

Flow simulation for fluid–structure interaction in the vibration problem of a travelling panel

Juha Jeronen

January 19, 2016

TECHNICAL NOTE

1 Introduction

- This research is related to paper making, specifically, open draws in paper machines. Important fundamental physics for moving materials: questions of stability.
- Previously investigated: small transverse vibrations of a travelling panel subjected to an axial potential flow, in 2D [refs here]
- Current aim: small transverse vibrations of a travelling panel subjected to an axial Euler flow, in 2D
- Why Euler? Air has small viscosity, may be able to ignore it. Also, Euler flow is the next logical step in systematic model building for this system. Comparison with earlier (potential flow) results will isolate the effect that the advective nonlinearity of the fluid component has on the vibration characteristics of the structure component.
- Euler flow = inviscid, but (in the general case) not irrotational; may be compressible or incompressible. Compare potential flow = inviscid, irrotational, incompressible.
- Advection-dominated problem, nonlinear first-order PDEs.
- Generally speaking, meshless methods are promising for fluid–structure interaction (FSI) problems due to being free from restrictions on mesh topology. The points may easily move in an arbitrary manner without the need for remeshing.
- A Lagrangean (material) formulation is very tempting, because it gets rid of one of the principal difficulties in fluid flow equations: the advection term. However, even with meshless methods, Lagrangean approaches need periodic resetting due to issues in point cloud density. As is known from MAC (marker-and-cell) methods in computer graphics (used e.g. in realtime treatment of free-surface flows), the point cloud tends to form clusters and voids as the points move along the flow.
- Thus, to keep things simple, an Eulerian formulation will be considered, using a stationary set of points (which may be arbitrarily distributed). This is sufficient for small vibrations of structures, where the geometry can be approximated as flat. This approach is especially applicable to stability analysis, where the local amplification characteristics of small disturbances are the main object of interest.
- In this approach, the vibration of the structure is seen by the flow only via a velocity boundary condition (the no-penetration condition) along the geometrically flat boundary. The non-flatness of the actual surface is accounted for when determining the local orientation of the surface for the boundary condition. The same approach was used in our earlier studies; thus the results will be directly comparable.
- Because Euler flow is inviscid (just like potential flow), it cannot push things (that would be a viscous phenomenon)! It may only cause structures to be displaced by differences in dynamic pressure (e.g. as in the classical explanation of how wings work). The flow is seen by the structure only as a local difference in pressure across the upper/lower surfaces.

- Numerical method: for simplicity, we choose the classical moving least squares meshless method (MLS).
- MLS is a local Taylor series optimization scheme to explicitly compute spatial derivatives (usually, up to 2nd order) of fields defined by their values on a set of points. Scalar, vector and tensor fields of any rank can be treated by MLS.
- MLS belongs to the family of collocation methods (sometimes called finite point methods). MLS is similar in spirit to finite differences, but generalized to arbitrary geometries. MLS is just one approach to perform this generalization, two others being interpolation functions on a background mesh (so that classical difference stencils can be used as-is), and developing new difference formulas to account for non-equal distances in the stencil.
- MLS has the advantage that it guarantees the least-squares optimal (i.e. in a sense the best possible) local Taylor series representation of the unknown function. In the category of optimization-based methods, MLS is also reasonably efficient; taking advantage of the quadratic form of the optimization problem, it is reduced to solving a linear equation system. For different points in the point cloud, these systems are local and independent, enabling parallelization.
- If MLS is used, an explicit approach is preferable for time integration, because MLS does not easily lead itself to implicit equations, or to steady-state problems such as elliptic equations.
- In incompressible flow, the pressure term acts as an instantaneous correction to keep the velocity field divergence-free [refs here]. The often used predictor-corrector approach requires solving a Poisson equation for the pressure. This equation appears via the Helmholtz projection trick, where (formally) the pressure field is chosen at each timestep such that the pressure gradient eliminates the divergence of the predicted velocity field. (This projects the predicted velocity field into divergence-free space.)
- Hence, for MLS, compressible flow is preferable, because fully explicit formulations are possible.
- Our problem has only Dirichlet boundary conditions. The velocity has a given inflow, and at the surfaces of obstacles, its normal component is zero. All the unknowns are transported by the velocity field, and there is no diffusion in any of the equations. Hence, considering information flow, if the velocity field does not penetrate obstacles, these other fields will not either. The scalar quantities, density and internal energy, have only given values at inflow.
- Of course, when computing derivatives, the point clouds (used in the Taylor series optimization) must be chosen such that the surface is treated as an obstacle, so that the discontinuities are preserved correctly. For example, points in the flow simulation near the upper surface of the panel must use only points from the upper side; the points below the lower surface of the panel are not visible to them.
- To treat the obstacle boundary conditions, we write the equations in the local (tangent, normal) coordinates, solve the tangential component, set the normal component to zero, and then transform the result to global (x, y) coordinates by rotating the coordinate system. This requires only minor modifications to the solution procedure. This is more robust and arguably simpler than solving for the x (or y) component as a function of the other component.
- To specify the link between pressure and density, closing the equation system, we treat the air as an ideal gas.

2 Governing equations

For obtaining local balance laws as partial differential equations in continuum form, it is convenient to start from elementary physical considerations in integral form over a control volume, apply the Reynolds transport theorem, and finally extract the pointwise behavior by noting the arbitrariness of the control volume. For such a general treatment of conservation laws, see e.g. [ref AllenHerreraPinder, CSCVirtauslaskenta]. The final results, which provide our starting point, are as follows; we will give them with some comments relevant to our present study.

In the Eulerian (laboratory) frame, local dynamic mass balance is given by

$$\frac{\partial \rho}{\partial t} + \nabla \cdot (\rho \mathbf{u}) = 0. \quad (1)$$

Here ρ (SI unit: kg/m^3) is the density of the fluid and \mathbf{u} (m/s) is the velocity field. The divergence in (1) can be expanded, yielding

$$\frac{\partial \rho}{\partial t} + \mathbf{u} \cdot \nabla \rho + \rho \nabla \cdot \mathbf{u} = 0. \quad (2)$$

Moving one of the terms yields a form that lends itself to an intuitive physical interpretation of (1):

$$\frac{\partial \rho}{\partial t} + \mathbf{u} \cdot \nabla \rho = -\rho \nabla \cdot \mathbf{u}.$$

This is a scalar first-order transport equation with a source term. The density field ρ undergoes advection by the velocity field \mathbf{u} , and in addition experiences a source of strength $-\rho \nabla \cdot \mathbf{u}$ (i.e. a sink of strength $\rho \nabla \cdot \mathbf{u}$). That is, at points where $\nabla \cdot \mathbf{u} > 0$, i.e. where fluid flows outward from the point, the density decreases proportional to the existing density (and similarly, increases in the case of inflow toward the point).

The equation commonly known as *Euler's equation for fluid flow* describes the dynamic balance of linear momentum in an Euler (i.e. general inviscid) flow, written in an Eulerian (laboratory) frame:

$$\frac{\partial \mathbf{u}}{\partial t} + \mathbf{u} \cdot \nabla \mathbf{u} + \frac{\nabla p}{\rho} = \mathbf{g}. \quad (3)$$

We have written (3) in terms of accelerations (m/s^2) in the standard manner.¹ The new quantities that have appeared in (3) are the pressure p (Pa) and the acceleration of body forces \mathbf{g} (m/s^2). The latter can be used to represent e.g. gravity.

For the gradient, in this document we use the “transpose Jacobian” convention

$$(\nabla \mathbf{a})_{ij} := \partial_i a_j, \quad (4)$$

which is based on the natural reading order of the index notation. As the name suggests, this is the transpose of the standard Jacobian

$$(J(\mathbf{a}))_{ij} := \frac{\partial a_i}{\partial x_j} \equiv \partial_j a_i, \quad (5)$$

where the ordering is based on the natural reading order of the classical partial derivative notation.

We have chosen to use the transpose Jacobian convention (4) for convenience, as it makes some things simpler. In a Cartesian (i.e. orthonormal) tensor setting, it allows manipulating the symbol ∇ using the exact same rules as vectors, simplifying manipulation of equations and thus reducing the chance of mistakes. The directional derivative operator is written as $\mathbf{a} \cdot \nabla$, as usual, but now the notation can be interpreted literally, $\mathbf{a} \cdot \nabla = a_i \partial_i$. For directional derivatives of vector fields, we just translate the symbols literally from left to right, $(\mathbf{a} \cdot \nabla \mathbf{b})_j = a_i \partial_i b_j$. The direction vector naturally belongs to the left, avoiding any special-case rules for nabla.²

Equation (3) follows from the general dynamic balance equation of linear momentum, which is a fundamental law valid for both fluids and solids:

$$\rho \left(\frac{\partial \mathbf{u}}{\partial t} + \mathbf{u} \cdot \nabla \mathbf{u} \right) - \nabla \cdot \boldsymbol{\sigma}^T = \mathbf{f}, \quad (6)$$

where $\boldsymbol{\sigma}$ is the stress tensor and $(\cdot)^T$ denotes the rank-2 transpose, $(A^T)_{ij} = A_{ji}$. We define $\mathbf{g} = \mathbf{f}/\rho$, and choose $\boldsymbol{\sigma}$ as the stress tensor of an inviscid fluid,

$$\sigma_{ij} = -\delta_{ij} p, \quad (7)$$

where δ_{ij} is the Kronecker delta. An inviscid fluid only resists compression isotropically; there is no shear resistance. Note that choosing “inviscid fluid” in (6) is a constitutive modeling assumption, partially specifying the constitutive model for the continuum being modelled. Then

$$(\nabla \cdot \boldsymbol{\sigma}^T)_j = \partial_i \sigma_{ji} = -\partial_i (\delta_{ji} p) = -\partial_j p = -(\nabla p)_j, \quad (8)$$

yielding (3).

¹For an alternative viewpoint emphasizing the momentum density $\rho \mathbf{u}$, see <https://www.av8n.com/physics/euler-flow.pdf>

²Even when using the “standard Jacobian” convention, many authors write the directional derivative of a vector field as $\mathbf{a} \cdot \nabla \mathbf{b}$, but one must then keep in mind to interpret it as special notation for $(\nabla \mathbf{b}) \cdot \mathbf{a} = (\partial_j b_i) a_i$.

Dynamic balance of specific internal energy e (J/kg) requires

$$\frac{\partial e}{\partial t} + \mathbf{u} \cdot \nabla e + \frac{p}{\rho} \nabla \cdot \mathbf{u} = 0 . \quad (9)$$

This is also a scalar transport equation with a source (sink) term.

The balance of the fourth and final conserved quantity in mechanics, angular momentum, requires only that the stress tensor is symmetric.

At this point there are three equations and four unknowns: ρ , \mathbf{u} , p and e . In order to close the system of equations, the constitutive model must be completed. In the following subsections, we will consider alternative constitutive modeling assumptions that can be used for this.

2.1 Incompressible flow

One of the simplest ways³ to complete the constitutive model is to take a barotropic equation of state: $\rho = \rho(p)$. A special case of this is $\rho = \text{const.}$, reducing (1) to $\nabla \cdot \mathbf{u} = 0$, which states that the flow is incompressible. A typical case is any liquid.

If we instead begin by requiring incompressibility, $\nabla \cdot \mathbf{u} = 0$, equations (1) and (9) transform into

$$\frac{d\rho}{dt} = \frac{\partial \rho}{\partial t} + \mathbf{u} \cdot \nabla \rho = 0 , \quad (10)$$

$$\frac{de}{dt} = \frac{\partial e}{\partial t} + \mathbf{u} \cdot \nabla e = 0 , \quad (11)$$

where d/dt is the material derivative. That is, in any incompressible flow, the density of each fluid parcel remains constant in time, as does its internal energy. The initial distributions of ρ and e are simply transported by the flow. Physically, of course, in a single-phase flow of a liquid of a single type, it is reasonable to then take $\rho = \text{const.}$, although this is not mathematically required by just the property of incompressibility.

In a constant-density flow, only equation (3), with the constraint $\nabla \cdot \mathbf{u} = 0$, is needed for the flow simulation.

For an incompressible flow, no additional constitutive equation is needed to connect p and ρ . This is because in the incompressible flow model, the pressure term acts as a Lagrange multiplier for the incompressibility constraint. The pressure field instantaneously adjusts itself such that the velocity field remains divergence-free at all times. Note that because this happens instantaneously across the whole domain, the speed of sound in an incompressible fluid is infinite.

Remaining is the question of obtaining the pressure field in practice. Some approaches will be summarized below.

2.1.1 Explicit equation for pressure

An explicit equation for the pressure can be obtained by taking the divergence of the linear momentum balance (3), and then simplifying the result using the incompressibility constraint $\nabla \cdot \mathbf{u} = 0$. We have

$$\nabla \cdot \left(\frac{\partial \mathbf{u}}{\partial t} + \mathbf{u} \cdot \nabla \mathbf{u} + \frac{\nabla p}{\rho} \right) = \nabla \cdot \mathbf{g} .$$

This is easiest to work with by using Cartesian tensors in index notation. Starting from

$$\partial_i \left(\partial_t u_i + u_j \partial_j u_i + \frac{1}{\rho} \partial_i p \right) = \partial_i g_i ,$$

we first obtain

$$\partial_i (\partial_t u_i) + \partial_i (u_j \partial_j u_i) + \partial_i \left(\frac{1}{\rho} \partial_i p \right) = \partial_i g_i .$$

Expanding the second term, we have

$$\partial_i (\partial_t u_i) + (\partial_i u_j) (\partial_j u_i) + u_j (\partial_i \partial_j u_i) + \partial_i \left(\frac{1}{\rho} \partial_i p \right) = \partial_i g_i .$$

³See https://astro.uni-bonn.de/~jonathan/misc/hydro_notes.pdf

Assuming sufficient continuity so that we can reorder derivatives, we can rewrite this as

$$\partial_t(\partial_i u_i) + (\partial_i u_j)(\partial_j u_i) + u_j \partial_j(\partial_i u_i) + \partial_i \left(\frac{1}{\rho} \partial_i p \right) = \partial_i g_i .$$

Now using $\partial_i u_i = 0$ (incompressibility) and $\rho = \text{const.}$ yields

$$(\partial_i u_j)(\partial_j u_i) + \frac{1}{\rho} \partial_i \partial_i p = \partial_i g_i .$$

Converting back to nabla notation, the equation for pressure in a constant-density incompressible flow is

$$\nabla \mathbf{u} : \nabla \mathbf{u} + \frac{1}{\rho} \Delta p = \nabla \cdot \mathbf{g} , \quad (12)$$

where the double-dot product is $\mathbf{A} : \mathbf{B} = A_{ij} B_{ji}$ (note the ordering of the indices).

Some authors write the first term, equivalently, as

$$(\partial_i u_j)(\partial_j u_i) = \text{tr} \left((\nabla \mathbf{u})^2 \right) , \quad (13)$$

where $\text{tr}(\mathbf{A}) = A_{ii}$ is the trace and $\mathbf{A}^2 = \mathbf{A} \cdot \mathbf{A}$, i.e. $(\mathbf{A}^2)_{ij} = A_{ik} A_{kj}$.

Note that if \mathbf{g} is, for example, a uniform gravitational field, the right-hand side of (12) vanishes.

Solving the Poisson equation (12) at each timestep, the pressure field is obtained. As boundary conditions, known pressure (Dirichlet, essential) or the normal derivative of pressure (Neumann, natural) can be used.

2.1.2 Predictor-corrector methods

A classical way to treat incompressible flow numerically is to use predictor-corrector methods, which first evolve \mathbf{u} to the end of the timestep, ignoring the pressure term. Then, as the corrector step, the Helmholtz decomposition (or in general, Hodge decomposition) is utilized to project the predicted velocity field onto its divergence-free part⁴. This approach also leads to a Poisson equation for p .

The technique is based on the fundamental theorem of vector calculus, which states that any smooth vector field \mathbf{v} in three dimensions can be decomposed into the sum of divergence-free and curl-free parts:

$$\mathbf{v} = \mathbf{w} + \nabla \varphi , \quad (14)$$

where $\nabla \cdot \mathbf{w} = 0$ and φ is the scalar potential of the curl-free part (note $\nabla \times \nabla(\cdot) \equiv 0$). Rearranging, we immediately obtain the divergence-free projection of \mathbf{v} ,

$$\mathbf{w} = \mathbf{v} - \nabla \varphi , \quad (15)$$

and the only remaining question is how to compute the scalar potential φ .

Let $\tilde{\mathbf{u}}$ be the predicted velocity field, which may have nonzero divergence. Taking its divergence, using the decomposition (14) and denoting the scalar potential by p , we have

$$\nabla \cdot \tilde{\mathbf{u}} = \nabla \cdot (\mathbf{w} + \nabla p) = \nabla \cdot \mathbf{w} + \nabla \cdot \nabla p = \nabla \cdot \nabla p \equiv \Delta p . \quad (16)$$

Given $\tilde{\mathbf{u}}$, this is a Poisson equation for p . As the boundary conditions, one may specify either known pressure, or the normal derivative of pressure. The divergence-free velocity field is then obtained by (15).

In two dimensions, one may use the Hodge decomposition. Alternatively, for a simple approach, motivated by the above, we begin by attempting to set

$$\Delta p = \nabla \cdot \tilde{\mathbf{u}} \quad (17)$$

for some unknown scalar field p . Now, because the laplacian $\Delta(\cdot) \equiv \nabla \cdot \nabla(\cdot)$,

$$\nabla \cdot \nabla p = \nabla \cdot \tilde{\mathbf{u}} .$$

Rearranging gives

$$\nabla \cdot (\tilde{\mathbf{u}} - \nabla p) = 0 ,$$

i.e. the expression inside the parentheses is divergence-free. Choosing $\mathbf{u} := \tilde{\mathbf{u}} - \nabla p$, we have obtained a divergence-free velocity field (in any number of space dimensions). Just like above, given that $\tilde{\mathbf{u}}$ is known, (17) is a Poisson equation for p . The same note about boundary conditions applies.

⁴This procedure is also called Leray projection, see e.g. the course notes at <https://www.math.ucdavis.edu/~hunter/notes/euler.pdf>

2.1.3 Divergence-free function spaces

A modern, mathematically especially elegant, way to treat incompressible flow is to use a finite element discretization where the basis is able to directly represent divergence-free spaces. This allows looking for the solution in the correct space in the first place, avoiding the need for a separate correction step.⁵

2.2 Compressible flow of an ideal gas

As was noted in the introduction, the incompressible model is difficult to handle with MLS. To keep the numerical implementation as simple as possible, we will instead opt for compressible flow. This will be the topic of this subsection.

In a compressible flow, one simple constitutive model is to treat the flowing substance as an ideal gas. Since we are dealing with a gas (air) as opposed to a liquid, this is a reasonable approximation (although in the range of mach numbers below 0.1, the incompressible model is often used also for air).

For an ideal gas, we have the well-known equation of state,

$$pV = nRT, \quad (18)$$

where p (Pa) is the pressure, V (m³) is the volume being considered, n (mol) is the amount of ideal gas in the volume V , the quantity $R = 8.3144598$ J/mol K is the universal gas constant, and T (K) is the temperature. To bring (18) into a usable form, we may use the relation

$$n = \frac{m}{\mu}, \quad (19)$$

where m (kg) is the total mass of ideal gas in the volume V , and μ (kg/mol) is the mean molecular weight. For air⁶, $\mu = 28.97$ kg/kmol = $2.897 \cdot 10^{-2}$ kg/mol. This is accounting for a composition of approximately 78% N₂, 21% O₂, and 1% Ar (with negligible amounts of other constituents).

Inserting (19) into (18) and dividing both sides by $V \neq 0$, we have

$$p = \frac{m}{V} \frac{1}{\mu} RT. \quad (20)$$

Recognizing that equation (20) models the same substance as our balance equations, and taking the limit as the control volume $V \rightarrow 0^+$, we have $m/V \rightarrow \rho$ and thus for the continuum, we obtain the pointwise relation

$$p = \rho \frac{1}{\mu} RT. \quad (21)$$

We may stop at this point, and approximate the temperature as constant, e.g. $T = 293.15$ K. The constant-temperature assumption makes (21) a barotropic relation, $p = p(\rho)$. Moreover, (21) is linear in ρ , the constant of proportionality being

$$\frac{RT}{\mu} = \frac{8.3144598 \text{ J/mol K} \cdot 293.15 \text{ K}}{2.897 \cdot 10^{-2} \text{ kg/mol}} \approx 8.413 \cdot 10^4 \frac{\text{J}}{\text{kg}}. \quad (22)$$

In physical terms, the constant-temperature assumption should be valid if any compression and expansion that occurs in the gas can be considered small enough to cause no significant temperature changes. Then, because now e does not appear in (1), (3) or (21), if we are not interested in the internal energy, we may simply drop (9) and consider only (1) and (3). Thus we obtain a closed system of two equations in the two unknowns ρ and \mathbf{u} . The pressure p is given by (21).

Equations (1), (3) and (9) with (21) providing p allow the use of an explicit integration method, each equation for the unknown that appears in the $\partial/\partial t$ term. When the fields ρ , \mathbf{u} (and optionally e) are known at time $t = t_0$, an explicit timestep can be taken (using e.g. RK4) to obtain their new values at $t = t_0 + \Delta t$. (Of course, boundary conditions must also be enforced.)

For initializing the simulation, we can consider e.g. the ambient (far-field, free-stream) pressure p_∞ and temperature T_∞ given, and determine the corresponding ambient density ρ_∞ from (21).

⁵For this approach, see e.g.

<http://www.math.udel.edu/~szhang/research/p/uj.pdf>

<http://www.math.udel.edu/~szhang/research/p/review.pdf>

http://www.dam.brown.edu/people/jguzman/documents/Stokes2D_000.pdf.

⁶See http://www.engineeringtoolbox.com/molecular-mass-air-d_679.html

2.3 Compressible flow of an ideal gas with temperature variations

A slightly more sophisticated variant of the model can be used to account for temperature changes due to compression or expansion of the ideal gas. The total internal energy of an ideal gas (note that “ideal gas” implies no phase changes occur) is given by

$$U = c_V n T, \quad (23)$$

where U (J) is the total internal energy for mass m , i.e. $U/m = e$, and c_V (J/kg K) is the specific heat capacity (of the ideal gas) at constant volume. Dividing both sides of (23) by m and using (19), we obtain the relation between the internal energy and the temperature,

$$e = \frac{c_V}{\mu} T. \quad (24)$$

For air, the specific heat capacity itself depends on temperature; in the range 250...300 K, we can use the value⁷ $c_V \approx 0.717 \text{ kJ/kg K} = 7.17 \cdot 10^2 \text{ J/kg K}$. Solving equation (24) for T and inserting the result into (21) obtains

$$p = \rho \frac{1}{\mu} R \left[\frac{e\mu}{c_V} \right] = \rho \frac{R}{c_V} e, \quad (25)$$

which explicitly gives p in terms of the other unknowns ρ and e . Also from (21), we have

$$\rho = \frac{\mu}{RT} p = \frac{c_V p}{R e}. \quad (26)$$

Equations (24) and (26) are useful for initializing the simulation. It is sufficient to give the ambient pressure p_∞ and temperature T_∞ ; the corresponding specific internal energy e_∞ and density ρ_∞ can then be determined automatically.

Equations (1)–(9) and (25) form a closed system of four equations in the four unknowns ρ , \mathbf{u} , e and p . If we wish, we may eliminate p by inserting (25) in (3) and (9), obtaining

$$\begin{aligned} \frac{\partial \mathbf{u}}{\partial t} + \mathbf{u} \cdot \nabla \mathbf{u} + \frac{R}{c_V} \frac{\nabla(\rho e)}{\rho} &= \frac{\partial \mathbf{u}}{\partial t} + \mathbf{u} \cdot \nabla \mathbf{u} + \frac{R}{c_V} \frac{e \nabla \rho + \rho \nabla e}{\rho} \\ &= \frac{\partial \mathbf{u}}{\partial t} + \mathbf{u} \cdot \nabla \mathbf{u} + \frac{R}{c_V} \left[e \frac{\nabla \rho}{\rho} + \nabla e \right] = \mathbf{g}. \end{aligned} \quad (27)$$

and

$$\frac{\partial e}{\partial t} + \mathbf{u} \cdot \nabla e + \frac{R}{c_V} \frac{\rho e}{\rho} \nabla \cdot \mathbf{u} = \frac{\partial e}{\partial t} + \mathbf{u} \cdot \nabla e + \frac{R}{c_V} e \nabla \cdot \mathbf{u} = 0. \quad (28)$$

Equations (1), (27) and (28) form a closed system of three equations in the three unknowns ρ , \mathbf{u} and e . If desired, the pressure p can be then computed from (25).

Either way, equations (1), (3) and (9) with (25) allow the use of an explicit integration method. Note that it is not a priori known whether it is numerically better to eliminate p analytically by inserting (25), as was done in (27)–(28), or just keep p in (3) and (9), and evaluate $p = p(\rho, e)$ from (25).

3 Implementation strategies

In this section, we will discuss different strategies that can be used to discretize the model for a practical software implementation.

For simplicity, we choose the model variant with compressible flow at a constant temperature, and ignore the internal energy of the air. The equations for the three unknowns ρ , \mathbf{u} and p are, respectively, (1), (3) and (21). Expanding the parentheses in (1), we have (2). In summary:

$$\begin{aligned} \frac{\partial \rho}{\partial t} + \mathbf{u} \cdot \nabla \rho + \rho \nabla \cdot \mathbf{u} &= 0, \\ \frac{\partial \mathbf{u}}{\partial t} + \mathbf{u} \cdot \nabla \mathbf{u} + \frac{\nabla p}{\rho} &= \mathbf{g}, \end{aligned}$$

⁷See https://www.ohio.edu/mechanical/thermo/property_tables/air/air_Cp_Cv.html

$$p = \alpha \rho ,$$

where we have defined

$$\alpha := \frac{RT}{\mu} . \quad (29)$$

The value of the constant α is given by (22). Note that α is fairly large (on the order of 10^5), which may make the equation system numerically stiff. Physically, this is because the pressure response is very fast compared to the advection, and thus the physical behavior of the flow has separate fast and slow timescales.⁸⁹

Note that this numerical stiffness cannot be worked around by including temperature variations in the model. The magnitude of the internal energy variable e is large with respect to the other variables. As can be observed from (24), e.g. at an initial temperature of $T = 293.15$ K, the ratio $c_V T / \mu$ is in the order of 10^6 .

In index notation, using Cartesian tensors (valid in any orthonormal coordinate system), equations (1) and (3) become

$$\begin{aligned} \partial_t \rho + \partial_i (\rho u_i) &= 0 , \\ \partial_t u_i + u_j \partial_j u_i + \frac{1}{\rho} \partial_i p &= g_i . \end{aligned}$$

Expanding the parentheses in the first equation and inserting (21) into the second, the final equations for this model are

$$\partial_t \rho + (\partial_i \rho) u_i + \rho (\partial_i u_i) = 0 , \quad (30)$$

$$\partial_t u_i + u_j \partial_j u_i + \alpha \frac{1}{\rho} \partial_i \rho = g_i , \quad (31)$$

with α as defined in (29). Summation over repeated indices is implied.

3.1 Notes on moving least squares space discretization

Using moving least squares (MLS) for space discretization, we obtain explicit access to the start-of-timestep values of the space derivatives. Hence, we may simply replace the space differentiations in the original continuum form by calls to the MLS algorithm; this allows us to keep the space-discrete form of the equations formally the same as the original continuum form.

We also have the freedom to choose different sets of neighboring points for different parts of the differential operator, e.g. introducing upwinding only for the advection terms.

At everywhere except obstacles, we may use (x, y) coordinates, with axis 1 taken as x and axis 2 as y . At points on an obstacle surface, we can use (τ, n) coordinates, using (36) to update u_τ (the tangential component), and set the value of u_n (the normal component) from the boundary conditions. This gives a velocity vector $\mathbf{u} = (u_\tau, u_n)$, which can then be transformed into (x, y) coordinates.

When using MLS, one approach to do this is to first rotate the set of neighboring points (i.e. those used for MLS evaluation) from the (x, y) system into the local (τ, n) system. The distances between the points in the point cloud are of course invariant with respect to the orientation of the coordinate axes, but the components of the relative position vectors are not.

After this rotation, MLS is used to compute derivatives in (τ, n) coordinates. Then u_τ and u_n are updated, and finally the updated velocity vector is transformed into (x, y) coordinates. This final conversion makes the new value of $\mathbf{u} = (u_x, u_y)$ (at the point being processed) available for neighbors inside the domain at the next timestep.

Because we are dealing with a moving (vibrating) surface, this processing must be repeated for every obstacle surface point at every timestep. This is computationally heavy, but on the other hand points on the obstacle surface are a small subset of all points in the flow domain.

⁸This is easily seen by applying operator splitting (see subsection 3.4 for the idea) to the linear momentum balance equation, treating the advection and pressure response as separate subproblems. If \mathbf{u} and ρ are $O(1)$, the time derivative in the advection subproblem is $O(1)$, but in the pressure response subproblem it is $O(\alpha) \approx O(10^5)$.

⁹This is also the mathematical reason why slow airflows can be considered incompressible, but fast ones (typically Mach > 0.1 , see e.g. [refs Anderson Lighthill]) cannot. For slow flows, \mathbf{u} is $O(1)$, and the extremely fast pressure response is then modeled as instantaneous. For fast flows, \mathbf{u} is larger, and thus the disparity in the timescales (in the operator splitting consideration) becomes smaller. Hence, for fast flows, the advection and pressure response cannot be separated; in this region, the incompressible model, which assumes this timescale separation, becomes invalid.

An alternative, cheaper approach is to observe that the gradient is invariant with respect of rotations of the coordinate system, although its components are not. Thus we may compute the derivatives in the (x, y) coordinate system for all points, and only transform the result at the boundary points, which need it in (τ, n) coordinates. Then, as above, u_τ and u_n are updated, and the updated velocity vector is transformed into (x, y) coordinates.

3.2 Time integration

One approach for time integration is to transform the problem into a standard first-order system of ordinary differential equations (ODEs),

$$\frac{\partial \mathbf{w}}{\partial t} = f(\mathbf{w}, t), \quad (32)$$

with the initial condition $\mathbf{w}(0) = \mathbf{w}_0$, and then use classical time integrators for ODE systems.

In the form (32), \mathbf{w} is the space-discretized vector of unknowns. Timestepping (35)–(37) by an explicit integration method, only the old (known) values of the unknown fields are needed on the RHS. Alternatively, for implicit methods, an iterative procedure may be used.

To obtain an equation system in the form (32), we simply move all terms except the time derivative to the right-hand side in (30)–(31):

$$\partial_t \rho = -(\partial_i \rho) u_i - \rho (\partial_i u_i), \quad (33)$$

$$\partial_t u_i = g_i - u_j \partial_j u_i - \alpha \frac{1}{\rho} \partial_i \rho. \quad (34)$$

Even more explicitly, in component form (for convenience of software implementation)

$$\partial_t \rho = -(\partial_1 \rho) u_1 - (\partial_2 \rho) u_2 - \rho (\partial_1 u_1 + \partial_2 u_2), \quad (35)$$

$$\partial_t u_1 = g_1 - u_1 (\partial_1 u_1) - u_2 (\partial_2 u_1) - \alpha \frac{1}{\rho} (\partial_1 \rho), \quad (36)$$

$$\partial_t u_2 = g_2 - u_1 (\partial_1 u_2) - u_2 (\partial_2 u_2) - \alpha \frac{1}{\rho} (\partial_2 \rho). \quad (37)$$

Equations (35)–(37) are valid in any orthonormal coordinate system.

For a summary and discussion of some commonly used time integrators, see the separate document on integration of ODE systems (timeintegrators.pdf).

Note that MLS (in the form presented in this document) is of second order in space. Thus, if an explicit Runge–Kutta method is chosen, using RK3 or RK4 may not help the overall accuracy much. The second-order RK may be a preferable choice as the time integrator, since then the error orders are balanced, and no computational effort is wasted (at least judging asymptotically).

3.3 Spacetime MLS?

One possible approach is to use MLS in both space and time. This requires some extra thought, because the original MLS is designed to evaluate right-hand sides of equations like (33)–(34), so that some other method can then be used for utilizing the obtained value of the time derivative.

Theoretically, we may formulate an optimization problem to iteratively locally modify the new values of the unknown field, such that the MLS approximation of the time derivative (at the end of the timestep, evaluated using the iterative guess) best matches the value obtained from the right-hand side (also evaluated at the end of the timestep). This produces an implicit integrator based on MLS. It may be possible to bootstrap the iteration by a forward Euler initial guess, using the value of the time derivative as obtained based on the RHS at the start of the timestep.

The foreseeable advantages are that first, this allows taking neighborhoods in both space and time, and e.g. limiting the set of neighbors to the cone in which information has had time to travel during the timestep. Secondly, the timestep may be different for each spatial point; the points can be arbitrarily distributed in the spacetime domain, as long as there are locally enough of them to resolve the features of the solution.

However, as MLS is only of second order, this is probably not worth the computational cost of an optimization procedure, cheaper second-order methods being widely available.

3.4 Operator splitting

In computational fluid dynamics, operator splitting techniques are sometimes employed. Let us briefly review operator splitting.¹⁰

In *differential operator splitting*, the problem is recast into subproblems, which represent different physical processes. This is convenient, as the advection step may be separated out from the application of pressure and external forces. The subproblems are typically easier to solve than the original equation, and also, for certain standard subproblems, specialized solvers and algorithms may be available.

The original partial differential equation is

$$\frac{\partial u}{\partial t} + \mathcal{L}u = 0, \quad (38)$$

where \mathcal{L} is a differential operator (involving space derivatives). In operator splitting, suboperators \mathcal{L}_k are chosen such that

$$\mathcal{L} = \sum_{k=1}^n \mathcal{L}_k. \quad (39)$$

In general, the splitting is not unique (this is trivial; just group the terms of the original \mathcal{L} differently when defining the suboperators \mathcal{L}_k).

The first-order Marchuk–Yanenko method (with splitting error $O(\Delta t)$) works as follows:

$$\begin{aligned} \frac{\partial u^{(k)}}{\partial t} + \mathcal{L}_k u^{(k)} &= 0 \quad \text{in } (t_n, t_{n+1}) \quad \text{for } k = 1, 2, \dots, n \\ u^{(k)}(t_n) &= u^{(k-1)}(t_{n+1}), \\ u^{(0)}(t_{n+1}) &= u(t_n), \\ u(t_{n+1}) &= u^{(n)}(t_{n+1}). \end{aligned}$$

The first equation represents the subproblems, the second connects the substeps (in a daisy chain, each one feeding its output as input to the next one), the third one initializes the first substep, and the final equation just specifies that the solution of the last subproblem is taken as the new value of u .

The second-order symmetrized Strang splitting method (with splitting error $O((\Delta t)^2)$) for $n = 2$ works as follows:

$$\begin{aligned} \frac{\partial u}{\partial t} + \mathcal{L}_1 u &= 0 \quad \text{in } (t_n, t_{n+1/2}) \\ \frac{\partial u}{\partial t} + \mathcal{L}_2 u &= 0 \quad \text{in } (t_n, t_{n+1}) \\ \frac{\partial u}{\partial t} + \mathcal{L}_1 u &= 0 \quad \text{in } (t_{n+1/2}, t_{n+1}) \end{aligned}$$

Cases with $n > 2$ can be reduced recursively to this one by grouping the suboperators.

In the Strang method, the first subproblem evolves u into a half-timestep value using the \mathcal{L}_1 subproblem. The result is then re-interpreted as the value at the beginning of the timestep for the \mathcal{L}_2 subproblem, and evolved for a whole timestep. Finally, this result is re-interpreted as the value at the half-timestep point, and evolved (by half a timestep) to the end of the timestep using the \mathcal{L}_1 subproblem.

In the lecture notes, it is pointed out that Strang splitting is second-order accurate, and unconditionally stable if the discrete counterparts of \mathcal{L}_1 and \mathcal{L}_2 are positive definite matrices¹¹. The timestepping procedures used for the subproblems must be at least second-order accurate.

3.4.1 Semi-Lagrangian advection

When using operator splitting, the advection subproblem is often solved separately. One approach to solving this subproblem is semi-Lagrangian advection, summarized here.

This approach is mostly suitable for methods that allow interpolation to anywhere in the domain. Thus, it is not suitable for use with MLS, but is documented here for future extensions of this study into GPU computing. On

¹⁰For course material by prof. D. Kuzmin the following summary is based on, see <http://www.mathematik.uni-dortmund.de/~kuzmin/cfdintro/lecture11.pdf>

¹¹Which, of course, is not the case for the advection subproblem.

GPUs, for maximal speed a regular grid is preferable, because then no geometry information needs to be communicated, and the exact same computational kernel may be used for all pixels (or voxels in 3D) in the flow domain. Additionally, GPUs have hardware support for bilinear/trilinear interpolation, allowing fast interpolation of the solution field to any point in the domain.

A scalar field ϕ , subjected to advection by a velocity field \mathbf{u} , satisfies the first-order transport equation

$$\frac{\partial \phi}{\partial t} + \mathbf{u} \cdot \nabla \phi = 0, \quad (40)$$

when viewed in the Eulerian (laboratory) frame. If the transported quantity is a vector or a tensor, its components (which are scalar fields) can be treated independently.

Semi-Lagrangian advection is based on the physical interpretation of (40), namely that the field ϕ “belongs” to material particles, which are transported by the velocity field \mathbf{u} .

When taking a timestep of size Δt , although our representation of ϕ is Eulerian, we may try to locate the old position of the material particle now at the current position (following upwind along the streamlines of the flow), and look up the field value there. Then we assign that value to the current position. Using a simple linear approximation to locate the old position, we have:

$$\phi^{n+1}(\mathbf{x}) \leftarrow \phi^n(\mathbf{x} - \Delta t \cdot \mathbf{u}(\mathbf{x})), \quad (41)$$

where \mathbf{x} is the position vector (in any number of space dimensions) and n indexes the timestep. Out-of-bounds data are filled based on the inflow boundary condition. (Note that because we are moving against the flow, the procedure may end up outside the domain only via parts of the boundary that experience inflow. Strictly speaking this conclusion is valid only if the streamlines are traced exactly, but in practice it is good enough.)

This is a very simple, intuitive, physically based way to solve the transport equation, but the drawbacks are that it must be possible to evaluate the representation of ϕ at an arbitrary point \mathbf{x} (not just at grid nodes!), and that in this simple form the method is only first-order accurate (exhibiting large amounts of numerical diffusion).

3.4.2 MacCormack correction

To bring the accuracy of the semi-Lagrangian advection procedure to second order, the following sequence of operations may be used (refs CraneEtAl, SelleEtAl):

$$\begin{aligned} \hat{\phi}^{n+1} &\leftarrow A(\phi^n) \\ \hat{\phi}^n &\leftarrow A^{-1}(\hat{\phi}^{n+1}) \\ \phi^{n+1} &\leftarrow \hat{\phi}^{n+1} + \frac{1}{2}(\phi^n - \hat{\phi}^n), \end{aligned}$$

where ϕ is any scalar field, n indexes the timestep, and $A(\cdot)$ is the semi-Lagrangian advection procedure (with A^{-1} indicating advection backward in time). This is a modified version of the classical MacCormack correction procedure, requiring only two advection operations, due to ref SelleEtAl. The procedure is based on estimating the error introduced by the semi-Lagrangian advection; the estimate is then used to cancel most of the error.

The disadvantage is that while the original semi-Lagrangian advection procedure is unconditionally stable, this modified version is not, and will indeed create new spurious extremal values, leading to the Gibbs phenomenon and numerical instability. The authors suggest remedying this by using a limiter (similarly to flux limiters used in the finite volume method). Two limiters were suggested. Either

1. Clamp the values of ϕ^{n+1} between the min and max of ϕ^n , with the extrema searched over the set of grid nodes¹² which contribute to ϕ^{n+1} , or
2. Detect new extrema, and replace ϕ^{n+1} at those points by $\hat{\phi}^{n+1}$ (i.e. do not use the correction at those points; revert to the first-order solution there).

According to numerical tests by ref SelleEtAl, both limiters work well, and it does not matter much which one is used.

¹²This uses the property of linear (bilinear, trilinear) interpolation that it generates no new extrema; hence the extrema must be at the grid nodes.

4 Discretization by moving least squares (MLS)

In this section, we will briefly explain the moving least squares method (MLS). As was noted in the introduction, MLS is a local Taylor series optimization scheme to explicitly compute spatial derivatives (usually, up to 2nd order) of fields defined by their values on a set of points. It belongs to the family of collocation methods (sometimes called finite point methods). MLS is similar in spirit to finite differences, but generalized to arbitrary geometries.

The standard name “moving least squares” is a misnomer, as nothing actually moves in this method; the core idea would be better described by “local least squares”. The “moving” in the name comes from early serial implementations, where the local fitting procedure “moves” to each point in the point cloud in turn. For different points in the point cloud, the optimization problems are actually local and independent, so in practice it is preferable to process them in parallel.

Consider a set P , consisting of N distinct points x_i ($i = 1, 2, \dots, N$) in \mathbb{R}^d , which we will call the *point cloud*. The points in the set may be in any arbitrary order, and no mesh topology is assumed. Let us denote a set of geometric neighbors of the point i (excluding point i itself) by S_i , with each S_i a subset of P . Let the global indices of the points S_i (i.e. the k in x_k where $x_k \in S_i$) be denoted by I_i .

Usually, the set S_i is taken as the m nearest neighbors of x_i (by euclidean distance; for some arbitrarily chosen small integer m), but considering that we are solving an advection-dominated problem, it may be preferable, for numerical reasons, to look at the current value of the velocity field at the point i , and “upwind” the choice of neighbors to account for the direction of information flow.

Let f_i , $i = 1, \dots, N$ be a given, discrete set of data values, and let f denote the mapping $f(x_i) \mapsto f_i$, defined on the points in P . We will discuss the case of scalar data. If the data is rank-1 or higher (i.e. a vector or tensor), it can be decomposed into component form, each component treated separately.

In MLS, in principle we seek a set of independent, sufficiently continuous local approximations $\hat{f}_i(x)$, $x \in \mathbb{R}^d$, with each \hat{f}_i defined in a small neighborhood of x_i . We require that

$$\hat{f}_i(x_i) = f_i, \quad (42)$$

$$\sum_{k \in I_i} [f_k - \hat{f}_i(x_k)]^2 \rightarrow \min. \quad (43)$$

In other words, each \hat{f}_i must coincide with the data value f_i at the point x_i , and it must, in a least-squares sense, best fit the data values f_k over the chosen set of neighbor points $x_k \in S_i$ (corresponding to $k \in I_i$).

For simplicity, we will restrict our consideration to the two-dimensional case ($d = 2$), and second-order derivatives. Although the Euler flow equations are of the first order, this keeps the presentation more general, while remaining reasonably simple. Also, this makes the approximations second-order accurate (the first omitted terms in the Taylor series being of the third order). If we instead truncate the Taylor series after the linear terms, we obtain only first-order accuracy.

For the following discussion, let i be fixed. Although we do not know the local approximation function \hat{f}_i , we may write its value $\hat{f}_i(x_k)$ for any $k \in I_i$ (i.e. the value of \hat{f}_i at one of the neighbors of x_i), via multivariate Taylor expansion up to the second order, as

$$\hat{f}_i(x_k) = \hat{f}_i(x_i) + h_k a_1 + \ell_k a_2 + \frac{h_k^2}{2} a_3 + h_k \ell_k a_4 + \frac{\ell_k^2}{2} a_5 + O(h_k^3, \ell_k^3), \quad (44)$$

where

$$h_k := (x_k)_1 - (x_i)_1, \quad (45)$$

$$\ell_k := (x_k)_2 - (x_i)_2, \quad (46)$$

i.e. h_k and ℓ_k denote the x and y components of the relative position vector taken pointing from x_i to x_k . In order not to overburden the notation with indices, we have written simply h_k and ℓ_k , although the relative position vector depends on both i and k .

Generally, one must expand the Taylor series up to as many orders as is the highest derivative one wishes to approximate. This also specifies the degree of continuity required for \hat{f}_i (e.g. C^2 for second order, as here), as well as specifies the asymptotic accuracy of the method (which depends on the first omitted terms).

We can now trivially satisfy (42) just by choosing $\hat{f}_i(x_i) = f_i$, where the right-hand side is the data value f_i . Only (43) remains to be satisfied. If we drop the asymptotic error term from (44), we obtain the approximation

$$\hat{f}_i(x_k) \approx f_i + h_k a_1 + \ell_k a_2 + \frac{h_k^2}{2} a_3 + h_k \ell_k a_4 + \frac{\ell_k^2}{2} a_5 =: \bar{f}_k. \quad (47)$$

By the Taylor expansion, the quantities $a_j, j = 1, 2, \dots, 5$ in (44) and (47) are (note the numbering)

$$\begin{aligned} a_1 &= \frac{\partial \hat{f}_i}{\partial x} \Big|_{x=x_i}, & a_2 &= \frac{\partial \hat{f}_i}{\partial y} \Big|_{x=x_i}, \\ a_3 &= \frac{\partial^2 \hat{f}_i}{\partial x^2} \Big|_{x=x_i}, & a_5 &= \frac{\partial^2 \hat{f}_i}{\partial y^2} \Big|_{x=x_i}, \\ a_4 &= \frac{\partial^2 \hat{f}_i}{\partial x \partial y} \Big|_{x=x_i}, \end{aligned} \quad (48)$$

Note that the a_j depend on i (because they are specific to the function \hat{f}_i), but not on k . For all neighbor points $x_k \in S_i$, we have the same a_j .

Our problem reduces to finding these coefficients a_j . Let us denote (again, note the numbering)

$$\begin{aligned} c_k^{(1)} &:= h_k, & c_k^{(2)} &:= \ell_k, \\ c_k^{(3)} &:= \frac{h_k^2}{2}, & c_k^{(5)} &:= \frac{\ell_k^2}{2}, \\ c_k^{(4)} &:= h_k \ell_k. \end{aligned} \quad (49)$$

Each of the quantities $c_k^{(j)}$, which consist of the relative distances, depends on both i and k .

Observe that by (47) and (49),

$$\frac{\partial \bar{f}_k}{\partial a_j} = c_k^{(j)}, \quad (50)$$

which we will use later.

The requirement (43) seeks to best fit the local function \hat{f}_i to the data values f_k at the points x_k . However, we do not have access to the exact value $\hat{f}_i(x_k)$; we have only its approximation \bar{f}_k . If the neighbor points x_k are close enough to x_i , the asymptotic term in (44) is small, and in practice we may modify (43) slightly, replacing it with

$$\sum_{k \in I_i} [f_k - \bar{f}_k]^2 \rightarrow \min. \quad (51)$$

To solve (51), let us define the difference between the data f_k and the approximation \bar{f}_k at point x_k ,

$$e_k := f_k - \bar{f}_k, \quad (52)$$

and (one-half of) the total squared error

$$G(a_1, \dots, a_5) := \frac{1}{2} \sum_{k \in I_i} e_k^2. \quad (53)$$

The function G depends on the a_j via (52) and (47). Because $G \geq 0$ for any values of the a_j , and G is a quadratic function of the a_j , it has a unique extremal point, which is a minimum. The least-squares fit is given by this unique minimum of G :

$$\{a_1, \dots, a_5\}_{\text{optimal}} = \arg \min_{a_1, \dots, a_5} G(a_1, \dots, a_5).$$

To find this optimal point, we set all the partial derivatives of G to zero (w.r.t the a_j), obtaining a system of five equations:

$$\frac{\partial G}{\partial a_j} = 0, \quad j = 1, \dots, 5. \quad (54)$$

By solving the equation system (54), we obtain the optimal a_j . Let us write out (54). We have

$$\begin{aligned} \frac{\partial G}{\partial a_j} &= \sum_{k \in I_i} e_k \frac{\partial e_k}{\partial a_j} \\ &= \sum_{k \in I_i} [f_k - \bar{f}_k(a_1, \dots, a_5)] \left[-\frac{\partial \bar{f}_k}{\partial a_j} \right] = 0, \quad j = 1, \dots, 5, \end{aligned} \quad (55)$$

where we have used (52). Note that the data value f_k does not depend on a_j . Now the rest is essentially technique. Expanding the first \bar{f}_k in (55) by (47), applying the minus sign from the last term, and inserting (50), we have

$$\sum_{k \in I_i} \left(\left[-f_k + f_i + c_k^{(1)} a_1 + c_k^{(2)} a_2 + c_k^{(3)} a_3 + c_k^{(4)} a_4 + c_k^{(5)} a_5 \right] c_k^{(j)} \right) = 0, \quad j = 1, \dots, 5.$$

This can be written as a standard linear equation system

$$\sum_{n=1}^5 A_{jn} a_n = b_j, \quad j = 1, \dots, 5, \quad (56)$$

where

$$A_{jn} = \sum_{k \in I_i} c_k^{(n)} c_k^{(j)}, \quad (57)$$

$$b_j = \sum_{k \in I_i} [f_k - f_i] c_k^{(j)}. \quad (58)$$

Solving the linear equation system (56) produces the derivative approximations a_j , up to the second order.

The matrix A_{jn} is symmetric, $A_{jn} = A_{nj}$. Observe that the size of the system (56) does not depend on the number of neighbor points x_k used for the computation; the size is determined by the order of the Taylor approximation.

Finally, note that both A_{jn} and b_j depend on the index i (because of f_i , and the fact that $c_k^{(j)}$ consist of relative distances; recall the definition (49)). That is, each point x_i comes with its own A_{jn} and b_j , and thus (57)–(58) must be re-evaluated and (56) solved separately for each point x_i in P . However, the problems for different i are fully independent, and can be processed in parallel.

5 Initial and boundary conditions

- Ambient (far-field, free-stream) values u_∞ , p_∞ and T_∞ (implying also e_∞ and ρ_∞)
- Initial condition = ambient state. At the obstacle, velocity field must be modified to account for the no-penetration boundary condition.
- Inlet values u_{in} , p_{in} and T_{in} . For compatibility, best to use the same values as for the initial conditions.
- At an obstacle, $\mathbf{n} \cdot \mathbf{u} = \mathbf{n} \cdot \mathbf{u}_{\text{structure}}$, where \mathbf{n} is the unit normal vector of the surface of the obstacle. (The local normal component of fluid velocity must match the normal velocity of the obstacle.) For details on how to treat this for axially moving materials, see refs BanichukEtAlBook JeronenPhd2011.
- Normal vector of the obstacle determined from the transverse panel displacement function $w(x)$. From geometry, we have $\partial w / \partial x = \tan(\theta)$, where θ is the angle of the panel measured counterclockwise from $+x$. For compatibility, the panel may be taken to be initially perfectly flat.
 - In (x, y) coordinates, $\mathbf{u}_{\text{structure}} = (V_0 + V_0 \frac{\partial u}{\partial x} + \frac{\partial u}{\partial t}, V_0 \frac{\partial w}{\partial x} + \frac{\partial w}{\partial t})$, where (u, w) are, respectively, the axial and transverse displacement functions, and V_0 is the axial drive speed. The axial displacement u is measured with respect to a reference state moving toward $+x$ at constant velocity V_0 (ref KoivurovaSalonen).
 - The tangent and normal vectors of the panel surface are, respectively, $\mathbf{t} = (\cos \theta, \sin \theta)$ and $\mathbf{n} = (-\sin \theta, \cos \theta)$
 - Conversion from \mathbf{x} to \mathbf{X} coordinates in the plane:

$$\mathbf{v}_X = \mathbf{M} \mathbf{v}_x,$$

where the projection matrix \mathbf{M} is

$$\mathbf{M} = \begin{bmatrix} \hat{\mathbf{x}} \cdot \hat{\mathbf{X}} & \hat{\mathbf{y}} \cdot \hat{\mathbf{X}} \\ \hat{\mathbf{x}} \cdot \hat{\mathbf{Y}} & \hat{\mathbf{y}} \cdot \hat{\mathbf{Y}} \end{bmatrix},$$

and the axes of the coordinate systems are specified by the pairs of unit vectors $(\hat{\mathbf{x}}, \hat{\mathbf{y}})$ and $(\hat{\mathbf{X}}, \hat{\mathbf{Y}})$, respectively. In our case, $\hat{\mathbf{x}} = (1, 0)$, $\hat{\mathbf{y}} = (0, 1)$, $\hat{\mathbf{X}} = \mathbf{t}$, $\hat{\mathbf{Y}} = \mathbf{n}$:

$$\mathbf{M} = \begin{bmatrix} \mathbf{t}_x & \mathbf{t}_y \\ \mathbf{n}_x & \mathbf{n}_y \end{bmatrix} = \begin{bmatrix} \cos \theta & \sin \theta \\ -\sin \theta & \cos \theta \end{bmatrix} = \mathbf{R}(-\theta) ,$$

– Rotation in the plane:

$$\mathbf{v}_{\text{rotated}} = \mathbf{R} \mathbf{v}_{\text{original}} ,$$

where the rotation matrix $\mathbf{R} = \mathbf{R}(\theta)$ is

$$\mathbf{R}(\theta) = \begin{bmatrix} \cos \theta & -\sin \theta \\ \sin \theta & \cos \theta \end{bmatrix} .$$

- The flow field at any time t_0 (and its corresponding position, velocity data for the structure component) can be saved and later used as an initial condition.

6 Pressure difference across the structure

At any given point in the flow, the dynamic pressure is

$$q = \frac{1}{2} \rho u^2 , \quad (59)$$

where the fields ρ and u are sampled at the desired point. The total air pressure is

$$p = p_\infty - q = p_\infty - \frac{1}{2} \rho u^2 , \quad (60)$$

where p_∞ is the ambient pressure. Thus the pressure difference across the panel, driving the aerodynamic reaction force, is

$$\Delta p = p^+ - p^- = \frac{1}{2} \left[\rho^- (u^-)^2 - \rho^+ (u^+)^2 \right] , \quad (61)$$

where we have accounted for the fact that the ambient pressure p_∞ is the same on both sides. The plus and minus superscripts denote a limit from the indicated side ($+$ = upper, $-$ = lower).

7 Determining natural frequencies

- Apply classical measurement techniques to the direct time simulation, which plays the role of a physical experiment. Two approaches:
 - Impulse response: knock the structure at the beginning, and then “listen”: record time history and apply a fast Fourier transform (FFT) to obtain the frequency spectrum
 - Forced harmonic oscillations: make one of the supports oscillate at a prescribed frequency; change this frequency quasistatically (between simulations); look for frequencies which produce the largest maximum amplitude
- The pressure data can also be used to record audio from any desired point in the flow field. (Maybe taking the average from a small local area to filter out noise.)