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

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 1=3 of period (c) and at 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 $\frac{\mathrm{d}}{\mathrm{d}}C_D$ and $\frac{\mathrm{d}}{\mathrm{d}}C_L$ w.r.t. design vector are periodic.

! Optimization objective: Outputs averaged over a period.

$$J(\ ; C_D) = \frac{1}{T(\ )} \int_{t_{tr}}^{T(\ ) \ t_{tr}} C_D(t; \ ) \, \mathrm{d}t \tag{1}$$

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 $\frac{\mathrm{d}}{\mathrm{d}}C_D$ and $\frac{\mathrm{d}}{\mathrm{d}}C_L$ w.r.t. design vector are periodic.

! Optimization objective: Outputs averaged over a period.

$$J(\ ; C_D) = \frac{1}{T(\ )} \int_{t_{tr}}^{T(\ ) \ t_{tr}} C_D(t; \ ) \, \mathrm{d}t \tag{1}$$

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

$$\frac{\mathrm{d}}{\mathrm{d}} J(\ ; C_D) = \frac{\mathrm{d}}{\mathrm{d}} \frac{1}{T(\ )} \int_{t_{tr}}^{T(\ ) \ t_{tr}} C_D(t; \ ) \, \mathrm{d}t \tag{2}$$

Finite time averaging

- Average over finite time M and hope for convergence as $M \to \infty$,

$$J_M(\ ; M; C_D) = \frac{1}{M} \int_{t_{tr}}^{M\ t_{tr}} C_D(t;\ )\ \mathrm{d}t: \tag{3}$$

- Similar for time averaged sensitivity

$$\frac{\mathrm{d}}{\mathrm{d}}J_M(\ ; M; C_D) = \frac{1}{M} \int_{t_{tr}}^{M\ t_{tr}} \frac{\mathrm{d}}{\mathrm{d}}C_D(t;\ )\ \mathrm{d}t: \tag{4}$$

Convergence properties

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

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

Convergence properties

- Amplitude change in $\frac{\mathrm{d}}{\mathrm{d}} C_D$
- Increase M for convergence of $J_M(\ ; M; C_D)$ and $\frac{\mathrm{d}}{\mathrm{d}} J_M(\ )$?
- Fast growth in amplitude from sensitivity of phase and period length[1].
- Last values of $\frac{\mathrm{d}}{\mathrm{d}} C_D$ distort sensitivity results

$$J_M(\ )^{M\ !\ !^1} J(\ ), \text{ but } \frac{\mathrm{d}}{\mathrm{d}} J_M(\ )^{M\ !\ 6\ !\ !^1} \frac{\mathrm{d}}{\mathrm{d}} J(\ )$$

[1] Wilkins, A., Tidor, B., White, J. and Barton, P. , "Sensitivity Analysis for Oscillating Dynamical Systems", *SIAM Journal on Scienti c 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 $M \; ! \; 1$,

$$J_w(\;;M;C_D) = \frac{1}{M}\int_{t_{tr}}^{M+t_{tr}} w\left(\frac{t \quad t_{tr}}{M}\right) C_D(t;\;)\,\mathrm{d}t: \tag{5}$$

- Similar for time averaged sensitivity,

$$\frac{\mathrm{d}}{\mathrm{d}}J_w(\;;M;C_D) = \frac{1}{M}\int_{t_{tr}}^{M\quad t_{tr}} w\left(\frac{t \quad t_{tr}}{M}\right) \frac{\mathrm{d}}{\mathrm{d}}C_D(t;\;)\,\mathrm{d}t: \tag{6}$$

[2] Krakos, 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 $M \to \infty$,

$$J_w(\ ;M;C_D) = \frac{1}{M}\int_{t_{tr}}^{M+t_{tr}} w\left(\frac{t-t_{tr}}{M}\right) C_D(t;\ )\,\mathrm{d}t: \tag{5}$$

- Similar for time averaged sensitivity,

$$\frac{\mathrm{d}}{\mathrm{d}}J_w(\ ;M;C_D) = \frac{1}{M}\int_{t_{tr}}^{M\ t_{tr}} w\left(\frac{t-t_{tr}}{M}\right) \frac{\mathrm{d}}{\mathrm{d}}C_D(t;\ )\,\mathrm{d}t: \tag{6}$$

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

$$\int_0^1 w(\ )\,\mathrm{d}\ = 1 \quad \text{and} \quad w(\ ) = 0; \quad \in \mathbb{R}\, n\,(0;1): \tag{7}$$

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

[2] Krakos, 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 $w \ 2 \ C^k$.

[2] Krakos, 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

$$|J_w(\cdot;M) - J(\cdot)| \leq \|h\|_1 \, O(M^{-p}); \tag{8}$$

$$\left|\frac{\mathrm{d}}{\mathrm{d}}J_w(\cdot;M) - \frac{\mathrm{d}}{\mathrm{d}}J(\cdot)\right| \leq \|w\|_\infty \|h\|_1 \, O(M^{-p}) + \frac{1}{T}\frac{\mathrm{d}T}{\mathrm{d}}\|w\|_1 \|\dot{h}\|_1 \, O(M^{-(p-1)}) \tag{9}$$

where

$$p = \begin{cases} 1; & k = 1; \\ k+1; & k \neq 0; k \text{ even}; \\ k+2; & k > 0; k \text{ odd}. \end{cases} \tag{10}$$

! No convergence for traditional finite time average given!

[2] Krakos, 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

$$\min \quad J_w(u(\ );\ ;M;C_D) \tag{11a}$$

$$s.t. \quad u^n = G\ u^n;\ ;u^{n\ 1};u^{n\ 2}\ ; \qquad n = 1;:::;N \tag{11b}$$

$$J_w(u(\ );\ ;M;C_L)\quad c\ 2\ \mathbb{R} \tag{11c}$$

- Discretize windowed objective with midpoint rule,

$$J_w(C_D)\quad \frac{1}{N\quad n_{tr}} \sum_{n=n_{tr}}^{N} w\ \frac{n\quad n_{tr}}{N\quad n_{tr}}\ C_D(u^n;\ ): \tag{12}$$

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

$$u^n = G(u^n;\ ;u^{n\ 1};u^{n\ 2}): \tag{13}$$

- Inequality constraints as windowed lift,

$$J_w(u(\ );\ ;M;C_L)\quad c: \tag{14}$$

Lagrange approach

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

! Lagrange Function reads

$$L([u^1;\dots;u^n]^T;\ ;[\overline{u}^1;\dots;\overline{u}^n]^T) = J_w(u;\ ;M;C_D) + \sum_{n=1}^{N} (\overline{u}^n)^T\ \ G\ \ u^n;\ \ ;u^n\ \ ^1;u^n\ \ ^2\ \ \ \ u^n\ \ : \qquad (15)$$

Lagrange approach

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

! Lagrange Function reads

$$L([u^1;\dots;u^n]^T;\ ;[\overline{u}^1;\dots;\overline{u}^n]^T) = J_w(u;\ ;M;C_D) + \sum_{n=1}^{N} (\overline{u}^n)^T\ \ G\ \ u^n;\ \ ;u^n\ \ ^1;u^n\ \ ^2\ \ \ \ u^n\ \ : \tag{15}$$

- Solve the KKT System

$$@_{\overline{u}^n} L = 0; \qquad n = 1;\dots;N \quad \text{state equations} \tag{16a}$$
$$@_{u^n} L = 0; \qquad n = 1;\dots;N \quad \text{adjoint equations} \tag{16b}$$
$$@\ \ L = 0; \qquad \text{design equation} \tag{16c}$$

Adjoint Sensitivity Computation

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

$$\overline{u}^n_{p+1} = H \quad \overline{u}^n_p; \quad ; \overline{u}^{n+1}; \overline{u}^{n+2} \quad ; \qquad n = 1; : : : ; N: \tag{17}$$

Adjoint Sensitivity Computation

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

$$\overline{u}^n_{p+1} = H\left(\overline{u}^n_p;\ ;\overline{u}^{n+1};\overline{u}^{n+2}\ ;\right), \qquad n = 1;\dots;N; \tag{17}$$

- In detail, we have

$$\begin{aligned} \partial_{u^n} L = {} & \overline{u}^n + \left(\partial_{u^n} G^n\right)^T \overline{u}^n + \underbrace{\left(\partial_{u^n} G^{n+1}\right)^T \overline{u}^{n+1} + \left(\partial_{u^n} G^{n+2}\right)^T \overline{u}^{n+2}}_{\text{from } 2^{\text{nd}} \text{ order BDF method in direct iterator}} \\ & + 1_{f n\ \ n_{tr} g} \frac{1}{N\ \ n_{tr}} w\left(\frac{n\ \ n_{tr}}{N\ \ n_{tr}}\right)\left(\partial_{u^n} C_D(u^n;\ \ )\right)^T : \\ & \underbrace{\qquad\qquad}_{\text{independent of } \overline{u}^n} \end{aligned} \tag{18}$$

Adjoint Sensitivity Computation

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

$$\overline{u}^n_{p+1} = H\ \ \overline{u}^n_p;\ \ ;\overline{u}^{n+1};\overline{u}^{n+2}\ \ ; \qquad n = 1;:::;N: \tag{17}$$

- In detail, we have

$$\begin{aligned} @_{u^n} L = \quad & \overline{u}^n + (@_{u^n} G^n)^T\, \overline{u}^n + \underbrace{@_{u^n} G^{n+1}}_{\text{from } 2^{\text{nd}} \text{ order BDF method in direct iterator}}{}^T\, \overline{u}^{n+1} + \underbrace{@_{u^n} G^{n+2}}_{\text{from } 2^{\text{nd}} \text{ order BDF method in direct iterator}}{}^T\, \overline{u}^{n+2} \\ & + 1_{fn\ \ n_{tr} g} \frac{1}{N\quad n_{tr}} w \quad \frac{n\quad n_{tr}}{N\quad n_{tr}} \quad (@_{u^n} C_D(u^n;\ \ ))^T : \\ & \underbrace{\qquad\qquad}_{\text{independent of } \overline{u}^n} \end{aligned} \tag{18}$$

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

$$k@_{\overline{u}^n} H^n k = \ \ (@_{u^n} G^n)^T \ \ = k(@_{u^n} G^n)k: \tag{19}$$

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

Adjoint Sensitivity Computation

- Design equation is given by $@\ L$,

$$@\ L = \sum_{n=0}^{N} 1_{fn\ \ n_{tr}g} \frac{1}{N\quad n_{tr}} w\quad \frac{n\quad n_{tr}}{N\quad n_{tr}}\ @\ C_D(u^n;\ \ ) + (\overline{u}^n)^T\ @\ G^n \tag{20}$$

! Windowing function $w\quad \frac{n\quad n_{tr}}{N\quad n_{tr}}$ only appears as a *seeding* factor. ! super cheap!

! Only one floating point operation difference to traditional adjoint.

! Easy to implement.

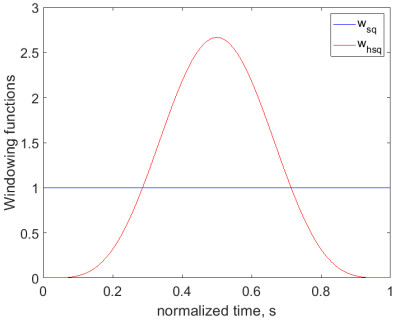
```
seeding = windowEvaluator.GetWndWeight(n,N,n_tr)/ (N-n_tr);
```

Test Case - NACA0012 Airfoil with high angle of attack[3]

- 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

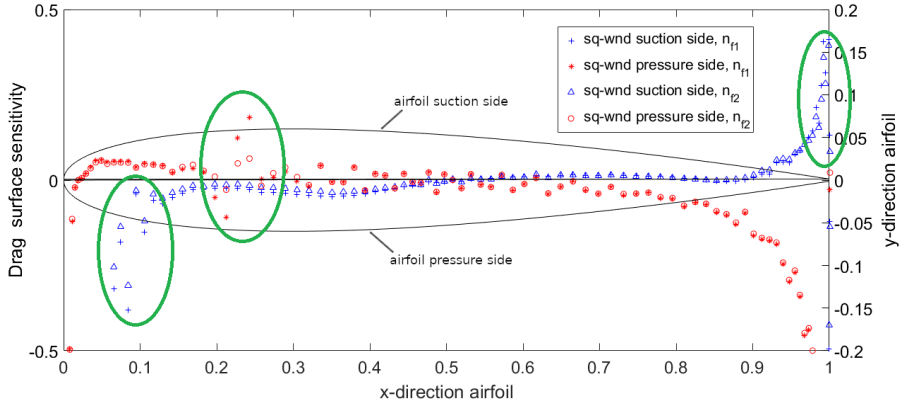
- Square: w_{sq} 2 C [1]
) $J_{w_{sq}}$ converges in $O(M$ [1]$)$
) @ $J_{w_{sq}}$ converges in $O(M^0)$
- Hann-Square: w_{hsq} 2 C^3
) $J_{w_{hsq}}$ converges in $O(M$ [5]$)$
) @ $J_{w_{hsq}}$ converges in $O(M$ [4]$)$
  Period boundaries are weighted less than the middle of the period.



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

Surface Sensitivities - Square Windowing

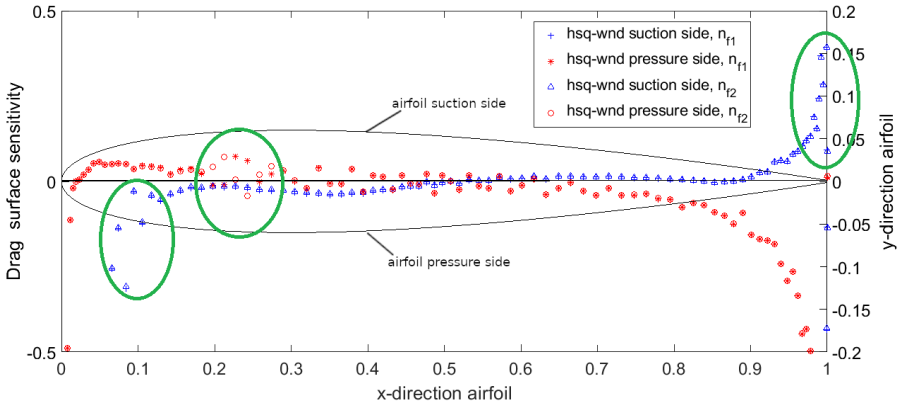
- Sensitivities of airfoil surface at final iterations n_{f1} and n_{f2}, where n_{f2} $n_{f1} = 1{=}3T$.



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

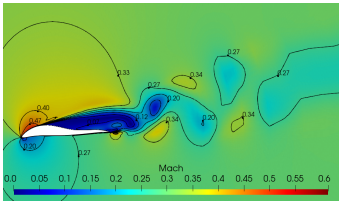
Surface Sensitivities - Hann-Square Windowing

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

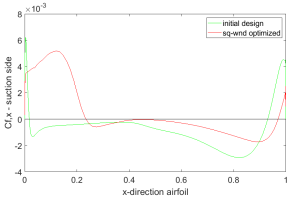


- Small difference of surface sensitivities for different final iterations!
- Hann-Square windowing is robust w.r.t. choice of final iteration n_f. ! Good for shape optimization!

Optimization Result - Square Windowing



(a) Initial design

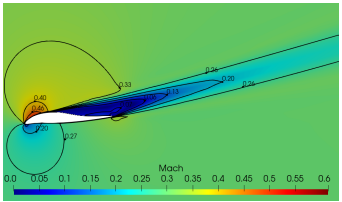


(b) After 15 design iterations

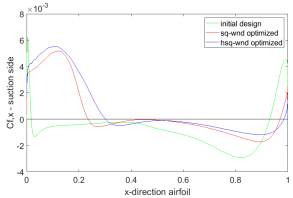
(c) Skin friction coefficient, x dir.

- Small reduction of the vortex street, still periodic behavior.
- $Cf\,;x$ 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

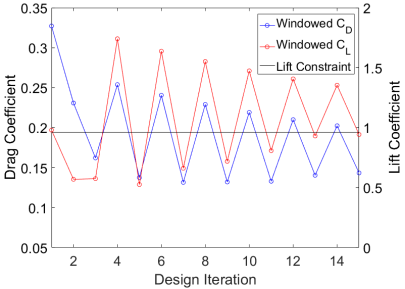


(b) After 15 design iterations

(c) Skin friction coefficient, x dir.

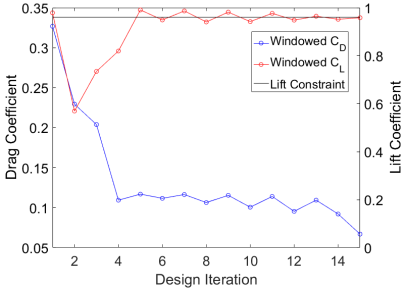
- Optimized flow is in steady state, no vortices left.
- $Cf\,;x$ has sign change at 30% airfoil length.
- The flow detaches itself 50% further back on the wing than in the Square windowed design.

Comparison of Design Processes



Square windowing

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



Hann-Square windowing

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

) Similar results for other high order windows.

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 optimization[3]

- 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`

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

Windowing in SU2 - Direct mode[4]

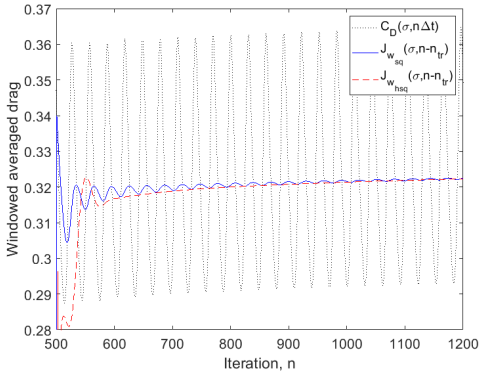
! Postprocessing step of the direct solver at each time iteration.
Compute $u^n = G(u^n; :::)$, then the *running* windowed average $J_w(u; n \quad n_{tr}; C_D)$.

- 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!

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

Windowing in SU2 - Direct mode convergence monitoring[4]

- 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

[4] Resources 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. $M = (N_{min} + buffer) - (n_{tr} + buffer)$ 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!