1 Fluid SA! (SA!)

As explained in section **??**, optimization is based on the calculations of gradients, also denoted as **SA!** (**SA!**).

1.1 Connection between discretization and differentiation

When it comes to calculating the gradients of a solution of a PDE, there are two fundamentally different approaches one could take.
Firstly one could think of first deriving the continous equation and then applying the discretization scheme. On the other hand its equaly legitimate to first apply the discretization and compute the gradients of that, approximated solution. This thesis focuses uses the latter one, the reasons being explained in Figure 1.

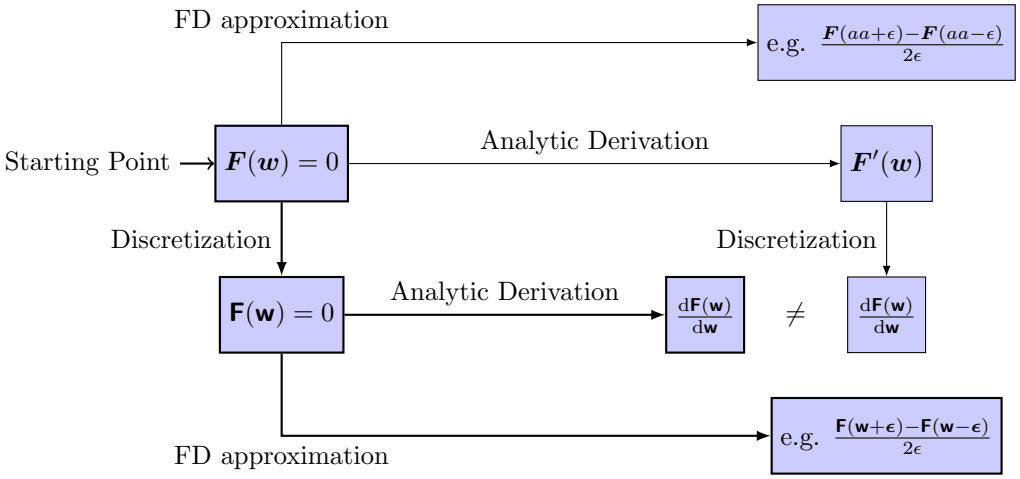

*Figure 1: Two different approaches of **SA!**. The question here is whether to discretize the system of equations first and then compute the derivatives of the approximate solution, or whether the continuous system of equation is first derived and discretized afterwards. Both approaches are valid, the final result however will generally not be the same. We have solemnly focused on the case of derivation after discretization in this thesis(thick lines). There are several reason to this. Firstly, if one chose to first derive and then compute a discretized solution, one would solve a completely different equation. If on the other hand the discretization(e.g. Finite Volumes) is done first, standard solvers can be utilized. What is more, approximating the derivative by a Finite difference becomes much easier, since the finite difference evaluation involves only a evaluation of* $\mathbf{F}$*, which a standard fluid solver is perfectly capable of, whereas the **FD!** approximation in the other case would involve evaluations of the continuous function and the discretization of the obtained expression which would be much more cumbersome. **Think about whether this is correct***

1.2 Sensitivity derivation

When it comes to **SA!** within the context of **CFD!** (**CFD!**), one also has to distinguish between **SA!** of the non-coupled fluid problem and the **SA!** in the fully

coupled aeroelastic case. The latter case is much more involved, since it requires additional terms from the Structure and mesh motion equations. In the following we will therefore begin with the **SA!** of a regular fluid problem. A generalization to a coupled FSI problem will follow.

As mentioned in Section **??**, we typically deal with an optimization criteria q_j, that is in this case dependent on the fluid state variables $\boldsymbol{w}$ that themselves may depend on abstract variables s_i

$$q_j = q_j(\boldsymbol{w}(s_i)) \tag{1}$$

$$\left.\frac{dq_j}{ds_i}\right|_{\boldsymbol{w}_0} = \underbrace{\left.\frac{\partial q_j}{\partial s_i}\right|_{\boldsymbol{w}_0}}_{\text{directy derived from the definition of } q} + \underbrace{\left.\frac{\partial q_j}{\partial \boldsymbol{w}}\right|_{\boldsymbol{w}_0}}_{\text{derived analytically or by FD!}} \underbrace{\left.\frac{d\boldsymbol{w}}{ds_i}\right|_{\boldsymbol{w}_0}}_{\text{derived from dynamic fluid equilibrium}} \tag{2}$$

eq:sensitivity_startingpoint

1.2.1 Calculation of $\left.\frac{\partial q_j}{\partial s_i}\right|_{\boldsymbol{w}_0}$

In this thesis, we restrict ourselves to Lift(L), Drag(D) and combinations of these quantities(e.g. Lift-Drag ration $\frac{L}{D}$) as optimization criteria q. As abstract variables we only consider the free stream mach-number M_∞, the free-stream angle of attack α_∞ and an airfoil shape parameter as explained in section **??**. The formula for the lift over an airfoil can be written as
If Γ_{FS} denotes the fluid structure interface, which in the **ALE!** (**ALE!**) context coincides with the airfoil surface, one can formulate the lift and drag of an airfoil in a steady state as:

$$L = \int_{\Gamma_{FS}} p(\boldsymbol{w}, \boldsymbol{x}) \boldsymbol{n}(\boldsymbol{x}) \cdot \boldsymbol{e}_L dS$$
$$D = \int_{\Gamma_{FS}} p(\boldsymbol{w}, \boldsymbol{x}) \boldsymbol{n}(\boldsymbol{x}) \cdot \boldsymbol{e}_D dS \tag{3}$$

eq:lift_and_drag_integrals

$$\tag{4}$$

where $\boldsymbol{e}_D$ is the unit vector pointing in the direction of the free stream, and $\boldsymbol{e}_L$ is the unit vector perpendicular to that. We therefor note, that $\frac{\partial L}{\partial s_i}$ and $\frac{\partial D}{\partial s_i}$ are zero for $s_i = M_\infty$ and $s_i s = \alpha_\infty$ and non-zero if s_i is a shape parameter.
Also, having determined the derivatives $\frac{\partial L}{\partial s_i}$ and $\frac{\partial D}{\partial s_i}$, the derivative of the Lift-Drag ratio follows simply as

$$-\frac{\partial(\frac{L}{D})}{\partial s_i}|_{\boldsymbol{w}_0} = -\frac{D_0 \frac{\partial L}{\partial s_i}|_{\boldsymbol{w}_0} - L_0 \frac{\partial D}{\partial s_i}|_{\boldsymbol{w}_0}}{{D_0}^2} \tag{5}$$

eq:lifttoddrag_by_absvar

1.2.2 Calculation of $\left.\frac{\partial q_j}{\partial \boldsymbol{w}}\right|_{\boldsymbol{w}_0}$

The derivative of the optimization criteria with respect to the fluid state vector can again be splitted into Lift and Drag part. For an aeroelastic simulations, the biggest difficulty here is the dependence on the fluid state by the fluid-srtucture interface itself. We are therefore faced with a derivative of an integral quantity, where the integration area itself is dependent on the quantiti of interest.
For this purpose, we recall the leibnitz rule from calculus:

$$\frac{d}{dx}\Big(\int_{a(x)}^{b(x)} f(x,t)dt\Big) = f(x,b(x))\cdot\frac{d}{dx}b(x) - f(x,a(x))\cdot\frac{d}{dx}a(x) + \int_{a(x)}^{b(x)}\frac{d}{dx}f(x,t)dt \tag{6}$$ eq:leibnitz_rule

Alternatively, we can directly switch to the discrete form, where the calculation of Lift and drag as outlined in Equation (3) can be performed as

$$L = \sum_{e\in\sum_{\Gamma_{FS}}}\sum_{i=1}^{n_g} g_i \mathsf{p}(\mathbf{w}_i,\mathbf{x}_i)\boldsymbol{n}_e\cdot e_L D = \sum_{e\in\sum_{\Gamma_{FS}}}\sum_{i=1}^{n_g} g_i \mathsf{p}(\mathbf{w}_i,\mathbf{x}_i)\boldsymbol{n}_e\cdot e_D \tag{7}$$ eq:discrete_lift_and_drag_formulas

where $\sum_{\Gamma_{FS}}$ is the set of surface elements on the fluid-structure interface, n_g is the number of gauss-points and g_i are the gauss weights. The dependency of the surface shape, on the fluid state vector is now reflected in non-zero derivatives of the gauss point locations.
Finally, the derivative of the lift-to-drag ratio can be obtained analogously to equation (5) as

$$\frac{\partial(\frac{\mathsf{L}}{\mathsf{D}})}{\partial\mathbf{w}}|_{\boldsymbol{w}_0} = \frac{\mathsf{D}_0\frac{\partial\mathsf{L}}{\partial\mathbf{w}}|_{\boldsymbol{w}_0} - \mathsf{L}_0\frac{\partial D}{\partial\mathbf{w}}|_{\mathbf{w}_0}}{{\mathsf{D}_0}^2} \tag{8}$$ eq:lifttodrag_by_dfstate

1.2.3 Calculation of $\left.\frac{d\boldsymbol{w}}{ds_i}\right|_{\boldsymbol{w}_0}$

Also, we keep in mind, that the state equation of the fluid can be expressed as

$$\mathcal{F}_{gov}(\boldsymbol{w}(s_i),\dot{\boldsymbol{x}}(s_i),s_i) = \frac{\partial\boldsymbol{w}(s_i)}{\partial t} + \nabla\cdot\mathcal{F}(\boldsymbol{w}(s_i)) + \nabla\mathcal{G}(\boldsymbol{w}(s_i)) + S = \mathbf{0} \tag{9}$$

After discretization, we have derived our discrete governing equation as

$$\frac{\partial\bar{\mathbf{w}}_i}{\partial t} + \sum_{j\in\kappa(i)}\boldsymbol{\phi}_{ij}(\mathbf{w}_{ij},\mathbf{w}_{ji},\boldsymbol{\nu}_{ij}) - \sum_{T_i\in\lambda(i)}\int_{T_j}\mathbb{K}\nabla\mathbf{w}\nabla\phi_j dx = \mathbf{0} \tag{10}$$ eq:nse_final_discretized_2

For the pupose of Sensitivity analysis around a given steady state, the fluid-state vector does not change in time, and can thus be ommitted, also we summaraize the

convective and the diffusivee part as $\mathbf{R}$

$$\underbrace{\underbrace{\cancelto{0}{\frac{\partial \bar{\mathbf{w}}_i}{\partial t}} + \underbrace{\sum_{j \in \kappa(i)} \boldsymbol{\phi}_{ij}(\mathbf{w}_{ij}, \mathbf{w}_{ji}, \boldsymbol{\nu}_{ij})}_{\mathbf{R}^i} - \underbrace{\sum_{T_i \in \lambda(i)} \int_{T_j} \mathbb{K} \nabla \mathbf{w} \nabla \phi_j dx}_{\mathbf{R}^v}}_{\mathbf{R}} = \mathbf{0}}_{\texttt{eq:nse\_final\_discretized\_notationchange}} \tag{11}$$

where $\mathbf{R}^i$ denotes the inviscid residual and $\mathbf{R}^v$ the viscous contribution. We now can express the steady state equation in very compact form as:

$$\mathbf{R} = \mathbf{0} \tag{12}$$ eq:discrete_steady_state

In general, this residual depends on the fluid state solution $\mathbf{w}$ as well as on a potential mesh movement $\mathbf{x}$, both of which can be dependent on the abstract design variable. Additionally, a direct dependence on s might be encountered too.

$$\mathbf{R}(\mathbf{w}(s), \mathbf{x}(s), s) = \mathbf{0} \tag{13}$$

The total derivative of the residual with respect to the design variable can thus be expanded via the chain rule to

$$\frac{\mathrm{d}\mathbf{R}}{\mathrm{d}s_i} = \mathbf{0} = \frac{\mathrm{d}\mathbf{R}}{\mathrm{d}s_i} + \frac{\partial \mathbf{R}}{\partial \mathbf{w}} \frac{\mathrm{d}\mathbf{w}}{\mathrm{d}s_i} + \frac{\partial \mathbf{R}}{\partial \mathbf{x}} \frac{\mathrm{d}\mathbf{x}}{\mathrm{d}s_i} \tag{14}$$

Therefore the total derivative of the fluid state with respect to the shape variable, needed in equation (2), can be obtained by solving

$$\frac{\partial \mathbf{R}}{\partial \mathbf{w}} \frac{\mathrm{d}\mathbf{w}}{\mathrm{d}s_i} = -\frac{\mathrm{d}\mathbf{R}}{\mathrm{d}s_i} - \frac{\partial \mathbf{R}}{\partial \mathbf{x}} \frac{\mathrm{d}\mathbf{x}}{\mathrm{d}s_i} \tag{15}$$ eq:dfstate_by_absvarI

In this equation, $\dfrac{\partial \mathbf{R}}{\partial s_i}$ can be computed analytically or by **FD!**. This part is the most cumbersome of the whole sensitivity analysis, which is why we denoted a full chapter **??** to it.
The derivative of the mesh motion with respect to the abstract variables is often denoted as "shape gradient". It can be divided into two components:

- The interface component $\dfrac{\mathrm{d}\dot{\boldsymbol{x}}_\Gamma}{\mathrm{d}s_i}$, which is associated with the grid points lying on the fluid boundary
- The interior component $\dfrac{\mathrm{d}\dot{\boldsymbol{x}}_\Omega}{\mathrm{d}s_i}$, which is associated with the grid points located in the interior Ω of the computational domain.

The interface component is determined by the structure. Having obtained this one, the interior component can be computed by solving an auxiliary, fictitious Dirichlet problem:

$$\frac{\mathrm{d}\dot{\boldsymbol{x}}_\Omega}{\mathrm{d}s_i} = -\left[\bar{\mathbf{K}}_{\Omega\Omega}^{-1}\bar{\mathbf{K}}_{\Omega\Gamma}\right]\frac{\mathrm{d}\dot{\boldsymbol{x}}_\Gamma}{\mathrm{d}s_i} \tag{16}$$

where $\bar{\mathbf{K}}$ is a pseudo stiffness matrix that can be obtained by a simple spring analogy or similar approaches. For the later introduced Embedded framework, $\frac{\mathrm{d}\dot{\boldsymbol{x}}}{\mathrm{d}s_i}$ is the position vector of the embedded discrete surface

1.3 Full sensitivity equation

After inserting (5), (8) and (15) into (2), one can derive the final sensitivity eqautions for the special case of a rigid or non-existing structure as

$$\left.\frac{\mathrm{d}q_j}{\mathrm{d}s_i}\right|_{\mathbf{w}_0} = -\left.\frac{\mathrm{d}q_j}{\mathrm{d}\boldsymbol{w}}\right|_{\mathbf{w}_0}\left[\left.\frac{\partial\mathbf{R}}{\partial\mathbf{w}}\right|_{\boldsymbol{w}_0}\right]^{-1}\left(\left.\frac{\partial\mathbf{R}}{\partial s_i}\right|_{\boldsymbol{w}_0} + \left[\alpha\left.\frac{\partial\mathbf{R}}{\partial\dot{\mathbf{x}}_\Omega}\right|_{\mathbf{w}_0}\left.\frac{\partial\mathbf{R}}{\partial\dot{\mathbf{x}}_\Gamma}\right|_{\mathbf{w}_0}\right]\begin{bmatrix}\alpha\bar{\mathbf{K}}_{\Omega\Omega}^{-1}\bar{\mathbf{K}}_{\Omega\Gamma}\\ \mathbf{I}\end{bmatrix}\frac{\mathrm{d}\dot{\mathbf{x}}_\Gamma}{\mathrm{d}s_i}\right) \tag{17}$$

eq:full_sa_nostruct

$$\alpha = \begin{cases}1 \text{ in } \textbf{ALE!} \text{ framework}\\ 0 \text{ in Embedded framework}\end{cases} \tag{18}$$

2 Aero-elastic SA!

The **SA!** approach applied in this thesis is based on the work of [**?**], for deriving the **GSE!** (**GSE!**) of coupled systems. As introduced by the authors of[**?**], we utilize the three-field formulation of [**?**].
The derivative of the optimization criterion q_j, as introduced in Equation (**??**), with respect to the optimization variable s_i gives:

$$\frac{dq_j}{ds_i} = \frac{\partial q_j}{\partial s_i} + \frac{\partial q_j}{\partial \boldsymbol{u}} \frac{d\boldsymbol{u}}{ds_i} + \frac{\partial q_j}{\partial \dot{\boldsymbol{x}}} \frac{d\dot{\boldsymbol{x}}}{ds_i} + \frac{\partial q_j}{\partial \boldsymbol{w}} \frac{d\boldsymbol{w}}{ds_i} \tag{19}$$

$$= \frac{\partial q_j}{\partial s_i} + \begin{bmatrix} \frac{\partial q_j}{\partial \boldsymbol{u}} \\ \frac{\partial q_j}{\partial \dot{\boldsymbol{x}}} \\ \frac{\partial q_j}{\partial \boldsymbol{w}} \end{bmatrix}^T \cdot \begin{bmatrix} \frac{d\boldsymbol{u}}{ds_i} \\ \frac{d\dot{\boldsymbol{x}}}{ds_i} \\ \frac{d\boldsymbol{w}}{ds_i} \end{bmatrix} \tag{20}$$

where the partial derivatives, $\frac{\partial q_j}{\partial \boldsymbol{u}}$, $\frac{\partial q_j}{\partial \dot{\boldsymbol{x}}}$ and $\frac{\partial q_j}{\partial \boldsymbol{w}}$ can be directly evaluated within the discretized structure and fluid model through the relation between structural, aerodynamic design and abstract optimization parameters defied in the design model **??**. The cumbersome part are the derivatives $\frac{d\boldsymbol{u}}{ds_i}$, $\frac{d\dot{\boldsymbol{x}}}{ds_i}$ and $\frac{d\boldsymbol{w}}{ds_i}$ To obtain them, the governing equations (**??**)**TODO write down governing equations** have to be derived:

$$\begin{bmatrix} \frac{\partial \mathcal{S}_{gov}}{\partial s_i} \\ \frac{\partial \mathcal{D}_{gov}}{\partial s_i} \\ \frac{\partial \mathcal{F}_{gov}}{\partial s_i} \end{bmatrix} + \underbrace{\begin{bmatrix} \frac{\partial \mathcal{S}_{gov}}{\partial \boldsymbol{u}} & \frac{\partial \mathcal{S}_{gov}}{\partial \dot{\boldsymbol{x}}} & \frac{\partial \mathcal{S}_{gov}}{\partial \boldsymbol{w}} \\ \frac{\partial \mathcal{D}_{gov}}{\partial \boldsymbol{u}} & \frac{\partial \mathcal{D}_{gov}}{\partial \dot{\boldsymbol{x}}} & \mathbf{0} \\ \mathbf{0} & \frac{\partial \mathcal{F}_{gov}}{\partial \dot{\boldsymbol{x}}} & \frac{\partial \mathcal{F}_{gov}}{\partial \boldsymbol{w}} \end{bmatrix}}_{\boldsymbol{A}} \begin{bmatrix} \frac{\partial \boldsymbol{u}}{\partial s_i} \\ \frac{\partial \dot{\boldsymbol{x}}}{\partial s_i} \\ \frac{\partial \boldsymbol{w}}{\partial s_i} \end{bmatrix} = \mathbf{0} \tag{21}$$

eq:governing_equations_derivative

In this equations $\frac{\partial \mathcal{S}_{gov}}{\partial s_i}$ and $\frac{\partial \mathcal{F}_{gov}}{\partial s_i}$ can be again directly evaluated using the relation specified in the design model. The matrix of first derivatives $\boldsymbol{A}$ is from now on denoted as the "Jacobian of the optimization problem".
Combining the previous two equations, it follows that the total derivative of the

optimization criterion with respect to the abstract variables can be expressed as:

$$\frac{dq_j}{ds_i} = \frac{\partial q_j}{\partial s_i} - \underbrace{\begin{bmatrix} \frac{\partial q_j}{\partial \boldsymbol{u}} \\ \frac{\partial q_j}{\partial \dot{\boldsymbol{x}}} \\ \frac{\partial q_j}{\partial \boldsymbol{w}} \end{bmatrix}^T}_{n_{\boldsymbol{q}} \times n_{eq}} \underbrace{\boldsymbol{A}^{-1}}_{n_{eq} \times n_{eq}} \underbrace{\begin{bmatrix} \frac{\partial \mathcal{S}_{gov}}{\partial s_i} \\ \frac{\partial \mathcal{D}_{gov}}{\partial s_i} \\ \frac{\partial \mathcal{F}_{gov}}{\partial s_i} \end{bmatrix}}_{n_{eq} \times n_{\boldsymbol{s}}} \quad \texttt{eq:totderiv\_optcritBYabsvar} \tag{22}$$

Where n_{eq} is the total number of equations(e.g. five fluid state equations for the compressible NSG in 3D, three equations of the mesh motions and another three equations for the structure motion), n_q is the number of optimization criteria and n_s is the number of abstract variables.

2.1 Direct vs. adjoint approach

Equation 22 suggests, that there are two alternatives to compute vector-matrix-vector product above.

Direct approach Firstly, one could first compute the derivatives of the aeroelastic response for each abstract variable and perform the matrix product with $\boldsymbol{A}$:

$$\begin{bmatrix} \frac{d\boldsymbol{u}}{ds_i} \\ \frac{d\dot{\boldsymbol{x}}}{ds_i} \\ \frac{d\boldsymbol{w}}{ds_i} \end{bmatrix} = -\boldsymbol{A}^{-1} \begin{bmatrix} \frac{\partial \mathcal{S}_{gov}}{\partial s_i} \\ \frac{\partial \mathcal{D}_{gov}}{\partial s_i} \\ \frac{\partial \mathcal{F}_{gov}}{\partial s_i} \end{bmatrix} \quad \text{and then} \quad \texttt{eq:direct\_approach} \tag{23}$$

$$\frac{dq_j}{ds_i} = \frac{\partial q_j}{\partial s_i} - \begin{bmatrix} \frac{\partial q_j}{\partial \boldsymbol{u}} \\ \frac{\partial q_j}{\partial \dot{\boldsymbol{x}}} \\ \frac{\partial q_j}{\partial \boldsymbol{w}} \end{bmatrix}^T \begin{bmatrix} \frac{d\boldsymbol{u}}{ds_i} \\ \frac{d\dot{\boldsymbol{x}}}{ds_i} \\ \frac{d\boldsymbol{w}}{ds_i} \end{bmatrix} \tag{24}$$

Where the total complexity can be approximated as $\mathcal{O}(n_{eq}^2 n_{\boldsymbol{s}} + n_{\boldsymbol{q}} n_{eq} n_{\boldsymbol{s}})$

Adjoint approach Secondly, one could also first compute the derivatives of the optimization criteria and multiply with the Jacobian before substituting this into Equation (22):

$$\begin{bmatrix} \boldsymbol{a_u} \\ \boldsymbol{a_{\dot{x}}} \\ \boldsymbol{a_w} \end{bmatrix} = \boldsymbol{A}^{-T} \begin{bmatrix} \frac{\partial q_j}{\partial \boldsymbol{u}} \\ \frac{\partial q_j}{\partial \dot{\boldsymbol{x}}} \\ \frac{\partial q_j}{\partial \boldsymbol{w}} \end{bmatrix} \qquad \texttt{eq:firststep\_adjoint} \tag{25}$$

$$\frac{dq_j}{ds_i} = \frac{\partial q_j}{\partial s_i} - \begin{bmatrix} \boldsymbol{a_u} \\ \boldsymbol{a_{\dot{x}}} \\ \boldsymbol{a_w} \end{bmatrix}_j^T \begin{bmatrix} \frac{\partial \mathcal{S}_{gov}}{\partial s_i} \\ \frac{\partial \mathcal{D}_{gov}}{\partial s_i} \\ \frac{\partial \mathcal{F}_{gov}}{\partial s_i} \end{bmatrix} \tag{26}$$

Where the total complexity can be approximated as $\mathcal{O}(n_{eq}^2 n_{\boldsymbol{q}} + n_{\boldsymbol{q}} n_{eq} n_{\boldsymbol{s}})$

If one or the other approach is to be preferred depends in on the optimization setup, particularly the number of optimization criteria and the number of optimization variables. Looking at the orders above, one can conclude that if the number of abstract parameters $n_{\boldsymbol{s}}$ is smaller than the number of optimization criteria, the direct approach is more efficient, otherwise the adjoint approach is to be preferred. Additionally the on can argue, that the relevant term in the orders above is the one with n_{eq}^2 since it dominates the sum. Therefore, depending on whether n_s or n_q is bigger, the direct or the adjoint method is to be preferred.

2.1.1 Direct SA! for the Euler equations

The A matrix for the direct approach in ALE formulation looks like

$$\boldsymbol{A} = \begin{bmatrix} \frac{\partial \mathcal{S}_{gov}}{\partial \boldsymbol{u}} & \frac{\partial \mathcal{S}_{gov}}{\partial \dot{\boldsymbol{x}}} & \frac{\partial \mathcal{S}_{gov}}{\partial \boldsymbol{w}} \\ \frac{\partial \mathcal{D}_{gov}}{\partial \boldsymbol{u}} & \frac{\partial \mathcal{D}_{gov}}{\partial \dot{\boldsymbol{x}}} & \boldsymbol{0} \\ \boldsymbol{0} & \frac{\partial \mathcal{F}_{gov}}{\partial \dot{\boldsymbol{x}}} & \frac{\partial \mathcal{F}_{gov}}{\partial \boldsymbol{w}} \end{bmatrix} = \begin{bmatrix} \mathbf{K} & \frac{\partial \boldsymbol{P}_T}{\partial \dot{\boldsymbol{x}}} & \frac{\partial \boldsymbol{P}_T}{\partial \boldsymbol{w}} \\ \begin{bmatrix} \mathbf{K}_{\Omega\Gamma}\mathbf{T}_u \\ \mathbf{T}_u \end{bmatrix} & \begin{bmatrix} \mathbf{K}_{\Omega\Omega} & \mathbf{0} \\ \mathbf{0} & \mathbf{I} \end{bmatrix} & \mathbf{0} \\ \mathbf{0} & \frac{\partial \mathbf{F}_2}{\partial \dot{\boldsymbol{x}}_\Omega} & \boldsymbol{H}_2 \end{bmatrix} \qquad \texttt{eq:Amatrix\_ALE} \tag{27}$$

Where, $\boldsymbol{H}_2$ is the Jacobian of the second order row flux. It shall be noted that constructing this Jacobian is not a trivial issue and takes up a lot of computational resources, especially for **FV!** (**FV!**), as described in [**?**]. Investigation into whether this term can be approximated at first order were carried out in [**?**] and [**?**].
Furthermore, [**?**] considered replacing the two mesh motion related matrices $\frac{\partial \boldsymbol{P}_T}{\partial \dot{\boldsymbol{x}}}$ and $\frac{\partial \mathbf{F}_2}{\partial \dot{\boldsymbol{x}}_\Omega}$ by a transpirational boundary condition. The consequences of this approach are also investigated in [**?**] and [**?**].
This thesis, however, does not use any of this simplifications.
The derivation of the sensitivities, can be achieved in a staggered scheme, very similar to the one, described in **??TODO**. It consists of five steps.

1) Update the structural displacement sensitivity to a new time step BY differentiating equations (**??**) and (**??**) and applying an under relaxation, we can obtain

$$\frac{\mathrm{d}\boldsymbol{u}^{(n)}}{\mathrm{d}s_i} = (1-\theta)\frac{\mathrm{d}\boldsymbol{u}^{(n)}}{\mathrm{d}s_i} + \theta\frac{\mathrm{d}\bar{\boldsymbol{u}}}{\mathrm{d}s_i} \quad \texttt{eq:underrelax\_structdisp} \tag{28}$$

where $\bar{\boldsymbol{u}}$ is obtained from:

$$\mathbf{K}\frac{\mathrm{d}\bar{\boldsymbol{u}}}{\mathrm{d}s_i} = \frac{\partial TODO}{\partial s_i} + \frac{\partial \mathbf{T}_u{}^{(n)}}{\partial s_i} - \frac{\partial \mathbf{K}}{\partial s_i}\boldsymbol{u} \quad \texttt{eq:fictious\_structdisp} \tag{29}$$

2) Transfer sensitivity of structure motion to the interface

$$\frac{\mathrm{d}\boldsymbol{u}_T^{(n)}}{\mathrm{d}s_i} = \mathbf{T}_u\frac{\mathrm{d}\boldsymbol{u}^{(n)}}{\mathrm{d}s_i} \quad \texttt{eq:interface\_projections} \tag{30}$$

3) Compute derivative of fluid mesh motion The fluid mesh motion is computed by solving the pseudo Dirichlet problem as described in [**?**]. By design, the fictions stiffness matrix $\bar{\mathbf{K}}$ does not depend on the abstract optimization variables $\boldsymbol{s}$

$$\bar{\mathbf{K}}_{\Omega\Omega}\frac{\mathrm{d}\dot{\boldsymbol{x}}_\Omega^{(n)}}{\mathrm{d}s_i} = -\bar{\mathbf{K}}_{\Omega\Gamma}\frac{\mathrm{d}\dot{\boldsymbol{x}}_\Gamma^{(n)}}{\mathrm{d}s_i} \quad \texttt{eq:mms\_domain} \tag{31}$$

with

$$\frac{\mathrm{d}\dot{\boldsymbol{x}}_\Gamma^{(n)}}{\mathrm{d}s_i} = \frac{\mathrm{d}\dot{\boldsymbol{x}}_\Gamma^{(n)}}{\mathrm{d}s_i} \tag{32}$$

4) Compute the sensitivity of the fluid state variables The derivatives of the fluid state variables are computed by

$$\boldsymbol{H}_2\frac{\mathrm{d}\boldsymbol{w}^{(n+1)}}{\mathrm{d}s_i} = \frac{\partial \boldsymbol{F}_2}{\partial s_i} - \frac{\partial \boldsymbol{F}_2}{\partial \dot{\boldsymbol{x}}}\frac{\mathrm{d}\dot{\boldsymbol{x}}}{\mathrm{d}s_i}^{(n)} \tag{33}$$

5) Compute the sensitivity of the structure load vector The derivative of the fluid load with respect to the abstract variables can be computed by the third of Equations (23) with the definition of $\boldsymbol{A}$ as specified in (27).

$$\frac{\partial \boldsymbol{P}_F{}^{(n+1)}}{\partial s_i} = \frac{\partial \boldsymbol{P}_F{}^{(n+1)}}{\partial \dot{\boldsymbol{x}}} \frac{\mathrm{d}\dot{\boldsymbol{x}}^{(n)}}{\mathrm{d}s_i} + \frac{\partial \boldsymbol{P}_F{}^{(n+1)}}{\partial \boldsymbol{w}} \frac{\mathrm{d}\boldsymbol{w}^{(n)}}{\mathrm{d}s_i} \tag{34}$$

eq:deriv_floadBYabsvar

and compute project it onto the structure via

$$\frac{\partial \boldsymbol{P}_T{}^{(n+1)}}{\partial s_i} = \mathbf{T}_p \frac{\partial \boldsymbol{P}_F{}^{(n+1)}}{\partial s_i} \tag{35}$$

The convergence of the staggered algorithm can be monitored via

$$\left\| \mathbf{K} \frac{\mathrm{d}\bar{\boldsymbol{u}}^{(n+1)}}{\mathrm{d}s_i} - \frac{\partial TODO}{\partial s_i} - \frac{\partial \boldsymbol{P}_T{}^{(n+1)}}{\partial s_i} + \frac{\partial \mathbf{K}}{\partial s_i} \right\|_2 \leq \epsilon^{SA} \left\| \mathbf{K} \frac{\mathrm{d}\bar{\boldsymbol{u}}^{(0)}}{\mathrm{d}s_i} - \frac{\partial TODO}{\partial s_i} - \frac{\partial \boldsymbol{P}_T{}^{(0)}}{\partial s_i} + \frac{\partial \mathbf{K}}{\partial s_i} \right\|_2 \tag{36}$$

$$\left\| \boldsymbol{H}_2 \frac{\mathrm{d}\boldsymbol{w}^{(n+1)}}{\mathrm{d}s_i} + \frac{\partial \boldsymbol{F}_2}{\partial s_i} + \frac{\partial \boldsymbol{F}_2}{\partial \dot{\boldsymbol{x}}} \frac{\mathrm{d}\dot{\boldsymbol{x}}}{\mathrm{d}s_i}^{(n+1)} \right\|_2 \leq \epsilon^{SA} \left\| \boldsymbol{H}_2 \frac{\mathrm{d}\boldsymbol{w}^{(0)}}{\mathrm{d}s_i} + \frac{\partial \boldsymbol{F}_2}{\partial s_i} + \frac{\partial \boldsymbol{F}_2}{\partial \dot{\boldsymbol{x}}} \frac{\mathrm{d}\dot{\boldsymbol{x}}}{\mathrm{d}s_i}^{(0)} \right\|_2 \tag{37}$$

2.1.2 Adjoint SA! for the Euler equations

The adjoint SA follows the same scheme as the direct one Equation (25) can be written as:

$$\begin{bmatrix} \mathbf{K} & \begin{bmatrix} \frac{\partial \boldsymbol{P}_T}{\partial \dot{\boldsymbol{x}}_\Omega} & \frac{\partial \boldsymbol{P}_T}{\partial \dot{\boldsymbol{x}}_\Gamma} \end{bmatrix} & \frac{\partial \boldsymbol{P}_T}{\partial \boldsymbol{w}} \\ \begin{bmatrix} \mathbf{K}_{\Omega\Gamma}\mathbf{T}_u \\ \mathbf{T}_u \end{bmatrix} & \begin{bmatrix} \mathbf{K}_{\Omega\Omega} & \mathbf{0} \\ \mathbf{0} & \mathbf{I} \end{bmatrix} & \mathbf{0} \\ \mathbf{0} & \begin{bmatrix} \frac{\partial \mathbf{F}_2}{\partial \dot{\boldsymbol{x}}_\Omega} & \frac{\partial \mathbf{F}_2}{\partial \dot{\boldsymbol{x}}_\Gamma} \end{bmatrix} & \boldsymbol{H}_2 \end{bmatrix}^T \begin{bmatrix} \boldsymbol{a}_{\boldsymbol{u}} \\ \begin{bmatrix} \boldsymbol{a}_{\dot{\boldsymbol{x}}_\Omega} \\ \boldsymbol{a}_{\dot{\boldsymbol{x}}_\Gamma} \end{bmatrix} \\ \boldsymbol{a}_{\boldsymbol{w}} \end{bmatrix} = \begin{bmatrix} \frac{\partial q_j}{\partial \boldsymbol{u}} \\ \begin{bmatrix} \frac{\partial \boldsymbol{q}}{\partial \dot{\boldsymbol{x}}_\Omega} \\ \frac{\partial \boldsymbol{q}}{\partial \dot{\boldsymbol{x}}_\Gamma} \end{bmatrix} \\ \frac{\partial q_j}{\partial \boldsymbol{w}} \end{bmatrix} \tag{38}$$

eq:adjoint_equation

TODO check if the index j is really required here! A stated earlier, the matrices $\frac{\partial \boldsymbol{q}}{\partial \dot{\boldsymbol{x}}}$ and $\frac{\partial \boldsymbol{q}}{\partial \boldsymbol{w}}$ can be computed analytically. As for $\boldsymbol{H}_2$, we follow the methodology outlined in [?] for evaluating ans storing it efficiently as the product of flux operators. Again the staggered procedure for solving the adjoint state problem shares the same computational kernels with the partitioned aeroelastic scheme described in [?]

1) Update the adjoint structure displacement to the new time step

$$\boldsymbol{a_u}^{(n+1)} = (1-\theta)\boldsymbol{a_u}^{(n)} + \theta\bar{\boldsymbol{a_u}}^{(n+1)} \tag{39}$$

$$\tag{40}$$

where $\bar{\boldsymbol{a_u}}^{(n+1)}$ is obtained from

$$\mathbf{K}\bar{\boldsymbol{a_u}}^{(n+1)} = \frac{\partial \boldsymbol{q}}{\partial \boldsymbol{u}} - \mathbf{K}_{\Omega\Gamma}\mathbf{T}_u\boldsymbol{a_{x\Omega}}^{(n)} + \mathbf{T}_u\boldsymbol{a_{x\Gamma}}^{(n)} \tag{41}$$

TODO derive this equation

2) Compute the adjoint fluid state by solving

$$\boldsymbol{H_2}^T\boldsymbol{a_w}^{(n+1)} = \frac{\partial \dot{\boldsymbol{x}}}{\partial \boldsymbol{w}} + \frac{\partial \boldsymbol{P}_T}{\partial \boldsymbol{w}^T}\boldsymbol{a_u}^{(n+1)} \tag{42}$$

TODO derive this equation Again, $\dfrac{\partial \boldsymbol{q}}{\partial \boldsymbol{w}}$ is computed analytically from the relations defined in the design and aeroelastic model.

3) Compute adjoint mesh motion in domain and on the interface

$$\bar{\mathbf{K}}_{\Omega\Omega}\boldsymbol{a_{x\Omega}}^{(n+1)} = \frac{\partial \boldsymbol{q}}{\partial \dot{\boldsymbol{x}}_\Omega} - \frac{\partial \boldsymbol{P}_T}{\partial \dot{\boldsymbol{x}}_\Omega}\boldsymbol{a_u}^{(n+1)} - \frac{\partial \mathbf{F}_1{}^T}{\partial \dot{\boldsymbol{x}}_\Omega}\boldsymbol{a_x}^{(n+1)} \text{ in } \Omega \tag{43}$$

TODO check equations And the adjoint mesh motion on the interface is computed as $\boldsymbol{a_{x\Gamma}}^{(n+1)}$

$$\boldsymbol{a_{x\Gamma}}^{(n+1)} = \frac{\partial \boldsymbol{q}}{\partial \dot{\boldsymbol{x}}_{Gamma}} + \frac{\partial \boldsymbol{P}_T}{\partial \dot{\boldsymbol{x}}_\Gamma}\boldsymbol{a_u}^{(n+1)} - \frac{\partial \mathbf{F}_2}{\partial \dot{\boldsymbol{x}})_\Gamma}^T \text{ on } \Gamma \tag{44}$$

where $\dfrac{\partial \boldsymbol{q}}{\partial \boldsymbol{w}}$ is computed analytically.

The convergence of the staggered adjoint optimization algorithm can be monitored via

$$\left\| \boldsymbol{R_q} - \boldsymbol{A}^T\boldsymbol{a}^{(n+1)} \right\| \leq \epsilon^{SA} \left\| \boldsymbol{R_q} \right\| \tag{45}$$