

Windowing Regularization Techniques for Unsteady Aerodynamic Shape Optimization

Steffen Schotthöfer, Beckett Y. Zhou, Tim Albring, Nicolas R. Gauger

AG Scientific Computing
TU Kaiserslautern

10.06.2020

Copyright © by Steffen Schotthöfer, TU Kaiserslautern.
Published by the SU2 Foundation, with permission.

Unsteady periodic flows

Many fluid flows in industry are unsteady and turbulent.

- Aerodynamic design of rotor-craft
- Wind turbine design
- Flows around bluff bodies, e.g. airfoil at high angle of attack

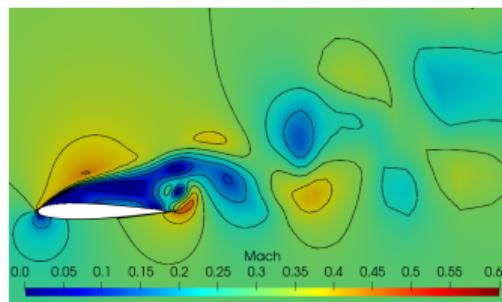
Unsteady periodic flows

Many fluid flows in industry are unsteady and turbulent.

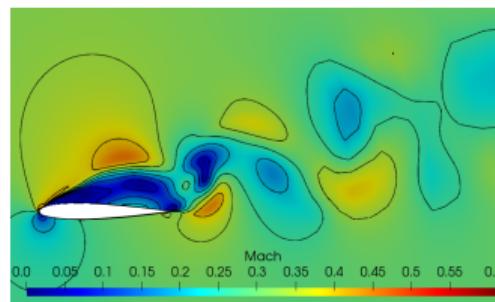
- Aerodynamic design of rotor-craft
- Wind turbine design
- Flows around bluff bodies, e.g. airfoil at high angle of attack

→ URANS flow simulations for separated flows result in periodic flow fields.

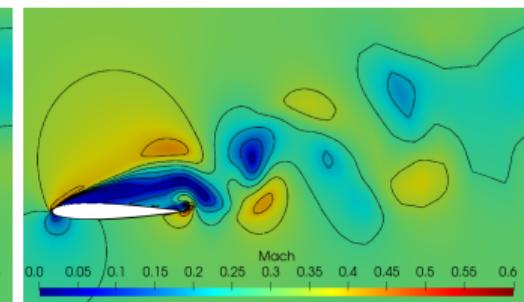
→ Flow field is smooth with clear periodicity.



(a) Snapshot at start of period



(b) at $\frac{1}{3}$ of period



(c) and at $\frac{2}{3}$ of period

Kármán vortex street behind airfoil ⇒ Undesirable!

Optimization objective in periodic flows

- Flow fields are often periodic after a transient time $t_{tr} = n_{tr} \Delta t$.
- Flow outputs such as Drag C_D or Lift C_L are periodic.
- Their sensitivities $\frac{d}{d\sigma} C_D$ and $\frac{d}{d\sigma} C_L$ w.r.t. design vector σ are periodic.
→ Optimization objective: Outputs averaged over a period.

$$J(\sigma; C_D) = \frac{1}{T(\sigma)} \int_{t_{tr}}^{T(\sigma)-t_{tr}} C_D(t, \sigma) dt \quad (1)$$

Optimization objective in periodic flows

- Flow fields are often periodic after a transient time $t_{tr} = n_{tr} \Delta t$.
- Flow outputs such as Drag C_D or Lift C_L are periodic.
- Their sensitivities $\frac{d}{d\sigma} C_D$ and $\frac{d}{d\sigma} C_L$ w.r.t. design vector σ are periodic.
→ Optimization objective: Outputs averaged over a period.

$$J(\sigma; C_D) = \frac{1}{T(\sigma)} \int_{t_{tr}}^{T(\sigma)-t_{tr}} C_D(t, \sigma) dt \quad (1)$$

- Period length $T(\sigma)$ unknown and dependent on σ !
- How to compute the sensitivity?

$$\frac{d}{d\sigma} J(\sigma; C_D) = \frac{d}{d\sigma} \frac{1}{T(\sigma)} \int_{t_{tr}}^{T(\sigma)-t_{tr}} C_D(t, \sigma) dt \quad (2)$$

Finite time averaging

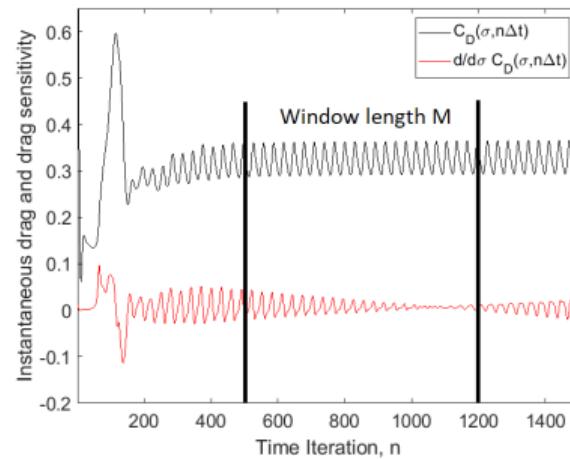
→ Average over finite time M and hope for convergence as $M \rightarrow \infty$,

$$J_M(\sigma, M; C_D) = \frac{1}{M} \int_{t_{tr}}^{M-t_{tr}} C_D(t, \sigma) dt. \quad (3)$$

■ Similar for time averaged sensitivity

$$\frac{d}{d\sigma} J_M(\sigma, M; C_D) = \frac{1}{M} \int_{t_{tr}}^{M-t_{tr}} \frac{d}{d\sigma} C_D(t, \sigma) dt. \quad (4)$$

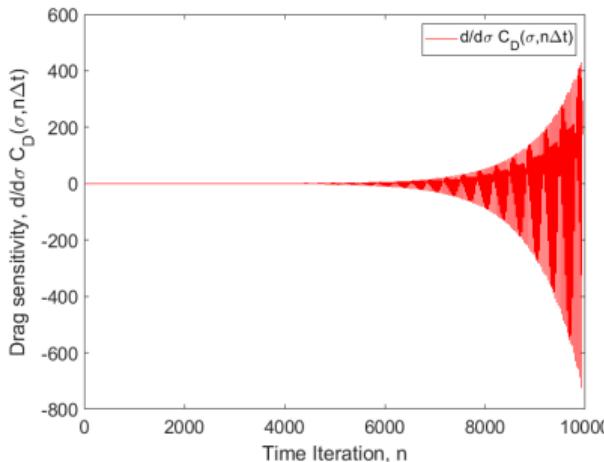
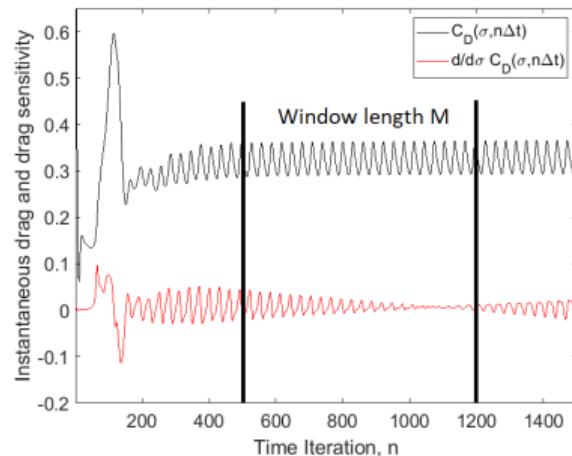
Convergence properties



- Amplitude change in $\frac{d}{d\sigma} C_D$
- Increase M for convergence of $J_M(\sigma, M; C_D)$ and $\frac{d}{d\sigma} J_M(\sigma)$?

¹Wilkins, A., Tidor, B., White, J. and Barton, P. , "Sensitivity Analysis for Oscillating Dynamical Systems", *SIAM Journal on Scientific Computing*, Vol. 31, No.4, 2009, pp. 2706-2732

Convergence properties



- Amplitude change in $\frac{d}{d\sigma} C_D$
- Increase M for convergence of $J_M(\sigma, M; C_D)$ and $\frac{d}{d\sigma} J_M(\sigma)$?

- Fast growth in amplitude from sensitivity of phase and period length¹.
- Last values of $\frac{d}{d\sigma} C_D$ distort sensitivity results

$$J_M(\sigma) \xrightarrow{M \rightarrow \infty} J(\sigma), \text{ but } \frac{d}{d\sigma} J_M(\sigma) \not\xrightarrow{M \rightarrow \infty} \frac{d}{d\sigma} J(\sigma)$$

¹Wilkins, A., Tidor, B., White, J. and Barton, P. , "Sensitivity Analysis for Oscillating Dynamical Systems", *SIAM Journal on Scientific Computing*, Vol. 31, No.4, 2009, pp. 2706-2732

Long time windowing by Krakos et al.²

→ Weighted average over finite time M and hope for convergence as $M \rightarrow \infty$,

$$J_w(\sigma, M; C_D) = \frac{1}{M} \int_{t_{tr}}^{M+t_{tr}} w\left(\frac{t - t_{tr}}{M}\right) C_D(t, \sigma) dt. \quad (5)$$

- Similar for time averaged sensitivity,

$$\frac{d}{d\sigma} J_w(\sigma, M; C_D) = \frac{1}{M} \int_{t_{tr}}^{M-t_{tr}} w\left(\frac{t - t_{tr}}{M}\right) \frac{d}{d\sigma} C_D(t, \sigma) dt. \quad (6)$$

²Krakos, J., Wang, Q., Hall, S. and Darmofal L. "Sensitivity analysis of limit cycle oscillations", *Journal of Computational Physics*, Vol. 231 (2012), 3228-3245

Long time windowing by Krakos et al.²

→ Weighted average over finite time M and hope for convergence as $M \rightarrow \infty$,

$$J_w(\sigma, M; C_D) = \frac{1}{M} \int_{t_{tr}}^{M+t_{tr}} w\left(\frac{t - t_{tr}}{M}\right) C_D(t, \sigma) dt. \quad (5)$$

- Similar for time averaged sensitivity,

$$\frac{d}{d\sigma} J_w(\sigma, M; C_D) = \frac{1}{M} \int_{t_{tr}}^{M-t_{tr}} w\left(\frac{t - t_{tr}}{M}\right) \frac{d}{d\sigma} C_D(t, \sigma) dt. \quad (6)$$

- $w(\tau) \in C^k$, $\tau = t/M$ is the **windowing function** with

$$\int_0^1 w(\tau) d\tau = 1 \quad \text{and} \quad w(\tau) = 0, \tau \in \mathbb{R} \setminus (0, 1). \quad (7)$$

→ $w = \mathbf{1}_{(0,1)} \in C^{k=-1}$ denotes a piecewise continuous window. (Finite time averaging is just a special case!)

²Krakos, J., Wang, Q., Hall, S. and Darmofal L. "Sensitivity analysis of limit cycle oscillations", *Journal of Computational Physics*, Vol. 231 (2012), 3228-3245

Convergence of long time windowing

- Convergence order depends on the smoothness of the windowing function $w \in C^k$.

²Krakos, J., Wang, Q., Hall, S. and Darmofal L. "Sensitivity analysis of limit cycle oscillations", *Journal of Computational Physics*, Vol. 231 (2012), 3228-3245

Convergence of long time windowing

- Convergence order depends on the smoothness of the windowing function $w \in C^k$.

Theorem (Krakos et al.²)

Let h be the period normalized flow output and $w \in C^k$. The windowed average error bound is given by

$$|J_w(\sigma, M) - J(\sigma)| \leq \|h\|_\infty \mathcal{O}(M^{-p}), \quad (8)$$

$$\left| \frac{d}{d\sigma} J_w(\sigma, M) - \frac{d}{d\sigma} J(\sigma) \right| \leq \|\partial_\sigma h\|_\infty \mathcal{O}(M^{-p}) + \left\| \frac{1}{T} \frac{dT}{d\sigma} \right\|_1 \|\partial_s h\|_\infty \mathcal{O}(M^{-(p-1)}) \quad (9)$$

where

$$p = \begin{cases} 1, & k = -1, \\ k+1, & k \geq 0, k \text{ even}, \\ k+2, & k > 0, k \text{ odd}. \end{cases} \quad (10)$$

→ No convergence for traditional finite time average given!

²Krakos, J., Wang, Q., Hall, S. and Darmofal L. "Sensitivity analysis of limit cycle oscillations", *Journal of Computational Physics*, Vol. 231 (2012), 3228-3245

Mathematical formulation of the optimization problem

$$\min_{\sigma} \quad J_w(u(\sigma), \sigma, M; C_D) \quad (11a)$$

$$s.t. \quad u^n = G(u^n, \sigma; u^{n-1}, u^{n-2}), \quad n = 1, \dots, N \quad (11b)$$

$$J_w(u(\sigma), \sigma, M; C_L) \geq c \in \mathbb{R} \quad (11c)$$

- Discretize windowed objective with midpoint rule,

$$J_w(C_D) \approx \frac{1}{N - n_{tr}} \sum_{n=n_{tr}}^{N-n_{tr}} w\left(\frac{n - n_{tr}}{N - n_{tr}}\right) C_D(u^n, \sigma). \quad (12)$$

- Flow constraint in fixed point formulation with 2nd order BDF method,

$$u^n = G(u^n, \sigma; u^{n-1}, u^{n-2}). \quad (13)$$

- Inequality constraints as windowed lift,

$$J_w(u(\sigma), \sigma, M; C_L) \geq c. \quad (14)$$

Lagrange approach

- Compute descent direction $\frac{d}{d\sigma} J_w$ for optimization.
- Avoid expensive gradients like $\frac{d}{d\sigma} u^n$, that appear using the chain rule.
→ Lagrange Function reads

$$L([u^1, \dots, u^n]^T, \sigma, [\bar{u}^1, \dots, \bar{u}^n]^T) = J_w(u, \sigma, M; C_D) + \sum_{n=1}^N (\bar{u}^n)^T (G(u^n, \sigma; u^{n-1}, u^{n-2}) - u^n). \quad (15)$$

Lagrange approach

- Compute descent direction $\frac{d}{d\sigma} J_w$ for optimization.
- Avoid expensive gradients like $\frac{d}{d\sigma} u^n$, that appear using the chain rule.
→ Lagrange Function reads

$$L([u^1, \dots, u^n]^T, \sigma, [\bar{u}^1, \dots, \bar{u}^n]^T) = J_w(u, \sigma, M; C_D) + \sum_{n=1}^N (\bar{u}^n)^T (G(u^n, \sigma; u^{n-1}, u^{n-2}) - u^n). \quad (15)$$

- Solve the KKT System

$$\partial_{\bar{u}^n} L = 0, \quad n = 1, \dots, N \quad \text{state equations} \quad (16a)$$

$$\partial_{u^n} L = 0, \quad n = 1, \dots, N \quad \text{adjoint equations} \quad (16b)$$

$$\partial_\sigma L = 0, \quad \text{design equation} \quad (16c)$$

Adjoint Sensitivity Computation

- Adjoint equations (adjoint iterator), given by $\partial_{\bar{u}^n} L = 0$,

$$\bar{u}_{p+1}^n = H(\bar{u}_p^n, \sigma; \bar{u}^{n+1}, \bar{u}^{n+2}), \quad n = 1, \dots, N. \quad (17)$$

Adjoint Sensitivity Computation

- Adjoint equations (adjoint iterator), given by $\partial_{u^n} L = 0$,

$$\bar{u}_{p+1}^n = H(\bar{u}_p^n, \sigma; \bar{u}^{n+1}, \bar{u}^{n+2}), \quad n = 1, \dots, N. \quad (17)$$

- In detail, we have

$$\begin{aligned} \partial_{u^n} L &= -\bar{u}^n + \overbrace{(\partial_{u^n} G^n)^T \bar{u}^n + (\partial_{u^n} G^{n+1})^T \bar{u}^{n+1} + (\partial_{u^n} G^{n+2})^T \bar{u}^{n+2}}^{\text{from 2nd order BDF method in direct iterator}} \\ &\quad + \underbrace{\mathbf{1}_{\{n \geq n_{tr}\}} \frac{1}{N - n_{tr}} \mathbf{w} \left(\frac{n - n_{tr}}{N - n_{tr}} \right) (\partial_{u^n} C_D(u^n, \sigma))^T}_{\text{independent of } \bar{u}^n}. \end{aligned} \quad (18)$$

Adjoint Sensitivity Computation

- Adjoint equations (adjoint iterator), given by $\partial_{\bar{u}^n} L = 0$,

$$\bar{u}_{p+1}^n = H(\bar{u}_p^n, \sigma; \bar{u}^{n+1}, \bar{u}^{n+2}), \quad n = 1, \dots, N. \quad (17)$$

- In detail, we have

$$\begin{aligned} \partial_{\bar{u}^n} L &= -\bar{u}^n + \overbrace{(\partial_{\bar{u}^n} G^n)^T \bar{u}^n + (\partial_{\bar{u}^n} G^{n+1})^T \bar{u}^{n+1} + (\partial_{\bar{u}^n} G^{n+2})^T \bar{u}^{n+2}}^{\text{from 2nd order BDF method in direct iterator}} \\ &\quad + \underbrace{\mathbf{1}_{\{n \geq n_{tr}\}} \frac{1}{N - n_{tr}} \mathbf{w} \left(\frac{n - n_{tr}}{N - n_{tr}} \right) (\partial_{\bar{u}^n} C_D(u^n, \sigma))^T}_{\text{independent of } \bar{u}^n}. \end{aligned} \quad (18)$$

- Contractivity of H^n ? → Given by contractivity of G^n ,

$$\|\partial_{\bar{u}^n} H^n\| = \|(\partial_{\bar{u}^n} G^n)^T\| = \|(\partial_{\bar{u}^n} G^n)\|. \quad (19)$$

→ Adjoint iterator H^n **inherits convergence** of direct iterator G^n regardless of the chosen window.

Adjoint Sensitivity Computation

- Design equation is given by $\partial_\sigma L$,

$$\partial_\sigma L = \sum_{n=0}^N \left[\mathbb{1}_{\{n \geq n_{tr}\}} \frac{1}{N - n_{tr}} w \left(\frac{n - n_{tr}}{N - n_{tr}} \right) \partial_\sigma C_D(u^n, \sigma) + (\bar{u}^n)^T \partial_\sigma G^n \right] \quad (20)$$

→ Windowing function $w \left(\frac{n - n_{tr}}{N - n_{tr}} \right)$ only appears as a *seeding* factor. → super cheap!

→ Only one floating point operation difference to traditional adjoint.

→ Easy to implement.

```
seeding = windowEvaluator.GetWndWeight(n,N,n_tr)/ (N-n_tr);
```

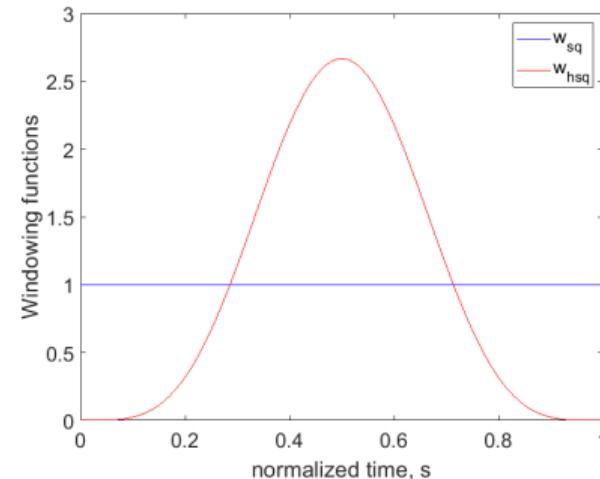
Test Case - NACA0012 Airfoil with high angle of attack³

- URANS Flow solver with JST flux.
- Spalart-Allmaras turbulence model with upwind flux.
- Reynolds number is 10^6 , Mach number is 0.3.

Used windowing functions

- Square: $w_{sq} \in C^{-1}$
 $\Rightarrow J_{w_{sq}}$ converges in $\mathcal{O}(M^{-1})$
 $\Rightarrow \partial_\sigma J_{w_{sq}}$ converges in $\mathcal{O}(M^0)$
- Hann-Square: $w_{hsq} \in C^3$
 $\Rightarrow J_{w_{hsq}}$ converges in $\mathcal{O}(M^{-5})$
 $\Rightarrow \partial_\sigma J_{w_{hsq}}$ converges in $\mathcal{O}(M^{-4})$

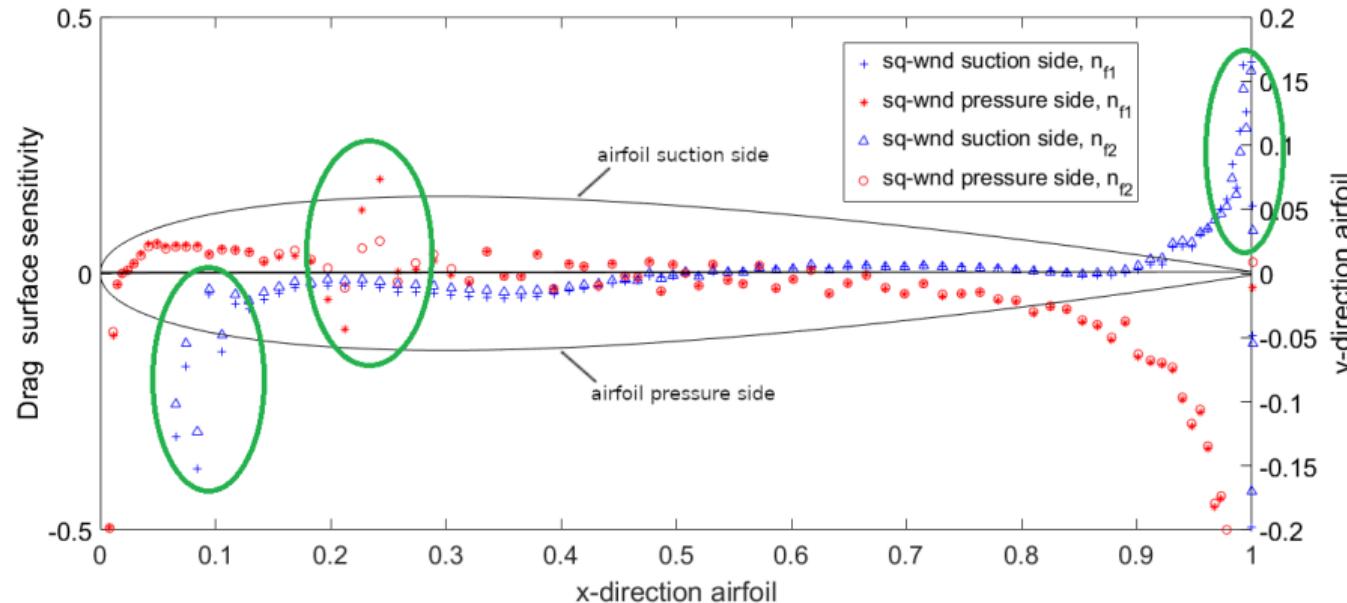
Period boundaries are weighted less than the middle of the period.



³Resources online at https://su2code.github.io/tutorials/Unsteady_Shape_Opt_NACA0012/

Surface Sensitivities - Square Windowing

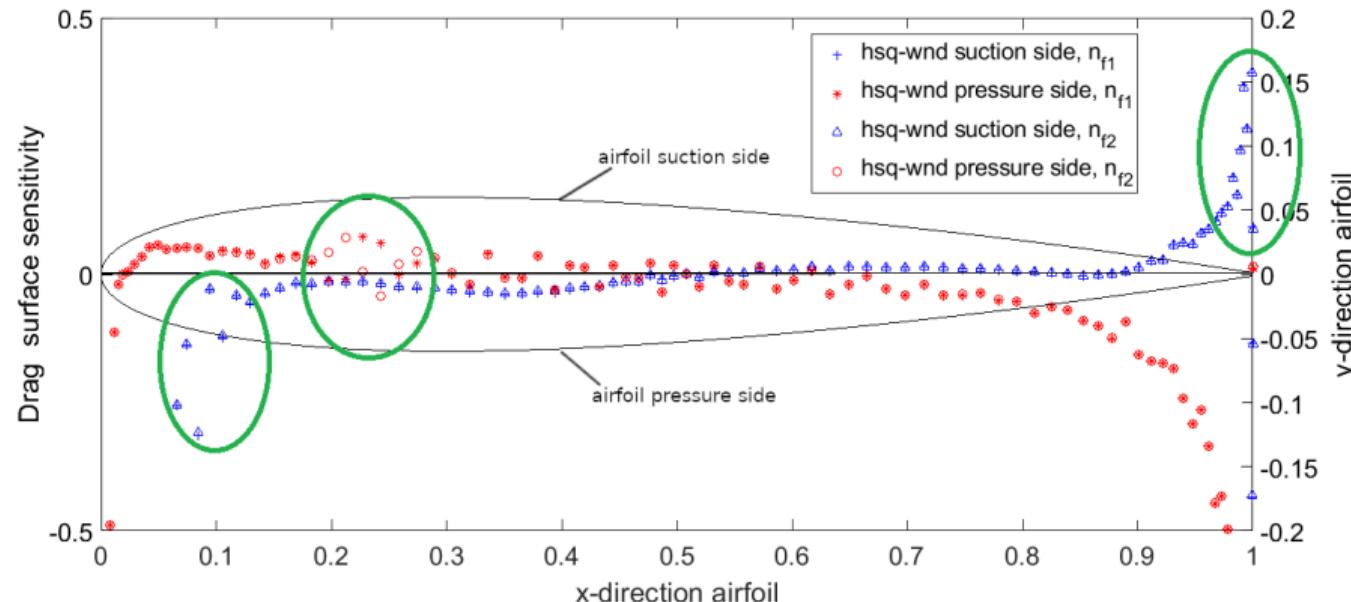
- Sensitivities of airfoil surface at final iterations n_{f1} and n_{f2} , where $n_{f2} - n_{f1} = 1/3 T$.



- Great difference of surface sensitivities for different final iterations!
- Square windowing is very sensitive w.r.t. choice of final iteration n_f .

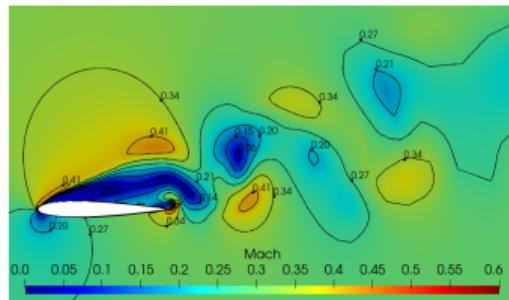
Surface Sensitivities - Hann-Square Windowing

- Sensitivities of airfoil surface at final iterations n_{f1} and n_{f2} , where $n_{f2} - n_{f1} = 1/3 T$.

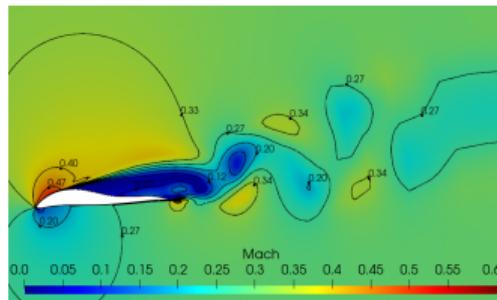


- Small difference** of surface sensitivities for different final iterations!
- Hann-Square windowing is **robust** w.r.t. choice of final iteration n_f . → Good for shape optimization!

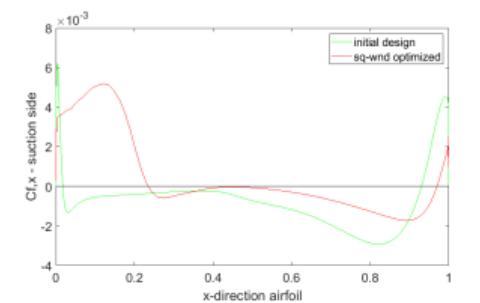
Optimization Result - Square Windowing



(a) Initial design



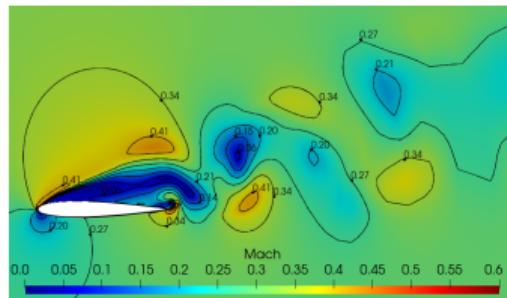
(b) After 15 design iterations



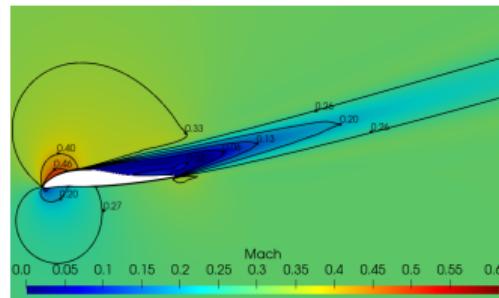
(c) Skin friction coefficient, \times dir.

- Small reduction of the vortex street, still periodic behavior.
 - C_f, x has sign change at 23% airfoil length.
 - The flow detaches itself further back on the wing than in the initial design.

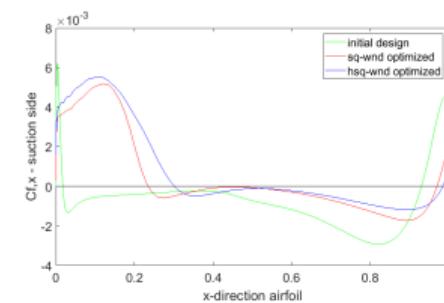
Optimization Result - Hann-Square Windowing



(a) Initial design



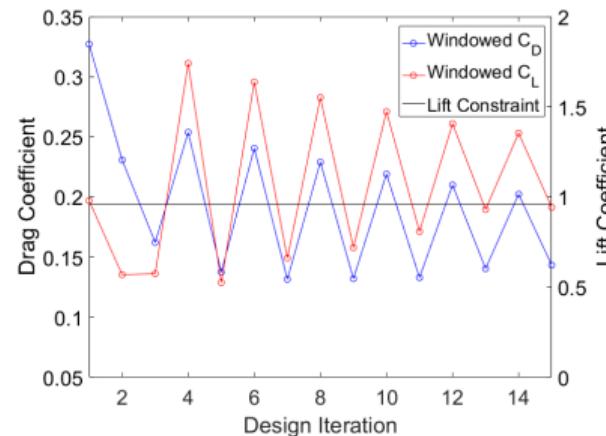
(b) After 15 design iterations



(c) Skin friction coefficient, x dir.

- Optimized flow is in steady state, no vortices left.
- $C_f \cdot x$ has sign change at 30% airfoil length.
- The flow detaches itself 50% further back on the wing than in the Square windowed design.

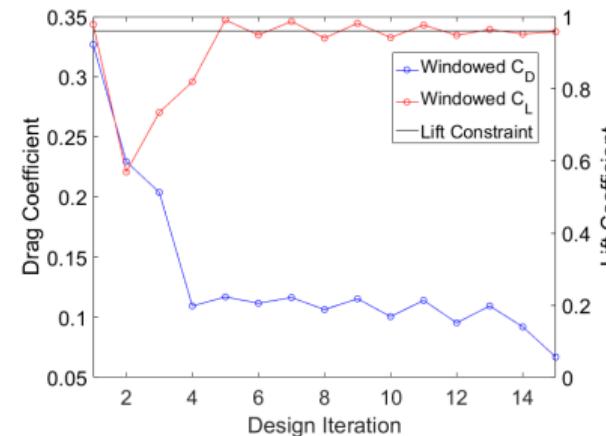
Comparison of Design Processes



Square windowing

- ca. 59% drag reduction.
- Many designs infeasible (line search fails).
- Low Drag \Rightarrow low Lift.

\Rightarrow Similar results for other high order windows.



Hann-Square windowing

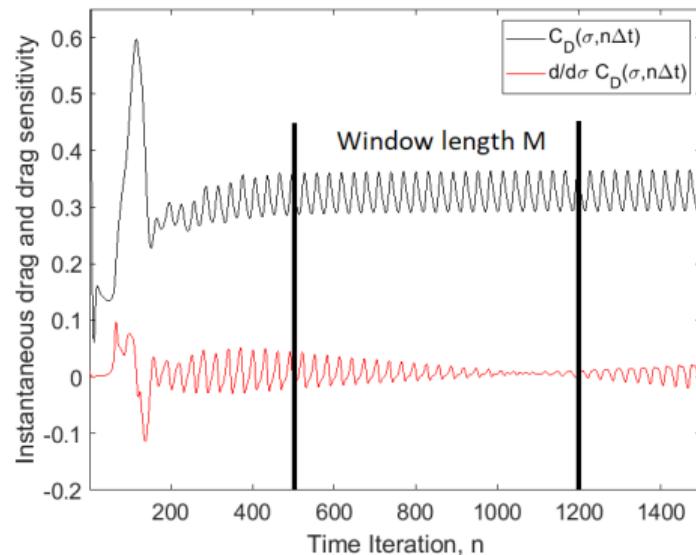
- ca. 81% drag reduction.
- Almost all designs feasible.
- Low Drag, but high Lift.

Windowing in SU2 - Usage and Implementation

- Two possibilities in SU2.
 - Adjoint sensitivity computation via Lagrange approach and automatic differentiation.
 - Windowed time averages of flow outputs or
Windowed time averages of sensitivities via direct (tangent) sensitivity computation via AD (forward mode).
- Managed by Class CWindowingTools → Easy to add new windowing functions!

Windowing in SU2 - Adjoint mode and shape optimization³

- Define your windowing function `WINDOW_FUNCTION = HANN`
- Specify the final time iteration for the direct run
`TIME_ITER = 1200`
- Specify the iteration to start the time average (after transient phase is over) `WINDOW_START_ITER = 500`
- Specify the iteration to start adjoint run
`UNST_ADJOINT_ITER = 1200`
- Specify the window length M `ITER_AVERAGE_OBJ = 700`
- Choose the optimization objective `OPT_OBJECTIVE = DRAG * 1.0`



³Resources online at https://su2code.github.io/tutorials/Unsteady_Shape_Opt_NACA0012/

Windowing in SU2 - Direct mode⁴

→ Postprocessing step of the direct solver at each time iteration.

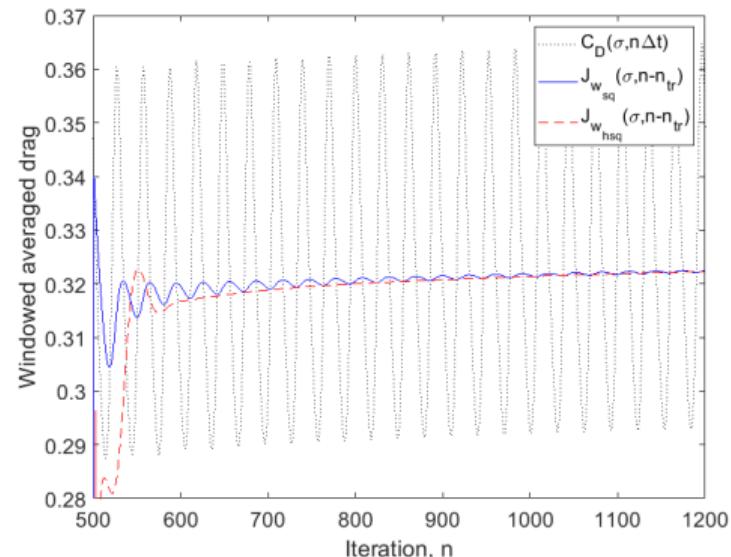
Compute $u^n = G(u^n, \dots)$, then the *running* windowed average $J_w(u, n - n_{tr}, C_D)$.

- Define your windowing function `WINDOW_FUNCTION = HANN_SQUARE`
- Define the iteration to start the time average (after transient phase is over) `WINDOW_START_ITER = 500`
- Choose output groups/fields to average by using the prefix `TAVG_` (time average) and `D_TAVG_` (time averaged sensitivity)
`SCREEN_OUTPUT = (TAVG_DRAG,D_TAVG_DRAG,...)`
`HISTORY_OUTPUT = (TAVG_AEROCOEFF,D_TAVG_AEROCOEFF,...)`
- Value is the windowed average from `WINDOW_START_ITER` up to the current iteration.
→ Can be used for time convergence monitoring!

⁴Resources online at https://su2code.github.io/tutorials/Unsteady_NACA0012/

Windowing in SU2 - Direct mode convergence monitoring⁴

- Idea: Output values should be a Cauchy sequence.
- Activate the window convergence criterion.
`WINDOW_CAUCHY_CRIT = YES`
- Specify convergence field(s).
`CONV_WINDOW_FIELD = (TAVG_DRAG, D_TAVG_LIFT)`
- Specify the number of elements to apply the criterion.
`CONV_WINDOW_CAUCHY_ELEMS = 100`
- Set Epsilon to control the series convergence.
`CONV_WINDOW_CAUCHY_EPS = 1E-2`



Convergence of windowed Averages

⁴Resources online at https://su2code.github.io/tutorials/Unsteady_NACA0012/

Windowing in SU2 - Efficient shape optimization workflows

Simulation

- Get end of transient phase n_{tr} (plus some buffer iterations).
- Save restart files, to avoid computing the transient phase in each design iteration.

Direct Differentiation

- Use windowed time convergence monitoring for sensitivity of objective function.
- Get minimal number N_{min} of time iterations needed for a converged windowed sensitivity.

Shape optimization

- Use restart files to skip transient phase in each design iteration.
- Use minimal time frame (plus some buffer iterations) for discrete adjoint computation, i.e.
 $M = (N_{min} + \text{buffer}) - (n_{tr} + \text{buffer})$ and save computational time.

Summary and Conclusion

- Traditional time averages fail to compute sensitivities correctly in periodic flows.
- Long time windowing yields **faster convergence** of period averages and **correct sensitivities**.
- Windowing is **easy to embed** in adjoint sensitivity solvers.
- Windowing gives **more robust** surface sensitivities with respect to changing final times.
- Windowed shape optimization processes converge faster and are able to fulfill design constraints.
 - The traditional approach fails to do this.
- Future work: Extension to DES flows. DES flows resolve multiple vortex scales.
 - Decompose flow field into base frequencies and apply windowing.

Thank you for your attention!