 airfoil. Identified by CST weights for top and bottom surface:

```
wu = [0.17085744048372403, 0.15451770272052714, 0.15810479767059346, 0.1359573306370066,
 0.14221259398479932, 0.14001089993692417]
wl = -1 .* [0.17085744048372403, 0.15451770272052714, 0.15810479767059346, 0.135957330637006
0.14221259398479932, 0.14001089993692417]
```

6 parameters for the top, and 6 for the bottom surface: β_i, $i = 1, ..., 12$, $i = 1, ...6$: top, $i = 7, ...12$ bottom.

Computing Reference values

Both primal and adjoint flow are solved on the original mesh Ω.

$$I = I_c + \int_\Omega \Psi \cdot \mathcal{R} = I_{1,ref} + I_{2,ref}$$

If we want to minimize C_D: $I_c = C_D$

```
I2 = sum(∫(ϕu⋅((∇(uh))'⋅uh + ∇(ph)))dΩ + ∫(ϕp⋅(∇⋅(uh)))dΩ)
```

Morphing mesh

1. A design variable β_i is perturbed by ϵ: $\beta_i + \epsilon$
2. The airfoil boundary and the mesh move accordingly obtaining a new Ω' domain
3. Evaluate

$$I' = I'_c + \int_{\Omega'} \Psi \cdot \mathcal{R} = I_{1,i} + I_{2,i}$$

4. The sensitivity is obtained through

$$\frac{\partial I}{\partial \beta_i} = \frac{I' - I}{\epsilon}$$

Morphing mesh

1 Implementation

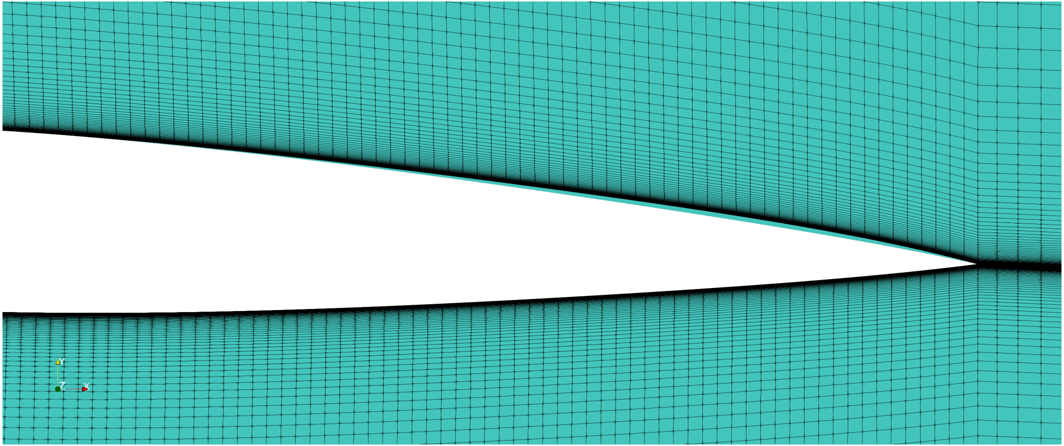


Figure: Ω in light blue, morphed mesh Ω' in black. Shift is emphasized for demonstration.

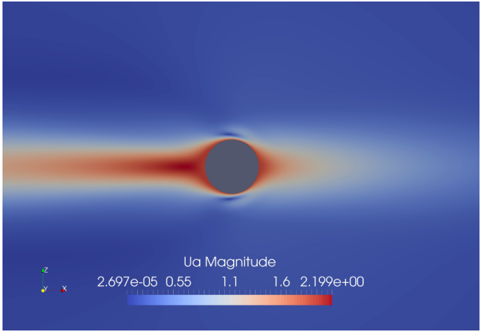
Is the Adjoint Flow Solved Correctly?

Not many quantitative solutions of the adjoint field are available. From[1] 2 adjoint flows are reported.

[1] Giuseppe Sorgiovanni, Maurizio Quadrio, and Raffaele Ponzini. "A robust open-source adjoint optimization method for external aerodynamics". In: (2016).

Cylinder Reynolds 50

2 Test Cases



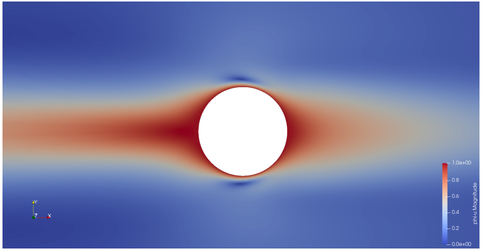
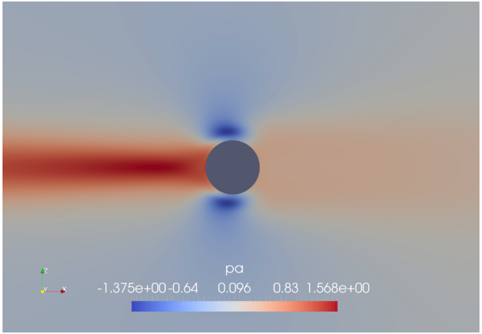


Figure: Adjoint Velocity

Cylinder Reynolds 50



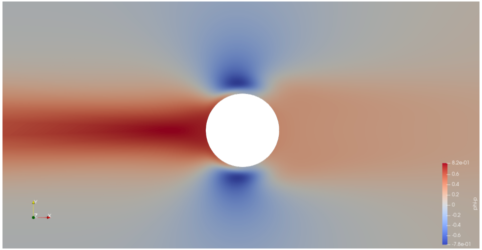


Figure: Adjoint Pressure

NACA0012 $AoA = 2.5°$, $Re = 1000$

Drag minimization. Boundary conditions:

- $\Gamma_{airfoil} : \Psi_u = [-1.0, 0.0]$ or $\Psi_u = [-C_{D,0}, 0.0]$
- $\Gamma_{inlet} : \Psi_p = 0.0$
- $\Gamma_{outlet} : \Psi_u = [0.0, 0.0]$
- $\Gamma_{limits} : \Psi_u = [0.0, 0.0]$

NACA0012 $AoA = 2.5^\circ, Re = 1000$

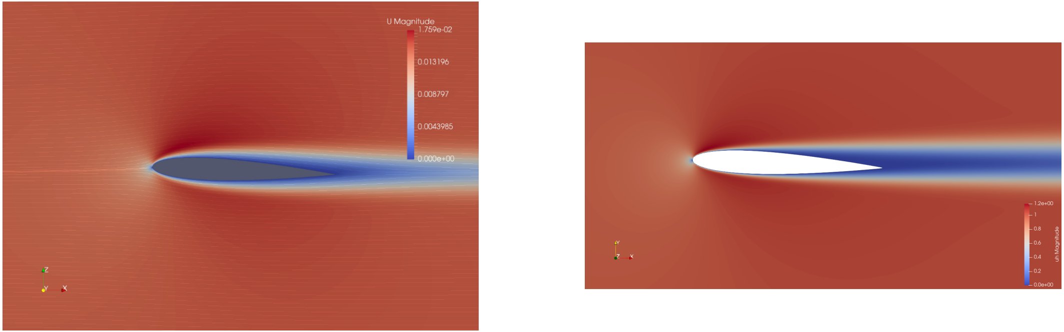
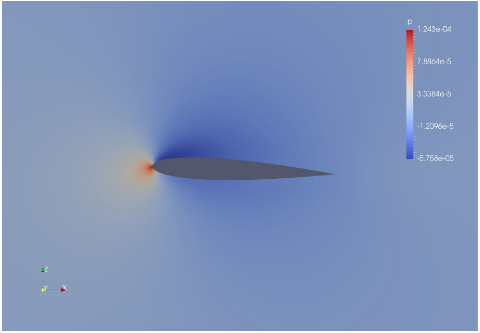


Figure: Primal Velocity



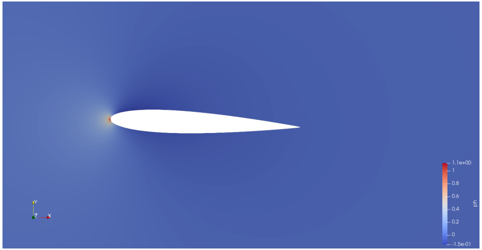
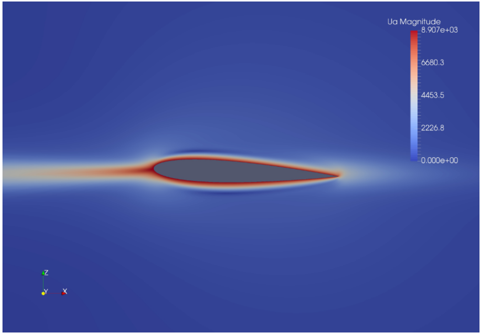


Figure: Primal Pressure



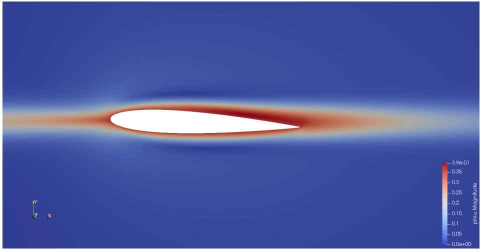


Figure: Adjoint Velocity

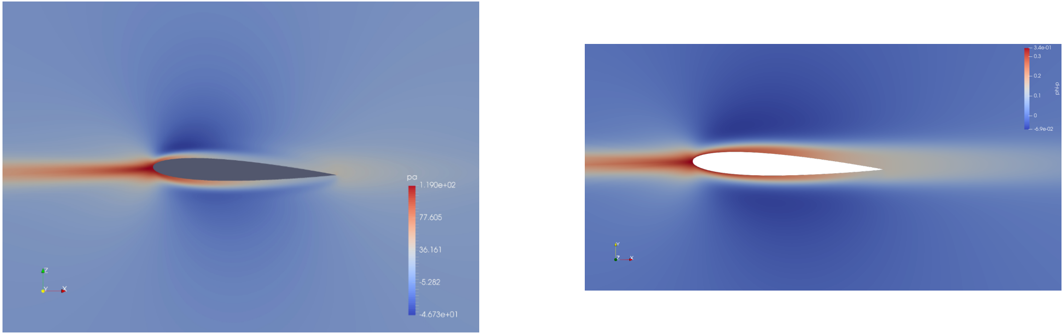


Figure: Adjoint Pressure

Finite Difference Tests

2 Test Cases

Computing the gradients for a STEADY case. NACA0012, $AoA = 0.0°$, Re=100. In FD, each β_i is perturbed by ϵ and the flow is solved.

$$\frac{\partial C_D}{\partial \beta_i} = \frac{C_D' - C_D}{\epsilon}$$

Gradients computed with FD and Adjoind should be the same, but they are not. The perturbation ϵ is the same for FD and adjoint.

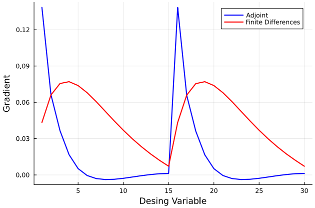


Figure: C_D gradient, finite difference vs adjoint

Finite Difference Tests

Gradients are symmetric in top and bottom design variables Order of magnitude really are close but not matching

Sections

3 Considerations

- Implementation
- Test Cases
- Considerations

- Gradients for FD and Adjoint do not depend on ϵ. $\epsilon = 0.001$, same results for $\epsilon = 0.01, 0.0001$.
- FD and Adjoint different results, but at least the sign is the same
- The adjoint results are sensible to the boundary condition on the airfoil surface. Fixing $\Psi_u = [-1.0, 0.0]$ or $\Psi_u = [-C_D, 0.0]$ impact the gradients.
- Error in boundary conditions? The error should be eliminated due to the difference $I' - I$.