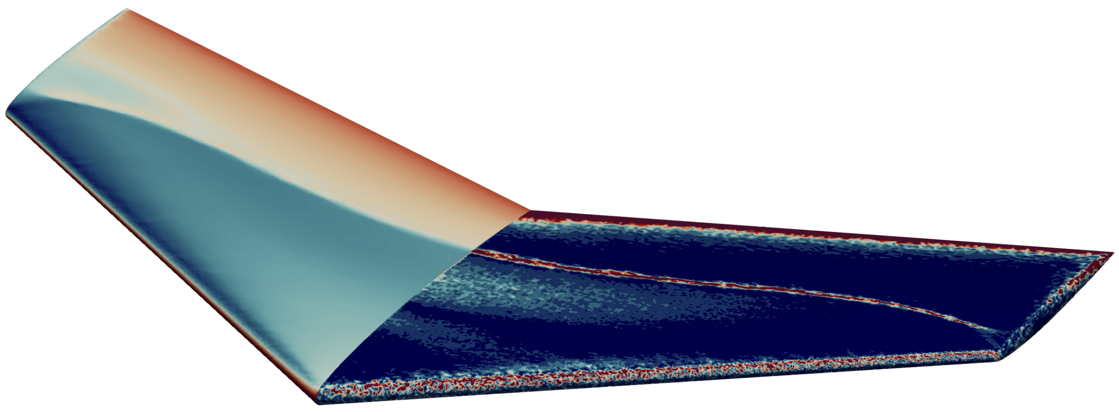
DARTFlo - Discrete Adjoint for Rapid Transonic Flows

Theory manual and quick reference guide

Adrien Crovato



Department of Aerospace & Mechanical Engineering

Abstract

This document provides the mathematical formulation of the main equations implemented in DARTFlo [1], version 1.3.0, December 2024. For more detailed information about the original implementation, please refer to the author's PhD thesis [1]. Additionally, more details about the mathematical foundation can be found in the journal article [2].

This theory manual and quick reference guide is organized as follows. Section 1 presents the formulation of the full potential equation and the mesh morphing laws. Section 2 presents the formulation of their partial gradients. Section 3 presents the direct and adjoint solution procedures. Finally, section 4 gives an overview of the available API as well as their configuration parameters.

[1] https://gitlab.uliege.be/am-dept/dartflo, accessed November 2024.

Contents

1 Model equations

This section presents the formulation and the discretization of the full potential model and the mesh morphing laws.

1.1 Full potential

This section details the formulation of the full potential equation written in residual form and of the aerodynamic loads.

1.1.1 Residuals

The steady full potential equation is derived form the Navier-Stokes equations by assuming that the fluid is inviscid, and that the flow is steady and isentropic. As a consequence, the vorticity is conserved. Since the freestream flow is irrotational, the whole flow is therefore irrotational and the velocity derives from a potential ϕ. Considering a domain Ω enclosed by a surface $\Gamma = \Gamma_\mathrm{f} \cup \Gamma_\mathrm{b}$, as depicted in figure 1.1, the full potential equation can be written in weak form as

$$\int_\Omega \rho\nabla\phi\cdot\nabla\psi\,dV - \int_\Gamma \overline{\rho\nabla\phi}\cdot\hat{\mathbf{n}}\psi\,dS = 0, \qquad \forall\psi\in\Omega, \tag{1.1}$$

where ψ is a test function, $\hat{\mathbf{n}}$ is the unit vector normal to Γ pointing inwards, and where the density ρ is given by the isentropic flow relationship,

$$\rho = \rho_\infty\left[1 + \frac{\gamma-1}{2}M_\infty^2\left(1-|\nabla\phi|^2\right)\right]^{\frac{1}{\gamma-1}}. \tag{1.2}$$

In Equation 1.2, ρ_∞ is the freestream density, γ is the heat capacity ratio and M_∞ is the freestream Mach number. Note that the term $|\nabla\phi|$ in Equation 1.2, which is the magnitude of the total velocity, has been normalized by the freestream velocity. An important limitation of the nonlinear potential equation is the isentropicity assumption, which restricts its use to transonic flows with embedded weak shocks only. A common upper limit for the local normal Mach number upstream of the shock is $M_n < 1.3$ [3].

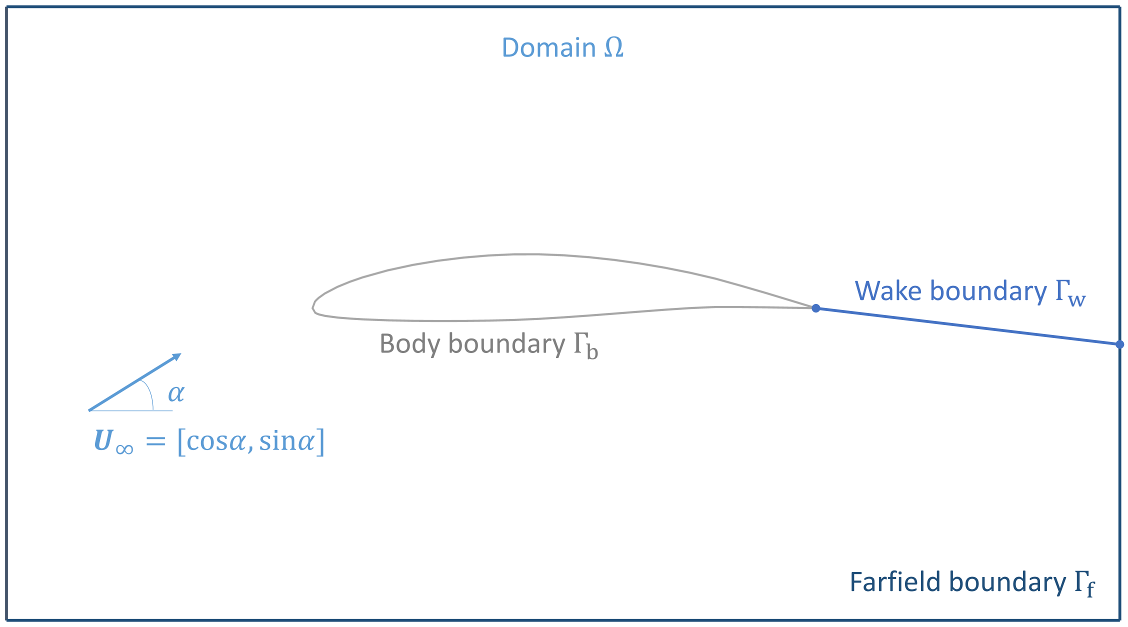

Domain Ω
Wake boundary Γ_w
Body boundary Γ_b
α
$\boldsymbol{U}_\infty = [\cos\alpha, \sin\alpha]$
Farfield boundary Γ_f

Figure 1.1: Typical domain used for a finite element computation, illustrated in two dimensions for simplicity.

The boundary surface Γ is split into a farfield boundary Γ_f, and a body boundary Γ_b, as depicted in Figure 1.1, onto which Neumann boundary conditions are applied. Such boundary conditions impose a flux through the boundaries of the domain and are naturally recovered in the second term of the weak formulation of the full potential equation 1.1. Since the derivative of the potential is the velocity, the weak form of the Neumann boundary condition can be written as

$$\begin{aligned} \int_{\Gamma_\mathrm{f}} \overline{\rho \nabla \phi} \cdot \hat{\mathbf{n}} \psi \, dS &= \int_{\Gamma_\mathrm{f}} \rho_\infty \mathbf{U}_\infty \cdot \hat{\mathbf{n}} \psi \, dS, \qquad && \forall \psi \in \Gamma_\mathrm{f}, \\ \int_{\Gamma_\mathrm{b}} \overline{\rho \nabla \phi} \cdot \hat{\mathbf{n}} \psi \, dS &= \int_{\Gamma_\mathrm{b}} \rho U_\mathrm{bl} \psi \, dS, && \forall \psi \in \Gamma_\mathrm{b}, \end{aligned} \tag{1.3}$$

where $\mathbf{U}_\infty$ is the freestream velocity vector given by

$$\mathbf{U}_\infty = \begin{bmatrix} \cos\alpha\cos\beta \\ \sin\beta \\ \sin\alpha\cos\beta \end{bmatrix}, \tag{1.4}$$

and where α is the angle of attack and β is the angle of sideslip. The second equation in (1.3) is a transpiration boundary condition. The compressible blowing velocity ρU_bl is set to zero in a purely inviscid case, while it is provided by a boundary layer solver such as BLASTER [5] [2] in the context of viscous-inviscid interaction. Additionally, wake boundary conditions and the Kutta condition need to be enforced to allow potential flows to produce aerodynamic loads. This is accomplished by creating a flat wake sheet, denoted Γ_w, extending from the trailing edge of

[2] https://gitlab.uliege.be/am-dept/blaster, accessed November 2024.

any lifting body to the farfield boundary located downstream of these bodies, as depicted in Figure 1.1. The unknown potential value is discontinuous across the wake, and two boundary conditions are applied to restore the continuity in the flow variables. The first condition prescribes the equality of the mass flux across the wake,

$$\int_{\Gamma_\mathrm{w}} [\![\rho\nabla\phi\cdot\hat{\mathbf{n}}]\!]\psi\, dS = 0, \qquad \forall\psi\in\Gamma_{\mathrm{w,l}}, \tag{1.5}$$

and the second condition prescribes the equality of the pressure across the wake,

$$\int_{\Gamma_\mathrm{w}} [\![|\nabla\phi|^2]\!]\Psi\, dS = 0, \qquad \forall\Psi\in\Gamma_{\mathrm{w,u}}, \tag{1.6}$$

where Ψ is a stabilized test function, the double square bracket indicates a jump between the quantities on the upper and lower sides of the wake, and the subscripts u and l refer to the upper and lower sides of the wake, respectively. Similarly, the Kutta condition, prescribing the equality of the pressure on both sides of the trailing edge, writes

$$\int_{\Gamma_{\mathrm{TE,u}}} \frac{1}{h^2}|\nabla\phi|^2\Psi\, dS - \int_{\Gamma_{\mathrm{TE,l}}} \frac{1}{h^2}|\nabla\phi|^2\Psi\, dS = 0, \qquad \forall\Psi\in\Gamma_{\mathrm{TE,u}}, \tag{1.7}$$

where h is the square root of the surface area of the trailing edge, and where $\Gamma_{\mathrm{TE,u}}$ and $\Gamma_{\mathrm{TE,l}}$ denote the suction and pressure sides of the trailing edge, respectively. Note that the transpiration boundary condition (1.3) is also enforced on wake sheets, on top of applying the wake boundary conditions.

Finally, supersonic regions of the flow need to be stabilized. The physical density is upwinded and replaced by

$$\tilde{\rho} = \rho - \mu(\rho - \rho_\mathrm{U}), \tag{1.8}$$

where ρ_U is the density evaluated at an upwind point, and where the switching function is defined as

$$\mu = \mu_\mathrm{C}\max\left(0,\ 1-\frac{M_\mathrm{C}^2}{M^2},\ 1-\frac{M_\mathrm{C}^2}{M_\mathrm{U}^2}\right). \tag{1.9}$$

The parameters μ_C, which controls the amplification of the density bias, and M_C, which controls the extent of the region where the bias is applied, are controlled by the numerical scheme. They are initialized to 2 and 0.925 in order to produce strong stabilization over a large portion of the flow. As the solution converges, they are varied to 1 and 0.975. These final values were chosen from the literature, as they are suitable for most cases. The parameters are updated each time the relative residual of the full potential equation drops below 10^{-2}. This specific switching function, whereby the Mach number M is replaced by the Mach number evaluated at an upwind point M_U whenever the supersonic flow decelerates, is chosen to bring additional numerical dissipation near shocks, hence improving the robustness of the method, as recommended by Habashi and Hafez [4].

The domain Ω and its boundary Γ are discretized using continuous Galerkin finite elements. An unstructured grid strategy is chosen in order to easily mesh three-dimensional complex shapes. The potential, test functions and coordinates are expressed as

$$\begin{aligned} \phi &= N_i\phi_i, \\ \psi &= N_i\psi_i, \\ x_k &= N_i x_{i,k} \end{aligned} \tag{1.10}$$

where N_i are the shape functions associated to an element, and interpolate the nodal values ϕ_i and ψ_i of the potential and the test functions, as well as the nodal coordinates $x_{i,k} = [x, y, z]_i$, on that element. Note that subscript k counts the dimensions. The shape functions are expressed locally on each element as

$$N_i = N_i(\xi_k) = N_i\left([\xi, \eta, \zeta]\right), \tag{1.11}$$

where ξ_k is the vector of coordinates attached to the reference frame of an element. The weak form of the full potential equation (1.1) must hold for any test function ψ. It can then be discretized and rewritten in residual form,

$$\begin{aligned} \mathbf{R}_\phi = &\sum_{\mathrm{e}} \int_{\Omega_\mathrm{e}} \tilde{\rho}_\mathrm{e} \nabla N_j\phi_j \cdot \nabla N_i \, dV_\mathrm{e} - \sum_{\mathrm{e}} \int_{\Gamma\mathrm{e}} \overline{\rho\nabla\phi}_\mathrm{e} \cdot \hat{\mathbf{n}}_\mathrm{e} N_i \, dS_\mathrm{e} \\ = &\, 0, \end{aligned} \tag{1.12}$$

where the subscript e refers to elemental quantities. The associated Neumann boundary conditions (1.3) become

$$\begin{aligned} \sum_{\mathrm{e}} \int_{\Gamma_{\mathrm{f}_\mathrm{e}}} \overline{\rho\nabla\phi}_\mathrm{e} \cdot \hat{\mathbf{n}}_\mathrm{e} N_i \, dS_\mathrm{e} &= \sum_{\mathrm{e}} \int_{\Gamma_{\mathrm{f}_\mathrm{e}}} \rho_\infty \mathbf{U}_\infty \cdot \hat{\mathbf{n}}_\mathrm{e} N_i \, dS_\mathrm{e}, \\ \sum_{\mathrm{e}} \int_{\Gamma_{\mathrm{b}_\mathrm{e}}} \overline{\rho\nabla\phi}_\mathrm{e} \cdot \hat{\mathbf{n}}_\mathrm{e} N_i \, dS_\mathrm{e} &= \sum_{\mathrm{e}} \int_{\Gamma_{\mathrm{b}_\mathrm{e}}} \rho U_{\mathrm{bl,e}} N_i \, dS_\mathrm{e}. \end{aligned} \tag{1.13}$$

The equality of mass flux across the wake (1.5) is enforced on the lower wake nodes as

$$\sum_{\mathrm{e}} \int_{\Gamma_{\mathrm{w,l}_\mathrm{e}}} \tilde{\rho}_\mathrm{e} \nabla N_j\phi_j \cdot \nabla N_i \, dS_\mathrm{e} - \sum_{\mathrm{e}} \int_{\Gamma_{\mathrm{w,u}_\mathrm{e}}} \tilde{\rho}_\mathrm{e} \nabla N_j\phi_j \cdot \nabla N_i \, dS_\mathrm{e} = 0, \tag{1.14}$$

and the equality of the pressure (1.6) is enforced on the upper wake nodes as

$$\sum_{\mathrm{e}} \int_{\Gamma_{\mathrm{w}_\mathrm{e}}} \left([\nabla\phi \cdot \nabla N_j\phi_j]_{\mathrm{w,u}} - [\nabla\phi \cdot \nabla N_j\phi_j]_{\mathrm{w,l}} \right) \left(N_i + \frac{h}{2}\tilde{U}_{\infty,k}\partial_{x_k} N_i \right)_{\mathrm{w,u}} dS_\mathrm{e} = 0. \tag{1.15}$$

where $\tilde{\mathbf{U}}_\infty = [1, 0, 0]$. Similarly, the equality of the pressure on the trailing edge (1.7) is prescribed as

$$\begin{aligned} &\sum_{\mathrm{e}} \int_{\Gamma_{\mathrm{TE,u}_\mathrm{e}}} \frac{1}{h^2} \nabla\phi \cdot \nabla N_j\phi_j \left(N_{i_{\mathrm{TE,u}}} + \frac{h}{2}\tilde{U}_{\infty,k}\partial_{x_k} N_i \right) dS_\mathrm{e} \\ -&\sum_{\mathrm{e}} \int_{\Gamma_{\mathrm{TE,l}_\mathrm{e}}} \frac{1}{h^2} \nabla\phi \cdot \nabla N_j\phi_j \left(N_{i_{\mathrm{TE,u}}} + \frac{h}{2}\tilde{U}_{\infty,k}\partial_{x_k} N_i \right) dS_\mathrm{e} = 0. \end{aligned} \tag{1.16}$$

Note that the contribution of the blowing velocity (1.13) is also added on wake elements.

1.1.2 Functional

The total aerodynamic load vector is obtained by multiplying the aerodynamic total load coefficient $\mathbf{C_F}$ by the freestream dynamic pressure,

$$\mathbf{F} = \frac{1}{2}\rho_\infty u_\infty^2 S_{\mathrm{ref}} \mathbf{C_F}. \tag{1.17}$$

The resulting aerodynamic load coefficient is computed by integrating the normalized pressure coefficient on the body surface,

$$\mathbf{C_F} = \frac{1}{S_{\mathrm{ref}}} \int_{\Gamma_{\mathrm{b}}} C_p \hat{\mathbf{n}} \, dS, \tag{1.18}$$

where S_{ref} is a reference area, and where the pressure coefficient is given by

$$C_p = \frac{2}{\gamma M_\infty^2} \left(\rho^\gamma - 1\right). \tag{1.19}$$

The aerodynamic load coefficients are obtained by projecting $\mathbf{C_F}$ on the lift and drag directions, yielding

$$C_L = \mathbf{C_F} \cdot \mathbf{e_L}, \qquad C_D = \mathbf{C_F} \cdot \mathbf{e_D}, \tag{1.20}$$

where the directions are defined with respect to the angle of attack α, and the angle of sideslip β,

$$\mathbf{e_L} = \begin{bmatrix} -\sin\alpha \\ 0 \\ \cos\alpha \end{bmatrix}, \qquad \mathbf{e_D} = \begin{bmatrix} \cos\alpha\cos\beta \\ \sin\beta \\ \sin\alpha\cos\beta \end{bmatrix}. \tag{1.21}$$

1.2 Mesh deformation

This section details the formulation of the linear elasticity laws written in residual form, driving the mesh morphing.

1.2.1 Residuals

An efficient way to deform the grid for the kind of wing deflections considered in practical aeroelasticity, is to use linear elasticity theory. The grid is assumed to behave like an elastic body, rigid near the deforming boundaries, and flexible elsewhere. Moreover, the linear elasticity equations can be easily solved by the finite element method, and require little supplementary implementation work.

For an elastic solid, the equilibrium between the internal and external forces can be written in weak form as

$$\int_\Omega \nabla\boldsymbol{\sigma} \cdot \nabla\psi \, dV - \int_\Gamma \overline{\nabla\boldsymbol{\sigma}} \cdot \hat{\mathbf{n}}\psi \, dS = \int_\Omega \mathbf{f}\psi \, dV, \qquad \forall\psi \in \Omega, \tag{1.22}$$

where the internal stress $\boldsymbol{\sigma}$ can be related to the displacement $\Delta\mathbf{x}$ using Hooke's constitutive law for linear isotropic solids,

$$\boldsymbol{\sigma} = \frac{E\nu}{2(1+\nu)(1-2\nu)}\mathrm{tr}\left(\nabla(\Delta\mathbf{x}) + \nabla(\Delta\mathbf{x})^{\mathrm{T}}\right)\mathbf{I} + \frac{E}{2(1+\nu)}\left(\nabla(\Delta\mathbf{x}) + \nabla(\Delta\mathbf{x})^{\mathrm{T}}\right). \tag{1.23}$$

The Young modulus E and the Poisson's ratio ν are constitutive parameters. In the present work, they are set to $1/V$ and 0, respectively, as suggested by Dwight [6]. As a result, the mesh behaves as a linear elastic solid, rigid close to the wing where the elements are small, and flexible in the farfield where the elements are large. Note that, in the context of mesh deformation, the external forces $\mathbf{f}$ are zero, and the deformation is driven by a Dirichlet boundary condition imposed on the moving boundary.

After discretization, equation (1.22) must hold for any test function ψ, and can therefore be rewritten as a set of equations,

$$\begin{aligned}\mathbf{R}_{\mathrm{x}} = \sum_{\mathrm{e}}\int_{\Omega_{\mathrm{e}}}\left[\frac{E_{\mathrm{e}}\nu_{\mathrm{e}}}{2(1+\nu_{\mathrm{e}})(1-2\nu_{\mathrm{e}})}\partial_{x_k}N_l\Delta x_{l,k}\delta_{ij} + \frac{E_{\mathrm{e}}}{2(1+\nu_{\mathrm{e}})}\left(\partial_{x_j}N_l\Delta x_{l,i} + \partial_{x_i}N_l\Delta x_{l,j}\right)\right]\partial_{x_j}N_l\,dV_{\mathrm{e}}\\ = 0.\end{aligned} \tag{1.24}$$

The Dirichlet boundary condition on the deforming surface are enforced as,

$$\overline{\Delta x_{i,j}}|_{\Gamma_{\mathrm{b}}} = \Delta x_{\mathrm{b}_{i,j}}. \tag{1.25}$$

On the wake, the periodic boundary conditions are discretized as follows. The upper wake volume element contributions are added to the lower wake equations, and the upper wake unknowns are prescribed to match the lower wake unknowns,

$$\begin{aligned}&\sum_{\mathrm{e}}\int_{\Omega_{\mathrm{w,l_e}}}\left[\frac{E_{\mathrm{e}}\nu_{\mathrm{e}}}{2(1+\nu_{\mathrm{e}})(1-2\nu_{\mathrm{e}})}\partial_{x_k}N_l\Delta x_{l,k}\delta_{ij} + \frac{E_{\mathrm{e}}}{2(1+\nu_{\mathrm{e}})}\left(\partial_{x_j}N_l\Delta x_{l,i} + \partial_{x_i}N_l\Delta x_{l,j}\right)\right]\partial_{x_j}N_l\,dV_{\mathrm{e}}\\ &+\sum_{\mathrm{e}}\int_{\Omega_{\mathrm{w,u_e}}}\left[\frac{E_{\mathrm{e}}\nu_{\mathrm{e}}}{2(1+\nu_{\mathrm{e}})(1-2\nu_{\mathrm{e}})}\partial_{x_k}N_l\Delta x_{l,k}\delta_{ij} + \frac{E_{\mathrm{e}}}{2(1+\nu_{\mathrm{e}})}\left(\partial_{x_j}N_l\Delta x_{l,i} + \partial_{x_i}N_l\Delta x_{l,j}\right)\right]\partial_{x_j}N_l\,dV_{\mathrm{e}}\\ &=0,\\ &\Delta x_{i,j}|_{\Gamma_{\mathrm{w,u}}} - \Delta x_{i,j}|_{\Gamma_{\mathrm{w,l}}} = 0.\end{aligned} \tag{1.26}$$

2 Partial gradients

This section presents the formulation of the discretized gradients of the full potential and mesh morphing equations. Note that the summation symbol has been dropped for conciseness.

2.1 Full potential

This section details the formulation of the partial gradients of full potential equation, and of the aerodynamic loads.

2.1.1 Residuals

The partial gradient of the potential residuals with respect to the potential variables, also known as the flow Jacobian, is given by,

$$\begin{aligned}\frac{\partial R_{\phi,i}}{\partial \phi_j} = &\int_{\Omega_\mathrm{e}} (1-\mu)\left[-M_\infty^2 \rho_\mathrm{e}^{2-\gamma} \partial_{x_k}\phi \partial_{x_k} N_j \partial_{x_k}\phi \partial_{x_k} N_i + \rho_\mathrm{e} \partial_{x_k} N_j \partial_{x_k} N_i\right] \, dV_\mathrm{e} \\ &+ \int_{\Omega_\mathrm{e}} \mu \left[-M_\infty^2 \rho_\mathrm{U}^{2-\gamma} \partial_{x_k}\phi_\mathrm{U} \partial_{x_k} N_{\mathrm{U},j} \partial_{x_k}\phi \partial_{x_k} N_i + \rho_\mathrm{U} \partial_{x_k} N_j \partial_{x_k} N_i\right] \, dV_\mathrm{e} \\ &- \int_{\Omega_\mathrm{e}} (\rho_\mathrm{e} - \rho_\mathrm{U}) \left[\frac{2\mu_\mathrm{C} M_\mathrm{C}^2}{M_\bullet^3} \left(\frac{1}{\sqrt{\partial_{x_k}\phi^2 a_\bullet^2}} + \frac{\gamma-1}{2} \frac{\sqrt{\partial_{x_k}\phi^2}}{\sqrt[3]{a_\bullet^2}}\right) \partial_{x_k}\phi \partial_{x_k} N_j \partial_{x_k}\phi \partial_{x_k} N_i\right] \, dV_\mathrm{e},\end{aligned} \tag{2.1}$$

where $a_\bullet$ is the speed of sound on an element and is computed as

$$a_\bullet = \sqrt{\frac{1}{M_\infty^2} + \frac{\gamma-1}{2}(1-|\nabla\phi|^2)}. \tag{2.2}$$

The Mach number $M_\bullet$ and the speed of sound $a_\bullet$ can be evaluated on the current element e or the upwind element U, depending on the switching function. In order to avoid non-physical large gradients which may appear at the trailing edge of the wingtip, the speed of sound is limited so that the local Mach number remains below $M < 1.7$. Similar to the residuals $\mathbf{R}_\phi$, the wake boundary conditions are prescribed in two steps. Firstly, the equality of the mass flux is enforced by adding the contributions of the upper wake nodes to the lower wake rows, instead of the upper wake rows, in the flow Jacobian matrix,

$$\frac{\partial R_{\phi,i}}{\partial \phi_j}|_{\mathrm{w,l}} \leftarrow \frac{\partial R_{\phi,i}}{\partial \phi_j}|_{\mathrm{w,l}} + \frac{\partial R_{\phi,i}}{\partial \phi_j}|_{\mathrm{w,u}}, \tag{2.3}$$

where the left pointing arrow denotes an assignment operator. Secondly, the following terms are then assembled on the upper wake rows,

$$\frac{\partial R_{\phi,i}}{\partial \phi_j}|_{\mathrm{w,u}} = 2\int_{\Gamma_\mathrm{we}} \left(N_i + \frac{h}{2}\tilde{U}_{\infty,k} \partial_{x_k} N_i\right)_{\mathrm{w,u}} \left([\partial_{x_k}\phi \partial_{x_k} N_j]_{\mathrm{w,u}} - [\partial_{x_k}\phi \partial_{x_k} N_j]_{\mathrm{w,l}}\right) \, dS_\mathrm{e}. \tag{2.4}$$

Finally, the Kutta condition is enforced by assembling the following terms on the upper trailing edge rows,

$$\begin{aligned}\frac{\partial R_{\phi,i}}{\partial \phi_j}|_{\mathrm{TE,u}} = &\int_{\Gamma_{\mathrm{TE,u_e}}} \frac{2}{h^2}\left(N_{i_{\mathrm{TE,u}}} + \frac{h}{2}\tilde{U}_{\infty,k}\partial_{x_k}N_i\right)\partial_{x_k}\phi\partial_{x_k}N_j \, dS_{\mathrm{e}} \\ &- \int_{\Gamma_{\mathrm{TE,l_e}}} \frac{2}{h^2}\left(N_{i_{\mathrm{TE,u}}} + \frac{h}{2}\tilde{U}_{\infty,k}\partial_{x_k}N_i\right)\partial_{x_k}\phi\partial_{x_k}N_j \, dS_{\mathrm{e}}.\end{aligned} \tag{2.5}$$

The partial gradient of the potential residuals with respect to the mesh coordinates is given by

$$\begin{aligned}\frac{\partial R_{\phi,i}}{\partial x_j} = &\int_{\Omega_{\mathrm{e}}} (1-\mu)\left(-M_\infty^2\rho_{\mathrm{e}}^{2-\gamma}\partial_{x_l}\phi\left(-J_{\mathrm{e},lk}^{-1}\partial_{x_j}J_{\mathrm{e},kl}\right)\partial_{x_l}\phi\right)\partial_{x_k}\phi\partial_{x_k}N_i \, dV_{\mathrm{e}} \\ &+ \int_{\Omega_{\mathrm{e}}} \mu\left(-M_\infty^2\rho_{\mathrm{U}}^{2-\gamma}\partial_{x_l}\phi_{\mathrm{U}}\left(-J_{\mathrm{U},lk}^{-1}\partial_{x_j}J_{\mathrm{U},kl}\right)\partial_{x_l}\phi_{\mathrm{U}}\right)\partial_{x_k}\phi\partial_{x_k}N_i \, dV_{\mathrm{e}} \\ &+ \int_{\Omega_{\mathrm{e}}} \left[(1-\mu)\rho_{\mathrm{e}} + \mu\rho_{\mathrm{U}}\right]\left[\partial_{x_l}\phi\left(-J_{\mathrm{e},lk}^{-1}\partial_{x_j}J_{\mathrm{e},kl}\right)\partial_{x_l}N_i + \partial_{x_l}N_i\left(-J_{\mathrm{e},lk}^{-1}\partial_{x_j}J_{\mathrm{e},kl}\right)\partial_{x_l}\phi\right] \, dV_{\mathrm{e}} \\ &- \int_{\Omega_{\mathrm{e}}} (\rho_{\mathrm{e}} - \rho_{\mathrm{U}})\left[\frac{2\mu_{\mathrm{C}}M_{\mathrm{C}}^2}{M_\bullet^3}\left(\frac{1}{\sqrt{\partial_{x_k}\phi^2 a_\bullet^2}} + \frac{\gamma-1}{2}\frac{\sqrt{\partial_{x_k}\phi^2}}{\sqrt[3]{a_\bullet^2}}\right)\partial_{x_l}\phi\left(-J_{\mathrm{e},lk}^{-1}\partial_{x_j}J_{\mathrm{e},kl}\right)\partial_{x_l}\phi\partial_{x_k}\phi\partial_{x_k}N_i\right] \, dV_{\mathrm{e}} \\ &+ \int_{\Omega_{\mathrm{e}}} \left[(1-\mu)\rho_{\mathrm{e}} + \mu\rho_{\mathrm{U}}\right]\partial_{x_k}\phi\partial_{x_k}N_i \, \partial_{x_j}dV_{\mathrm{e}}.\end{aligned} \tag{2.6}$$

The partial gradient of the Jacobian matrix of an element with respect to the mesh coordinates is computed as

$$\partial_{x_k}J_{\bullet,ij} = \frac{\partial}{\partial_{x_k}}\partial_{\xi_j}N_l x_{l,i}, \tag{2.7}$$

where $\bullet$ refers to a variable evaluated on the current element e or on the upstream element U. Since Gauss quadrature is used to compute the integrals in the finite elements method, computing the partial gradient of an elementary volume with respect to the mesh coordinates amounts to computing the partial gradient of the Jacobian matrix determinant of an element as

$$\partial_{x_k}dV_{\mathrm{e}} = \partial_{x_k}\det(J_{\mathrm{e},ij}) = \det(J_{\mathrm{e},ij})\mathrm{tr}(J_{\mathrm{e},ij}^{-1}\partial_{x_k}J_{\mathrm{e},ij}). \tag{2.8}$$

The contributions of the farfield boundary condition are not taken into account since the outer boundary is fixed. The partial gradient of the transpiration boundary condition on the body and on the wake surfaces with respect to the mesh coordinates is given by

$$\frac{\partial R_{\phi,i}}{\partial x_j}|_{\mathrm{b\cup w}} = \int_{\Gamma_{\mathrm{b_e}}\cup\Gamma_{\mathrm{w_e}}} \rho U_{\mathrm{bl,e}}N_i \, \partial_{x_j}dS_{\mathrm{e}}. \tag{2.9}$$

Computing the partial gradient of an elementary surface with respect to the mesh coordinates amounts to computing the partial gradient of the surface Jacobian matrix determinant of an element. For a two-dimensional surface in a three-dimensional space, this gradient is expressed as

$$\partial_{x_j}dS_{\mathrm{e}} = \partial_{x_j}\det(J_{\mathrm{S,e}}) = \frac{J_{\mathrm{S,e}}}{|J_{\mathrm{S,e}}|}\cdot\left(\partial_{x_j}\partial_\xi N_k x_k \times \partial_\eta N_k x_k + \partial_\xi N_k x_k \times \partial_{x_j}\partial_\eta N_k x_k\right). \tag{2.10}$$

The partial gradient of the wake boundary conditions with respect to the mesh coordinates are assembled in a similar way as the flow residuals: the contributions of the upper wake rows are first added to the lower wake rows, and the upper wake rows are then computed as

$$\begin{aligned}\frac{\partial R_{\phi,i}}{\partial x_j}|_{\mathrm{w,u}} = &\int_{\Gamma_{\mathrm{we}}} \left(\partial_{x_j}\left(\frac{h}{2}\tilde{U}_{\infty,k}\partial_{x_k}N_i\right)\right)_{\mathrm{w,u}} \left([\partial_{x_k}\phi\partial_{x_k}\phi]_{\mathrm{w,u}} - [\partial_{x_k}\phi\partial_{x_k}\phi]_{\mathrm{w,l}}\right)\,dS_{\mathrm{e}}\\ &+2\int_{\Gamma_{\mathrm{we}}}\left(N_i + \frac{h}{2}\tilde{U}_{\infty,k}\partial_{x_k}N_i\right)_{\mathrm{w,u}}\left[\partial_{x_l}\phi\left(-J^{-1}_{\mathrm{e},lk}\partial_{x_j}J_{\mathrm{e},kl}\right)\partial_{x_l}\phi\right]_{\mathrm{w,u}}\,dS_{\mathrm{e}}\\ &-2\int_{\Gamma_{\mathrm{we}}}\left(N_i + \frac{h}{2}\tilde{U}_{\infty,k}\partial_{x_k}N_i\right)_{\mathrm{w,u}}\left[\partial_{x_l}\phi\left(-J^{-1}_{\mathrm{e},lk}\partial_{x_j}J_{\mathrm{e},kl}\right)\partial_{x_l}\phi\right]_{\mathrm{w,l}}\,dS_{\mathrm{e}}\\ &+\int_{\Gamma_{\mathrm{we}}}\left(N_i + \frac{h}{2}\tilde{U}_{\infty,k}\partial_{x_k}N_i\right)_{\mathrm{w,u}}\left([\partial_{x_k}\phi\partial_{x_k}\phi]_{\mathrm{w,u}} - [\partial_{x_k}\phi\partial_{x_k}\phi]_{\mathrm{w,l}}\right)\,\partial_{x_j}dS_{\mathrm{e}}.\end{aligned} \tag{2.11}$$

Similarly, the partial gradient of the Kutta condition with respect to the mesh coordinates are computed as

$$\begin{aligned}\frac{\partial R_{\phi,i}}{\partial x_j}|_{\mathrm{TE,u}} = &-\int_{\Gamma_{\mathrm{TE,u_e}}}\frac{2}{h^3}\partial_{x_j}h\left(N_{i_{\mathrm{TE,u}}} + \frac{h}{2}\tilde{U}_{\infty,k}\partial_{x_k}N_i\right)\partial_{x_k}\phi\partial_{x_k}\phi\,dS_{\mathrm{e}}\\ &+\int_{\Gamma_{\mathrm{TE,u_e}}}\frac{1}{h^2}\partial_{x_j}\left(\frac{h}{2}\tilde{U}_{\infty,k}\partial_{x_k}N_i\right)\partial_{x_k}\phi\partial_{x_k}\phi\,dS_{\mathrm{e}}\\ &+\int_{\Gamma_{\mathrm{TE,u_e}}}\frac{2}{h^2}\left(N_{i_{\mathrm{TE,u}}} + \frac{h}{2}\tilde{U}_{\infty,k}\partial_{x_k}N_i\right)\partial_{x_l}\phi\left(-J^{-1}_{\mathrm{e},lk}\partial_{x_j}J_{\mathrm{e},kl}\right)\partial_{x_l}\phi\,dS_{\mathrm{e}}\\ &+\int_{\Gamma_{\mathrm{TE,u_e}}}\frac{1}{h^2}\left(N_{i_{\mathrm{TE,u}}} + \frac{h}{2}\tilde{U}_{\infty,k}\partial_{x_k}N_i\right)\partial_{x_k}\phi\partial_{x_k}\phi\,\partial_{x_j}dS_{\mathrm{e}}\\ &+\int_{\Gamma_{\mathrm{TE,l_e}}}\frac{2}{h^3}\partial_{x_j}h\left(N_{i_{\mathrm{TE,u}}} + \frac{h}{2}\tilde{U}_{\infty,k}\partial_{x_k}N_i\right)\partial_{x_k}\phi\partial_{x_k}\phi\,dS_{\mathrm{e}}\\ &-\int_{\Gamma_{\mathrm{TE,l_e}}}\frac{1}{h^2}\partial_{x_j}\left(\frac{h}{2}\tilde{U}_{\infty,k}\partial_{x_k}N_i\right)\partial_{x_k}\phi\partial_{x_k}\phi\,dS_{\mathrm{e}}\\ &-\int_{\Gamma_{\mathrm{TE,l_e}}}\frac{2}{h^2}\left(N_{i_{\mathrm{TE,u}}} + \frac{h}{2}\tilde{U}_{\infty,k}\partial_{x_k}N_i\right)\partial_{x_l}\phi\left(-J^{-1}_{\mathrm{e},lk}\partial_{x_j}J_{\mathrm{e},kl}\right)\partial_{x_l}\phi\,dS_{\mathrm{e}}\\ &-\int_{\Gamma_{\mathrm{TE,l_e}}}\frac{1}{h^2}\left(N_{i_{\mathrm{TE,u}}} + \frac{h}{2}\tilde{U}_{\infty,k}\partial_{x_k}N_i\right)\partial_{x_k}\phi\partial_{x_k}\phi\,\partial_{x_j}dS_{\mathrm{e}}.\end{aligned} \tag{2.12}$$

The partial gradient of the stabilization term in equations (2.11) and (2.12) can further be developed as

$$\partial_{x_j}\left(\frac{h}{2}\tilde{U}_{\infty,k}\partial_{x_k}N_i\right) = \partial_{x_j}\frac{h}{2}\tilde{U}_{\infty,k}\partial_{x_k}N_i + \frac{h}{2}\tilde{U}_{\infty,l}\phi J^{-1}_{\mathrm{e},lk}\partial_{x_j}J_{\mathrm{e},kl}\partial_{x_l}N_i. \tag{2.13}$$

The angle of attack affects the potential residuals only through the farfield boundary condition. The partial gradient of the potential residuals with respect to the angle of attack is thus given by

$$\frac{\partial R_{\phi,i}}{\partial\alpha} = \sum_{\mathrm{e}}\int_{\Gamma_{\mathrm{fe}}}\rho_\infty\frac{\partial\mathbf{U}_\infty}{\partial\alpha}\cdot\hat{\mathbf{n}}_{\mathrm{e}}N_i\,dS_{\mathrm{e}} \tag{2.14}$$

where the gradient of the freestream velocity with respect to the angle of attack is

$$\frac{\partial \mathbf{U}_\infty}{\partial \alpha} = \begin{bmatrix} -\sin\alpha\cos\beta \\ 0 \\ \cos\alpha\cos\beta \end{bmatrix}. \tag{2.15}$$

The partial gradient of the transpiration boundary condition with respect to the blowing velocity is required when viscous-inviscid interaction is enabled. It is given by

$$\frac{\partial R_{\phi,i}}{\partial \rho U_{\mathrm{bl},j}}|_{\mathrm{b}\cup\mathrm{w}} = \int_{\Gamma_{\mathrm{b_e}}\cup\Gamma_{\mathrm{w_e}}} \delta_{ijk} N_k \, dS_\mathrm{e}, \tag{2.16}$$

where the Kronecker's delta δ_{ijk} is one if $i = j = k$ and zero otherwise.

2.1.2 Functional

The partial gradient of the aerodynamic loads with respect to the potential variables is given by,

$$\begin{aligned} \frac{\partial F_i}{\partial \phi_j} &= \frac{1}{2}\rho_\infty u_\infty^2 \int_{\Gamma_{\mathrm{b,e}}} \partial_{\phi_j} C_{p_\mathrm{e}} \hat{n}_{\mathrm{e},i} \, dS_\mathrm{e} \\ &= -\rho_\infty u_\infty^2 \int_{\Gamma_{\mathrm{b,e}}} \rho_\mathrm{e}^\gamma \partial_{x_k}\phi \partial_{x_k} N_j \hat{n}_{\mathrm{e},i} \, dS_\mathrm{e}. \end{aligned} \tag{2.17}$$

The partial gradient of the aerodynamic loads with respect to the mesh coordinates is given by,

$$\begin{aligned} \frac{\partial F_i}{\partial x_j} &= \frac{1}{2}\rho_\infty u_\infty^2 \partial_{x_j} \int_{\Gamma_{\mathrm{b,e}}} C_{p_\mathrm{e}} \hat{n}_{\mathrm{e},i} \, dS_\mathrm{e} \\ &= \frac{1}{2}\rho_\infty u_\infty^2 \Bigg[\int_{\Gamma_{\mathrm{b,e}}} -2\rho_\mathrm{e}^\gamma \partial_{x_l}\phi \left(-J_{\mathrm{e},lk}^{-1} \partial_{x_j} J_{\mathrm{e},kl}\right) \partial_{x_l}\phi \hat{n}_{\mathrm{e},i} \, dS_\mathrm{e} \\ &\qquad + \int_{\Gamma_{\mathrm{b,e}}} C_{p_\mathrm{e}} \partial_{x_j} \hat{n}_{\mathrm{e},i} \, dS_\mathrm{e} \\ &\qquad + \int_{\Gamma_{\mathrm{b,e}}} C_{p_\mathrm{e}} \hat{n}_{\mathrm{e},i} \, \partial_{x_j} dS_\mathrm{e} \Bigg]. \end{aligned} \tag{2.18}$$

where the partial gradient of an elementary surface is computed as in equation (2.10), and the partial gradient of the unit normal vector is given by

$$\frac{\partial \hat{n}_i}{\partial x_j} = (I_{ik} - \hat{n}_i \hat{n}_k) \frac{1}{|n_k|} \partial_{x_j} n_k \tag{2.19}$$

where the partial gradient of the normal vector to a two-dimensional triangular area in a three-dimensional space is given by,

$$\frac{\partial \mathbf{n}}{\partial x_j} = \partial_{x_j} (\mathbf{x}_1 - \mathbf{x}_0) \times (\mathbf{x}_2 - \mathbf{x}_0) + (\mathbf{x}_1 - \mathbf{x}_0) \times \partial_{x_j} (\mathbf{x}_2 - \mathbf{x}_0). \tag{2.20}$$

The partial gradients of the aerodynamic load coefficients can be readily obtained from equations (2.17) and (2.18). Additionally, the partial gradients of the aerodynamic coefficients with respect to the angle of attack is given by,

$$\frac{C_L}{\partial\alpha} = \mathbf{C_F} \cdot \frac{\partial \mathbf{e_L}}{\partial\alpha}, \qquad \frac{C_D}{\partial\alpha} = \mathbf{C_F} \cdot \frac{\partial \mathbf{e_D}}{\partial\alpha}, \tag{2.21}$$

where the gradients of the directions are defined as,

$$\frac{\partial \mathbf{e_L}}{\partial\alpha} = \begin{bmatrix} -\cos\alpha \\ 0 \\ -\sin\alpha \end{bmatrix}, \qquad \frac{\partial \mathbf{e_D}}{\partial\alpha} = \begin{bmatrix} -\sin\alpha\cos\beta \\ 0 \\ \cos\alpha\cos\beta \end{bmatrix}. \tag{2.22}$$

2.2 Mesh deformation

This section details the formulation of the partial gradients of mesh morphing equations.

2.2.1 Residuals

The mesh deformation residuals only depend linearly on the mesh coordinates. The partial gradients of the mesh deformation residuals with respect to the mesh coordinates, also known as the mesh Jacobian, is therefore given by,

$$\begin{aligned} \frac{\partial R_{\mathrm{x},i}}{\partial x_j} &= J_{\mathrm{x},ij} \\ &= \sum_{\mathrm{e}} \int_{\Omega_\mathrm{e}} \left[\frac{E_\mathrm{e}\nu_\mathrm{e}}{2(1+\nu_\mathrm{e})(1-2\nu_\mathrm{e})} \partial_{x_k} N_l \delta_{ij} + \frac{E_\mathrm{e}}{2(1+\nu_\mathrm{e})} \left(\partial_{x_j} N_l + \partial_{x_i} N_l \right) \right] \partial_{x_j} N_l \, dV_\mathrm{e}. \end{aligned} \tag{2.23}$$

Note that, similar to the residuals $\mathbf{R}_\mathrm{x}$, periodic boundary conditions are prescribed on the wake by adding the upper wake volume element contributions to the lower wake equations, and by prescribing the upper wake unknowns to match the lower wake unknowns.

3 Solution procedures

This section presents the direct and adjoint solution procedures that are readily available in DART. If multiphysics computations are to be performed, the interfaces for CUPyDO [7, 8] [3] and MPhys [4], built on top of OpenMDAO [9] [5], can be used.

3.1 Direct solution

The full potential equation being nonlinear, it needs to be solved in an iterative fashion. A Taylor expansion around a solution vector $\boldsymbol{\phi}_\mathrm{s}$ allows to write

$$0 = \mathbf{R}_\phi + \frac{\partial \mathbf{R}_\phi}{\partial \boldsymbol{\phi}} \boldsymbol{\Delta\phi} + \mathcal{O}(\boldsymbol{\Delta\phi}^2), \tag{3.1}$$

where $\boldsymbol{\Delta\phi} = \boldsymbol{\phi} - \boldsymbol{\phi}_\mathrm{s}$. Neglecting second order terms, and given a known solution estimate $\boldsymbol{\phi}_n$ at iteration n, a better estimate of the solution, $\boldsymbol{\phi}_{n+1}$, can be found by solving

$$\frac{\partial \mathbf{R}_\phi}{\partial \boldsymbol{\phi}}|_{\boldsymbol{\phi}_n} (\boldsymbol{\phi}_{n+1} - \boldsymbol{\phi}_n) = -\mathbf{R}_\phi|_{\boldsymbol{\phi}_n}. \tag{3.2}$$

The Newton-Raphson method exhibits a second-order convergence rate as it gets closer to the solution. However, it might be unstable for transonic flow computations where the local Mach numbers are high. An effective way to stabilize the Newton method is to restrict the change in the solution using a line search procedure. In such a technique, the new solution vector is computed as

$$\boldsymbol{\phi}_{n+1} = \boldsymbol{\phi}_n + s_n(\boldsymbol{\phi}_{n+1} - \boldsymbol{\phi}_n), \tag{3.3}$$

where s_n is the step length of the line search. The Bank and Rose [10] algorithm has been implemented to find the optimal step length for a given iteration and is depicted in figure 3.1.

[3] http://github.com/ulgltas/cupydo, accessed November 2024.
[4] https://github.com/OpenMDAO/mphys, accessed November 2024.
[5] https://openmdao.org/, accessed November 2024.

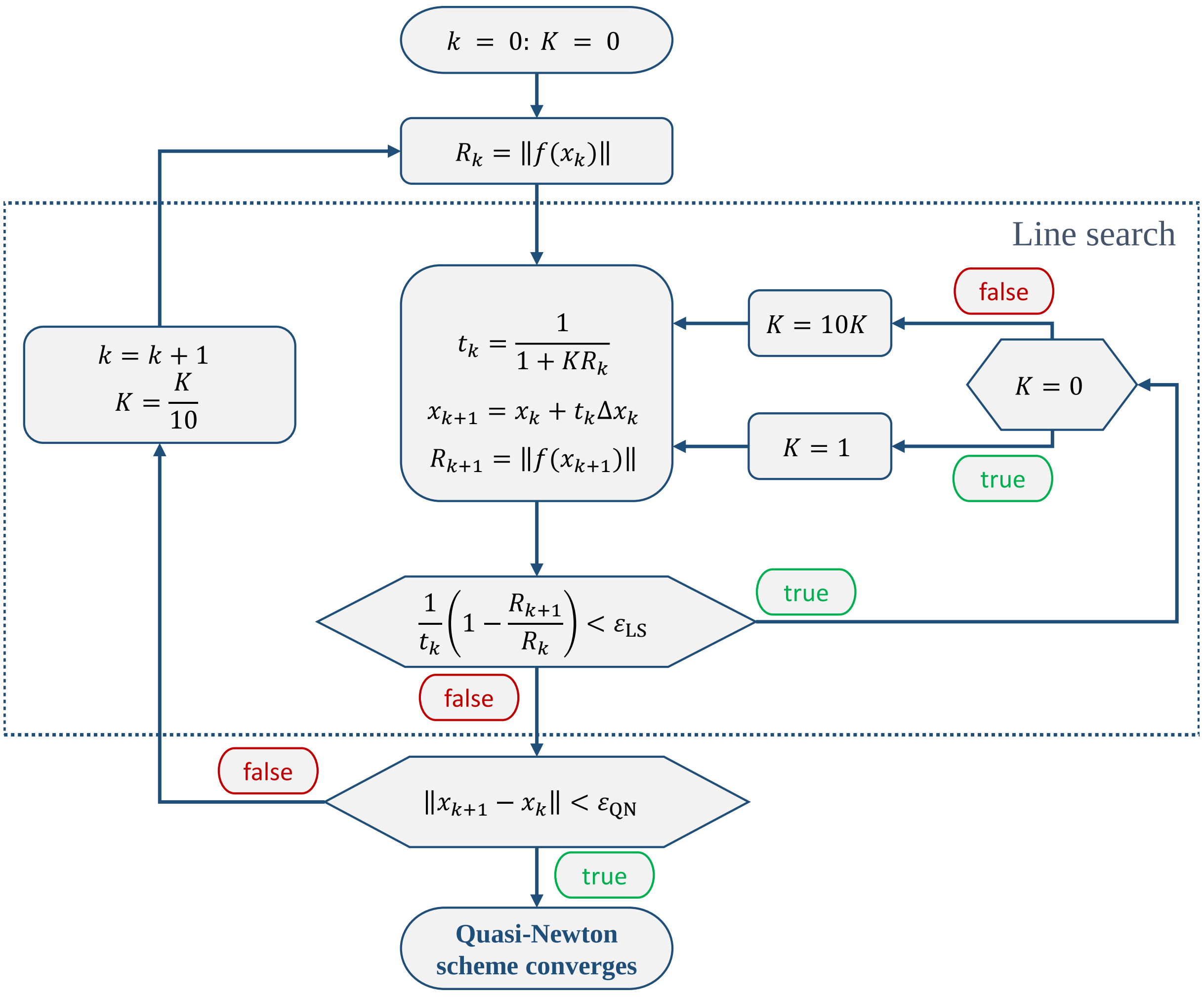

$k = 0$: $K = 0$
$R_k = \|f(x_k)\|$
Line search
$t_k = \frac{1}{1+KR_k}$
$x_{k+1} = x_k + t_k \Delta x_k$
$R_{k+1} = \|f(x_{k+1})\|$
$K = 10K$
$K = 1$
$K = 0$
false
true
$k = k+1$
$K = \frac{K}{10}$
$\frac{1}{t_k}\left(1 - \frac{R_{k+1}}{R_k}\right) < \varepsilon_{LS}$
true
false
$\|x_{k+1} - x_k\| < \varepsilon_{QN}$
false
true
Quasi-Newton scheme converges

Figure 3.1: Bank and Rose line search algorithm wrapped in a quasi-Newton method.

3.2 Adjoint solution

An adjoint method has also been implemented to compute the total gradients of the lift and the drag with respect to the angle of attack and the surface grid coordinates. These gradients can then be used in an optimization process. For pure aerodynamic optimization, the problem can be formulated as follows,

$$\begin{aligned} &\min_{\mathbf{x},\alpha} F_{\text{obj}}(\boldsymbol{\phi}, \mathbf{x}, \alpha) \\ &\text{s.t. } \mathbf{R}_\phi = 0 \\ &\quad\;\; \mathbf{R}_\text{x} = 0 \end{aligned} \tag{3.4}$$

where ϕ denote the vector of aerodynamic potential variables, α is the angle of attack, $\mathbf{R}_\phi$ represents the full potential equation noted in residual form, and F_{obj} is the functional (lift or drag) to be minimized. Since a nonlinear aerodynamic model is used, the full potential equations must be solved in the volume surrounding the geometry, which will deform according to follow the movement of the surface grid. In such a case, it is also convenient to explicitly introduce the vector of volume mesh coordinates, $\mathbf{x}$, and the mesh morphing laws residuals, $\mathbf{R}_\text{x}$, into the optimization formulation.

In order to minimize F_{obj}, the augmented Lagrangian $\mathcal{L}$ is first constructed as,

$$\mathcal{L} = F_{\text{obj}} + \boldsymbol{\lambda}_\phi \mathbf{R}_\phi + \boldsymbol{\lambda}_{\text{x}} \mathbf{R}_{\text{x}} \tag{3.5}$$

and then differentiated such that,

$$\delta\mathcal{L} = 0 \Rightarrow \begin{cases} \frac{\partial F_{\text{obj}}}{\partial \boldsymbol{\phi}} + \boldsymbol{\lambda}_\phi \frac{\partial \mathbf{R}_\phi}{\partial \boldsymbol{\phi}} + \boldsymbol{\lambda}_{\text{x}} \frac{\partial \mathbf{R}_{\text{x}}}{\partial \boldsymbol{\phi}} = 0 \\ \frac{\partial F_{\text{obj}}}{\partial \mathbf{x}} + \boldsymbol{\lambda}_\phi \frac{\partial \mathbf{R}_\phi}{\partial \mathbf{x}} + \boldsymbol{\lambda}_{\text{x}} \frac{\partial \mathbf{R}_{\text{x}}}{\partial \mathbf{x}} = 0 \\ \frac{\partial F_{\text{obj}}}{\partial \alpha} + \boldsymbol{\lambda}_\phi \frac{\partial \mathbf{R}_\phi}{\partial \alpha} + \boldsymbol{\lambda}_{\text{x}} \frac{\partial \mathbf{R}_{\text{x}}}{\partial \alpha} = 0 \\ \mathbf{R}_\phi = 0 \\ \mathbf{R}_{\text{x}} = 0 \end{cases} . \tag{3.6}$$

In order to obtain the total gradients of F_{obj}, the nonlinear potential and linear mesh equations, $\mathbf{R}_\phi = 0$ and $\mathbf{R}_{\text{x}} = 0$ must first be solved. The coupled set of linear adjoint equations,

$$\begin{bmatrix} \partial_{\boldsymbol{\phi}} \mathbf{R}_\phi{}^{\text{T}} & \partial_{\boldsymbol{\phi}} \mathbf{R}_{\text{x}}{}^{\text{T}} \\ \partial_{\mathbf{x}} \mathbf{R}_{\boldsymbol{x}}{}^{\text{T}} & \partial_{\mathbf{x}} \mathbf{R}_{\text{x}}{}^{\text{T}} \end{bmatrix} \begin{bmatrix} \boldsymbol{\lambda}_\phi \\ \boldsymbol{\lambda}_{\text{x}} \end{bmatrix} = - \begin{bmatrix} \partial_{\boldsymbol{\phi}} F_{\text{obj}}^{\text{T}} \\ \partial_{\mathbf{x}} F_{\text{obj}}^{\text{T}} \end{bmatrix} \tag{3.7}$$

must then be solved for the Lagrange multipliers $\boldsymbol{\lambda}_\phi$ and $\boldsymbol{\lambda}_{\text{x}}$. The total gradient with respect to the surface mesh coordinates can readily be recovered from λ_{x}, as they are a subset of this vector [11]. The total gradient with respect to the angle of attack can finally be obtained by injecting the solution into

$$\frac{dF_{\text{obj}}}{d\alpha} = \frac{\partial F_{\text{obj}}}{\partial \alpha}^{\text{T}} - \frac{\partial \mathbf{R}_\phi}{\partial \alpha}^{\text{T}} \boldsymbol{\lambda}_\phi . \tag{3.8}$$

4 Quick reference guide

This section lists the various Application Programming Interface (API) available in DART, as well as the parameters required to configure them. The full documentation is available at: https://gitlab.uliege.be/am-dept/dartflo/-/wikis/home, accessed November 2024.

4.1 Available API

The main API used to initialize and access the components making up DART is the so-called *core* API [6]. Three other API, built on top of the *core*, are also available: *internal*, *CUPyDO* and *MPhys* API. The *internal* [7] API is meant for users wanting to run a standard computational procedure, such as computing a polar curve, on a classical lifting configuration. The *CUPyDO* [8] and *MPhys* [9] API are direct interfaces to their respective software.

[6] https://gitlab.uliege.be/am-dept/dartflo/-/wikis/use_api_core, accessed November 2024.

[7] https://gitlab.uliege.be/am-dept/dartflo/-/wikis/use_api_internal, accessed November 2024.

[8] https://gitlab.uliege.be/am-dept/dartflo/-/wikis/use_api_cupydo, accessed November 2024.

[9] https://gitlab.uliege.be/am-dept/dartflo/-/wikis/use_api_mphys, accessed November 2024.

4.2 Usage

The list of parameters used to configure DART is provided below:

```
cfg = {
  # Options
  'Threads': int, # number of threads (optional, default=1)
  'Verb': int, # verbosity (optional, default=1)
  # Model (geometry or mesh)
  'File': str, # Input file containing the model
  'Pars': dict, # parameters for input file model
  'Dim': int, # problem dimension (2 or 3)
  'Format': str, # save format (vtk or gmsh)
  # Markers...
  'Fluid': str, # name of physical group containing the fluid
  'Farfield': list[str], # LIST of names of physical groups containing the farfield boundaries
      (downstream should be last element)
  # ... only 2D
  'Wing': str, # name of physical group containing the airfoil boundary (will be the body of
      interest for aerostructural and optimization)
  'Wake': str, # name of physical group containing the wake
  'Te': str, # name of physical group containing the trailing edge
  # ... only 3D
  'Wings': list[str], # LIST of names of physical groups containing the lifting surface
      boundary (first element will be the body of interest for aerostructural and optimization)
  'Wakes': list[str], # LIST of names of physical group containing the wake
  'WakeTips': list[str], # LIST of names of physical group containing the free edge of the
      wake (not for 2.5D)
  'Tes': list[str], # LIST of names of physical group containing the trailing edges
  # ... optional for 3D
  'Symmetry': str, # name of physical group containing the symmetry boundaries
  'Fuselage': str, # name of physical group containing the fuselage boundary
  'WakeExs': str, # LIST of names of physical group containing the free edge of the wake and
      the intersection of lifting surface with fuselage (to be excluded from Wake B.C.), only
      required if a 'Fuselage' if present, otherwise 'WakeTips' is sufficient
  # Freestream
  'M_inf': float, # freestream Mach number (optional, default=0)
  'AoA': float, # freestream angle of attack [deg] (optional, default=0)
  'AoS': float, # freestream angle of sideslip [deg] (optional, default=0)
  'Q_inf': float, # freesteam dynamic pressure (only required for aerostructural computations)
  # Geometry
  'S_ref': float, # reference surface length
  'c_ref': float, # reference chord length
  'x_ref': float, # x-coordinate of reference point for moment computation
  'y_ref': float, # y-coordinate of reference point for moment computation
  'z_ref': float, # z-coordinate of reference point for moment computation
  # Numerical
  'LSolver': str, # inner solver (PARDISO, MUMPS, SparseLU or GMRES)
  'G_fill': int, # fill-in factor for GMRES preconditioner (optional, default=2)
  'G_tol': float, # tolerance for GMRES (optional, default=1e-5)
  'G_restart': int, # restart for GMRES (optional, default=50)
  'Rel_tol': float, # relative tolerance on solver residual
  'Abs_tol': float, # absolute tolerance on solver residual
  'Max_it': int # maximum number of iterations for nonlinear solver
}
```

The *core* API can then be initialized using:

```python
from dart.api.core import initDart
_dart = initDart(cfg, scenario='aerodynamic', task='analysis', viscous=False)
```

where `scenario` can be `aerodynamic` or `aerostructural`, and `task` can be `analysis` or `optimization`, and `viscous` is a boolean indicating whether the solver should also be configured for viscous-inviscid interaction. `_dart` is a python dictionary containing the following objects (named after their key):

- `dim` is the number of dimensions (2 or 3),
- `qinf` is the freestream dynamic pressure (0 except if scenario='aerostructural'),
- `msh` is the mesh,
- `wrt` is the utility to write mesh/results on disk,
- `mrf` is the mesh morpher (None except if scenario='aerostructural' or task='optimization'),
- `pbl` is the formulation of the problem,
- `bnd` is the body of interest,
- `blwb` is the blowing boundary condition on the body (None except if viscous=True),
- `blww` is the blowing boundary condition on the wake (None except if viscous=True),
- `sol` is the direct (Newton) solver,
- `adj` is the adjoint solver (None except if task='optimization').

In order to use the other API, please refer to the main documentation.

References

[1] Adrien Crovato. *Steady Transonic Aerodynamic and Aeroelastic Modeling for Preliminary Aircraft Design*. PhD thesis, University of Liège, October 2020.

[2] Adrien Crovato, Alex P. Prado, Pedro H. Cabral, Romain Boman, Vincent E. Terrapon, and Grigorios Dimitriadis. An adjoint full potential solver for fast aerostructural optimization in preliminary aircraft design. *Aerospace Science and Technology*, 2023.

[3] Joseph L. Steger and Barrett S. Baldwin. Shock waves and drag in the numerical calculation of isentropic transonic flows. Technical report, NASA, 1972.

[4] Wadgi G. Habashi and Mohamed M. Hafez. Finite Element Solutions of Transonic Flow Problems. *AIAA Journal*, 20(10):1368–1376, 1982.

[5] Paul Dechamps, Amaury Bilocq, Adrien Crovato, Grigorios Dimitriadis, and Vincent Terrapon. Pseudo unsteady quasi-simultaneuous integral boundary layer methodology for preliminary aircraft design. *In preparation for submission to Journal of Aircraft*, 2024.

[6] Richard P. Dwight. Robust Mesh Deformation using the Linear Elasticity Equations. *Journal of Computational Fluid Dynamics*, 12:401–406, 2009.

[7] David Thomas, Marco-Lucio Cerquaglia, Romain Boman, Thomas Economon, Juan Alonso, Grigorios Dimitriadis, and Vincent E. Terrapon. CUPyDO: An integrated Python environment for coupled fluid-structure problems. *Advances in Engineering Software*, 2019.

[8] Marco-Lucio Cerquaglia, David Thomas, Romain Boman, Vincent E. Terrapon, and Jean-Phillipe Ponthot. A fully partitioned Lagrangian framework for FSI problems characterized by free surfaces, large solid deformations and displacements, and strong added-mass effects. *Computer Methods in Applied Mechanics and Engineering*, 2019.

[9] Justin S. Gray, John T. Hwang, Joaquim R. R. A. Martins, Kenneth T. Moore, and Bret A. Naylor. OpenMDAO: An open-source framework for multidisciplinary design, analysis, and optimization. *Structural and Multidisciplinary Optimization*, 59(4):1075–1104, April 2019.

[10] Randolph E. Bank and Donald J. Rose. Global Approximate Newton Method. *Numerische Mathematik*, 27:179–295, 1981.

[11] Markus Widhalm, Joël Brezillon, Caslav Ilic, and Tobias Leicht. Investigation on Adjoint Based Gradient Computations for Realistic 3d Aero-Optimization. In *13th AIAA/ISSMO Multidisciplinary Analysis Optimization Conference*. AIAA, September 2010.

[12] Anil Yildirim, K.E. Jacobson, Anibal J.L., Stanford B.K., Justin S. Gray, Mader C.A., Joaquim R. R. A. Martins, and Graeme J. Kennedy. MPhys: A Modular Multiphysics Library for Coupled Simulation and Adjoint Derivative Computation. *Structural and Multidisciplinary Optimization*, page In press, 2024.