The finite element method for the Navier-Stokes equations

Lucas Payne

November 1, 2021

Contents

Chapter 1

Continuum mechanics

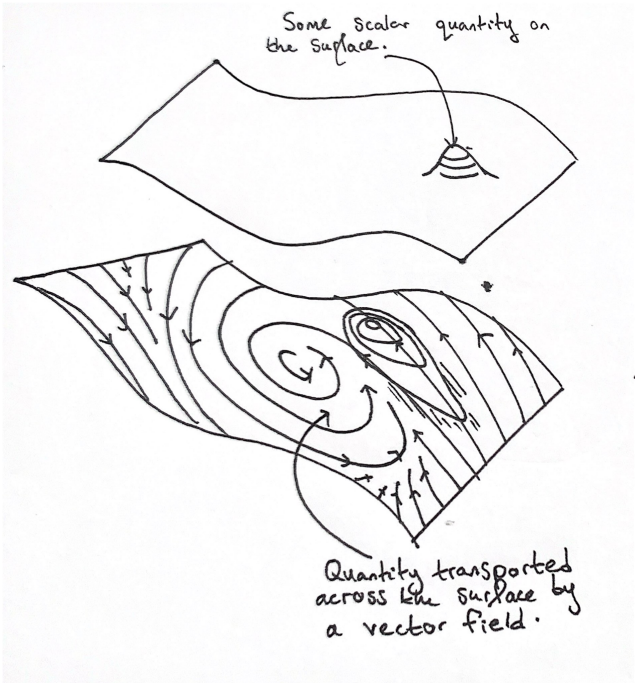
1.1 Introduction

— Introduction.

Although the focus later will be on fluid mechanics, a large set of fundamental concepts of solid and fluid mechanics are the same. The key distinction appears when one focuses on displacements and materials with "shear strength" (solids) versus flows and materials with no shear strength (common fluids).

1.2 Transport

Before considering continuum processes in the framework of Newtonian and Lagrangian mechanics, we will look at a fundamental notion of a "motion" of a point in a function space. Many continuum models in physics, such as the heat equation, Maxwell's equations, and the equations of fluid motion, are formed by *conservation equations*. These laws posit that the evolution of the state (represented by a function) is due to the transport of the quantity that the function measures, which is either pushed around (by some flux either predetermined or dependent on the current state), introduced into the system at sources, or leaves the system at sinks.



We consider here the transport of quantities (scalar, vector, and tensor) on a general finite-dimensional manifold M, colloquially called "the continuum". All transporting vector fields (or flux functions) are considered to be tangent to this manifold M.

1.2.1 Continuity equations and conservation laws

The integral form of a continuity equation

Consider some spatial quantity ϕ on M and a flux function j which by which this quantity flows around M. For clarity, we will begin by specializing ϕ to be a scalar, although later we will find it useful to transport vector quantities such as momentum. By definition we want this flux function to just push quantity around. The entering and exiting of quantity into and out of the system is determined by some arbitrary source function s (of the same kind as ϕ). These variables are related by the conservation condition

$$\frac{d}{dt}\int_{\Omega_0} \phi \, dx = \int_{\Omega_0} s \, dx + \int_{\partial\Omega_0} \phi j \cdot (-\hat{n}) \, dx \tag{1.1}$$

for arbitrary control volumes Ω_0 in the continuum. The term $-\hat{n}$ denotes the inward-pointing normal to the boundary of the control volume. This simply says that the change in the total quantity in the fixed control volume is accounted for exactly by that quantity pushed through the boundary by the flux function j, and the internal sources and sinks of quantity s.

The differential form of a continuity equation

A common technique in continuum modelling is the use of Stokes' theorem to simplify integral expressions. Stokes' theorem and its specialisations (such as the divergence theorem and Green's theorem) are really *definitions* of pointwise quantities such as the divergence and curl as limits of these integral expressions for arbitrarily small regions.

$$\nabla \cdot v \coloneqq \lim_{\epsilon \to 0} \frac{1}{\partial\Omega_\epsilon} \int_{\partial\Omega_\epsilon} v \cdot \hat{n} \, dx. \tag{1.2}$$

$$\int_{\partial\Omega} v \cdot \hat{n} \, dx = \sum_{i \in \mathcal{I}} \int_{\partial\Omega_i} v \cdot \hat{n} \, dx. \tag{1.3}$$

$$\int_{\partial\Omega} v \cdot \hat{n} \, dx = \int_{\Omega} \left[\lim_{\epsilon \to 0} \frac{1}{\partial\Omega_{x,\epsilon}} \int_{\partial\Omega_{x,\epsilon}} v \cdot \hat{n} \, dx' \right] dx. \tag{1.4}$$

Equation (1.1) becomes

$$\frac{\partial \phi}{\partial t} = s - \nabla \cdot (\phi j) \tag{1.5}$$

assuming that ϕj is sufficiently differentiable such that the limiting integral exists. Continuity relations are most naturally expressed in form (1.1), while the form (1.5) may be more useful for techniques such as finite differences. Later, when we discuss numerical methods for solving continuum models, we will not take this route. The methods of interest, *Galerkin* methods, work naturally with the integral form (1.1). It will be seen later that some constructions in the presentation of Galerkin methods, such as the "weak form" of a PDE, simply undo the differentialisation (for example (1.5)) of the original integral form of physical PDEs (for example (1.1)).

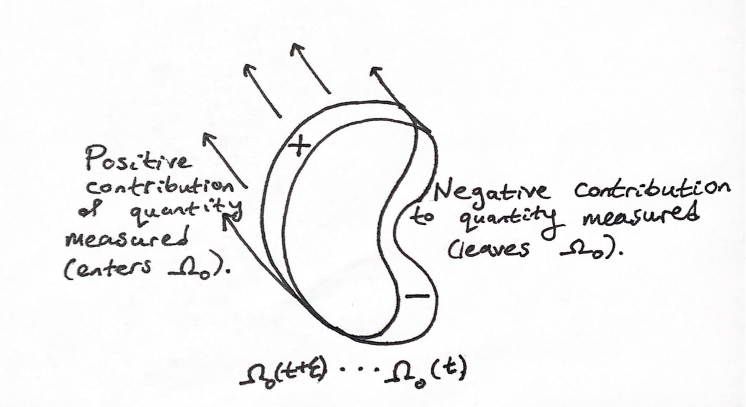
1.2.2 The Reynolds transport theorem

The integral form of Reynolds transport

With our integral formulation of a continuity relation (1.1), the control volume Ω_0 is fixed. We may change our perspective by considering, in addition to the flux function j (which transports quantity ϕ), another vector field $\hat{u}$ which will transport our control volume Ω_0. The rate of change of some time-dependent quantity γ in this *moving* control volume is expressed as

$$\frac{d}{dt}\int_{\Omega_0(t)} \gamma \, dx, \tag{1.6}$$

where $\Omega_0(t)$ implicitly denotes that Ω_0 is being transported under the flow of $\hat{u}$. Clearly, this rate of change of quantity γ is due to the motion of the control volume,



as well as internal changes of γ inside the (fixed) control volume. The formal expression of these contributions to the rate of change (1.6) is

$$\frac{d}{dt}\int_{\Omega_0(t)} \gamma \, dx = \int_{\Omega_0(t)} \frac{\partial \gamma}{\partial t} \, dx + \int_{\partial\Omega_0(t)} \gamma \hat{u} \cdot \hat{n} \, dx. \tag{1.7}$$

This result is called the *Reynolds transport theorem*, a generalisation of "differentiation under the integral sign" [10], otherwise named the Leibniz integral rule. See, for example, [8] for a formal derivation of (1.7).

The differential form of Reynolds transport

In the limit, with the routine application of Stokes' theorem, we can differentialise (1.7) to get

$$\frac{d}{dt}\int_{\Omega_0(t)} \gamma \, dx \quad \longrightarrow \quad \frac{\partial \gamma}{\partial t} + \nabla \cdot (\gamma \hat{u}), \tag{1.8}$$

as Ω_0 becomes small. The right-hand-side of (1.8) measures the change in volume of a quantity when a small control volume around the point of evaluation is moved, expanded or contracted by the flow field $\hat{u}$.

Reynolds transport applied to a continuity equation

Letting our quantity γ in (1.7) be the quantity ϕ transported by flux function j, described in continuity equation (1.1), we get a specialised form of the Reynolds transport theorem

for continuity equations. Term $\frac{\partial \gamma}{\partial t}$ in (1.7) becomes $\frac{\partial \phi}{\partial t}$ in the differential form of the continuity equation (1.5), giving

$$\begin{aligned} \frac{d}{dt}\int_{\Omega_0(t)} \phi \, dx &= \int_{\Omega_0(t)} -\nabla \cdot (\phi j) + s \, dx + \int_{\partial\Omega_0(t)} \phi \hat{u} \cdot \hat{n} \, dx \\ &= \int_{\Omega_0(t)} s \, dx + \int_{\partial\Omega_0(t)} \phi(\hat{u} - j) \cdot \hat{n} \, dx \end{aligned} \tag{1.9}$$

by Stokes' theorem. This has a clear interpretation. The $\hat{u} - j$ term is due to us wanting to measure the contributions to the total ϕ due to the moving boundary of Ω_0, where the motion that matters is *relative* to the flux of the quantity j. Specifically, if we move the control volume by the same flux function j (letting $\hat{u} = j$), we get

$$\frac{d}{dt}\int_{\Omega_0(t)} \phi \, dx = \int_{\Omega_0(0)} s \, dx. \tag{1.10}$$

In fact, (1.10) is just another form for the conservation law (1.1), where the "frame of reference" for measurement of ϕ follows the transport of ϕ. This simply means that as we follow some volume of quantity original situated in Ω_0, a conservation law posits that the only change detected is due to the source function s. In differential form (1.10) becomes

$$\frac{\partial \gamma}{\partial t} + \nabla \cdot (\gamma \hat{u}) = s, \tag{1.11}$$

a succint equivalent to (1.5). The idea of following the flow while making measurements is called the *Lagrangian* perspective, in contrast to the *Eulerian*, fixed, perspective.

1.2.3 Incompressible and compressible transport

Analogous to constraints on the motion of a finite mechanical system, we can constrain possible movement of our continuous quantity to *incompressible transport*. Much like how, in the framework of Lagrangian mechanics, constraints on motion are implicitly enforced by strong "virtual forces", constraining transport to be non-compressing will lead to the notion of *pressure*, derived in section 2.4.

Incompressibility

Incompressibility of control volumes gives a constraint on the form of a flux function j. We call this constrained flux function j non-compressing. By incompressibility we mean that a control volume being transported by j will have constant volume. While j may transport other quantities, we express incompressibility by requiring the flux function to transport a constant "volume quantity" with a corresponding null source function,

$$\phi_{\text{vol}} = 1, \quad s_{\text{vol}} = 0.$$

The corresponding conservation law, in differential form (1.5), is

$$\frac{\partial \phi_{\text{vol}}}{\partial t} = -\nabla \cdot (\phi_{\text{vol}} j) + s_{\text{vol}} \quad \Rightarrow \quad \nabla \cdot j = 0. \tag{1.12}$$

This is our non-compressing constraint on j, and has a clear interpretation, as there is a non-zero divergence of j if and only if there is an inward or outward flux which would contract or expand a transported control volume.

1.2.4 Transport of vector and tensor quantities

All previous discussion on the transport of scalar quantities applies trivially to vector and tensor quantities. This will soonest be of use in the discussion of conservation of linear momentum, a vector quantity However, some notational discussion is needed in order to establish differential forms of continuity equations and the Reynolds transport theorem.

Reynolds transport of vector and tensor quantities

For a general tensor quantity Γ, the integral form of Reynolds transport (1.7) is trivially

$$\frac{d}{dt}\int_{\Omega_0(t)} \Gamma \, dx \int_{\Omega_0(t)} \frac{\partial \Gamma}{\partial t} \, dx + \int_{\partial\Omega_0(t)} \Gamma \left(\hat{u} \cdot \hat{n}\right) \, dx. \tag{1.13}$$

The step to the differential form (1.8), however, needs some thought as rearranging

$$\text{“}\Gamma \left(\hat{u} \cdot \hat{n}\right) = \left(\Gamma \hat{u}\right) \cdot \hat{n}\text{”}$$

in order to apply the divergence theorem makes no sense. However, the divergence $\nabla\cdot$ was *defined* to evaluate the limit of this boundary integral for arbitrarily small Ω_0. We therefore have a natural generalisation of the divergence for arbitrary tensors Ψ, as the limit of the boundary integral of the *contraction* of Ψ with the outward normal $\hat{n}$ (which is a contravariant vector). The divergence of a rank n tensor is then a rank $n-1$ tensor,

$$\int_{\Omega_0} \nabla \cdot \Psi \, dx \coloneqq \int_{\partial\Omega_0} \Psi : \hat{n} \, dx. \tag{1.14}$$

We can then rewrite $\Gamma \left(\hat{u} \cdot \hat{n}\right)$ in (1.13) as

$$\Gamma \left(\hat{u} \cdot \hat{n}\right) = \left(\Gamma \otimes \hat{u}\right) : \hat{n},$$

where the tensor product $\otimes$ “defers contraction” of $\hat{u}$ with $\hat{n}$, by storing it as a component of product tensor $\Gamma \otimes \hat{u}$. This leads to a differentialisation of (1.13),

$$\frac{d_{\hat{u}}\Gamma}{d_{\hat{u}}t} = \frac{\partial \Gamma}{\partial t} + \nabla \cdot \left(\Gamma \otimes \hat{u}\right). \tag{1.15}$$

Conservation equations for vector and tensor quantities

With the previous ideas from tensor algebra, it will be easy to describe continuity relations for transport of tensors. The integral form of the scalar continuity equation (1.1), generalised to transported tensor Φ, trivially becomes

$$\frac{d}{dt}\int_{\Omega_0} \Phi \, dx = \int_{\partial\Omega_0} \Phi \left(j \cdot \left(-\hat{n}\right)\right) \, dx + \int_{\Omega_0} s \, dx. \tag{1.16}$$

By the same tensor algebra as above we have

$$\Phi \left(j \cdot \left(-\hat{n}\right)\right) = -\left(\Phi \otimes j\right) : \hat{n},$$

giving (1.16) in differential form as

$$\frac{\partial \Phi}{\partial t} = -\nabla \cdot \left(\Phi \otimes j\right) + s. \tag{1.17}$$

1.3 The kinematics of the continuum

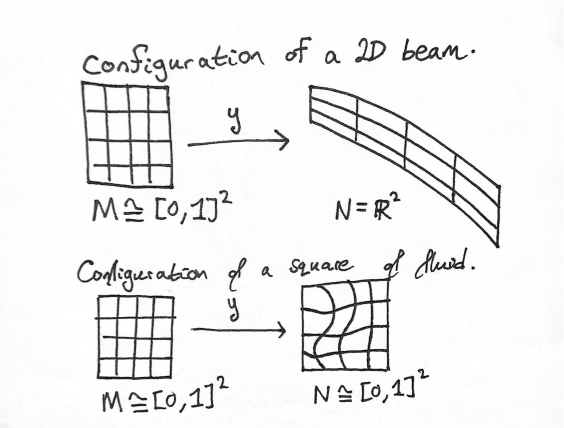
Transport equations are just one notion of "physical motion" in a continuum model. These transport equations, with prescribed flux and source functions, determine a continuous process on a fixed domain M. These conserved quantities are time-varying maps from M to some measurement space of scalars or tensors. Each map is a component of the total configuration space C, which clearly must be infinite-dimensional. We now consider another component of C which will let us model a physical domain with alterable shape. In our discussion we will consider a fixed time interval $[t_1, t_2]$ in which our physical motions will take place.

1.3.1 Position maps

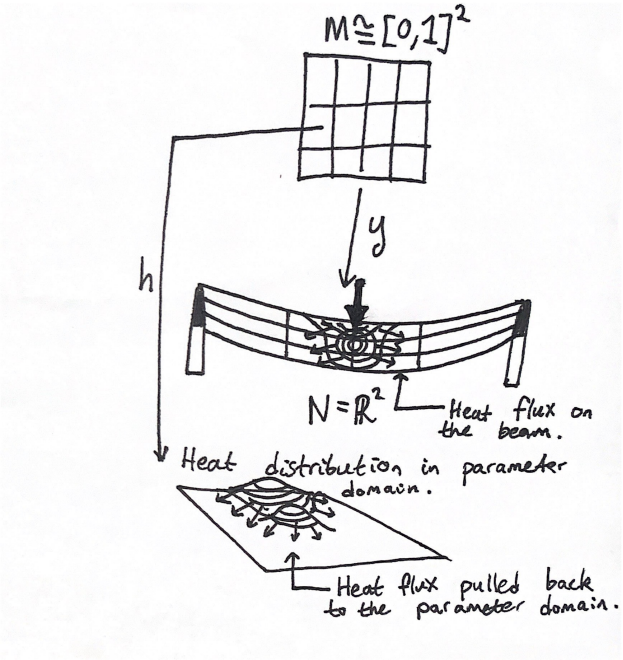
We may consider the manifold M as the parametric domain of some points living in an ambient manifold N. We will call this the "position map"

$$y : M \times [t_1, t_2] \to N.$$

In general, y needs not be continuous, differentiable, or invertible. These restrictions are only introduced in accord to the physical meaning of the position map. For example, models of small beam deflections may require continuity, and invertibility to prevent self-intersections.



As an example, suppose we are modelling the heat distribution of a 2D beam supporting a point load which is also a heat source. We could model the beam geometry as a smooth invertible map $y : [0, 1]^2 \to \mathbb{R}^2$, letting $M = [0, 1]^2$ and $N = \mathbb{R}^2$. The heat distribution on the beam could be represented by a function $h : M \to \mathbb{R}$, and a heat flux function j could be pulled back to M from N.

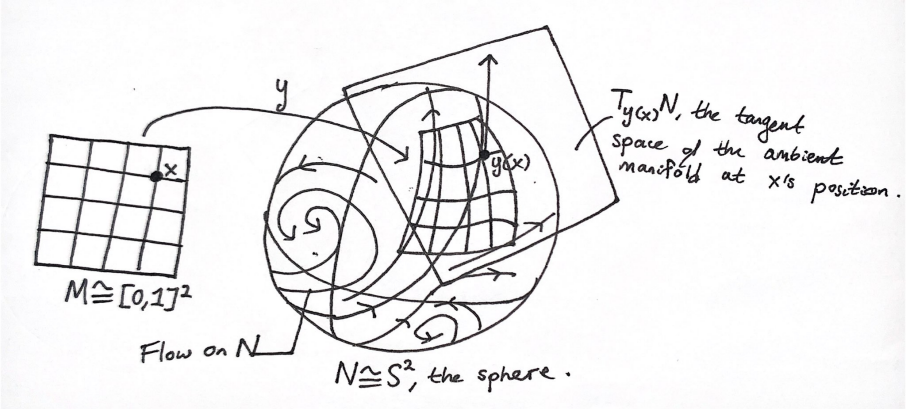


Although this model is so far hopelessly incomplete, we can see that position maps and transport equations are fundamental tools used for modelling the *geometry* of a problem.

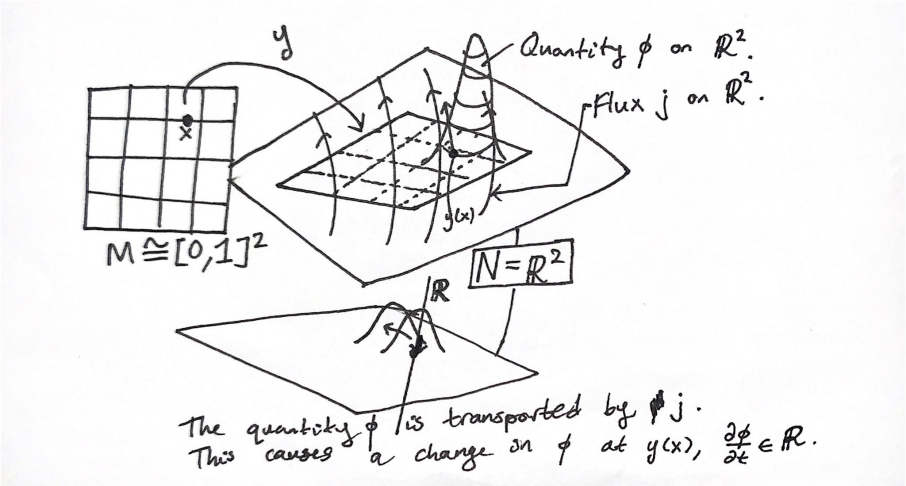
1.3.2 Velocity

As in the mechanics of a particle, each component of our state $q \in C$ will have a corresponding velocity which "generates" a physical motion of that component. In the case of the position map $y : M \times [t_1, t_2] \to N$, the velocity will be given by a vector in the tangent space of N at $y(x)$ for each parameter $x \in M$. This vector field is denoted $\dot{y}$. This is the *Lagrangian* description of motion. We will find it useful to instead use the *Eulerian* description, where we measure the velocity of the position map in the position domain N. Formally we denote this Eulerian velocity by the letter u, ubiquitous in fluid mechanics, and let

$$u(y,t) \coloneqq \dot{y}(y^{-1}(y,t),t) \quad \text{for all valid } y \in N.$$



For some transported scalar quantity $\phi : M \times [t_1, t_2] \to \mathbb{R}$, the tangent space at each point of $\mathbb{R}$ is $\mathbb{R}$, and therefore our velocity is represented by a scalar function $\frac{\partial \phi}{\partial t}$ giving local change in time of $\phi(x)$ for each $x \in M$.



We may denote our total velocities as a state variable $\dot{q}$. When we have state q, the corresponding velocity $\dot{q}$ will be in the tangent space of C at q, denoted T_qC. We can then define the space of velocities as the *tangent bundle* of the configuration space,

$$TC = \bigcup_{q \in C} T_qC.$$

1.3.3 The deformation and velocity gradients

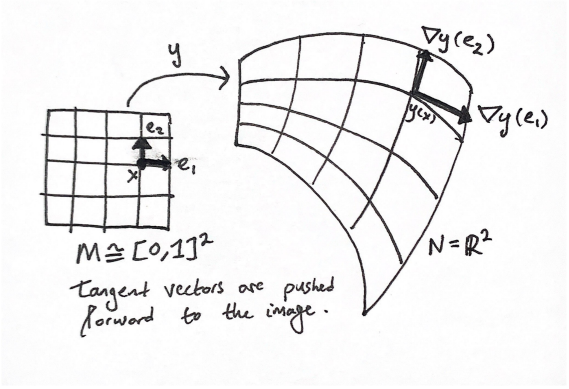
We may think of the mechanics of a particle as a special case of our continuum model, where M is a single point. In this case we have only one $x \in M$, so we cannot vary x. However, for a continuum parameter domain M we can take derivatives with respect to our parameter as well as time. We can extract important geometric/kinematic information from the spatial derivatives of our position map y.

The deformation gradient

The gradient of the position map y with respect to parameter $x \in M$ is called the *deformation gradient*

$$\nabla y. \tag{1.18}$$

The deformation gradient is equivalent to the *Jacobian matrix*, used to compute the *pushforward* of tangent vectors under the displacement map.



The determinant of the Jacobian matrix is usually denoted $J = \det(\nabla y)$, and is called the Jacobian.

The velocity gradient

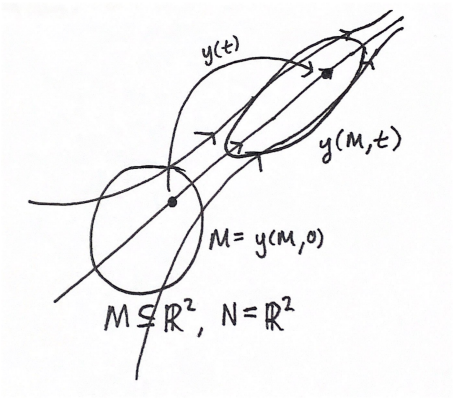
Letting u be our Eulerian velocity as defined above, we may express the position map through an ODE,

$$\frac{d}{dt}y(x,t) = u(y(x,t),t), \quad y(x,0) = y_0(x). \tag{1.19}$$

It is common, especially in flow problems, to let M be a subset of N which is the "initial geometry". In this case we could let the initial position map be the identity map

$$y_0(x) = x \in M \subset N.$$

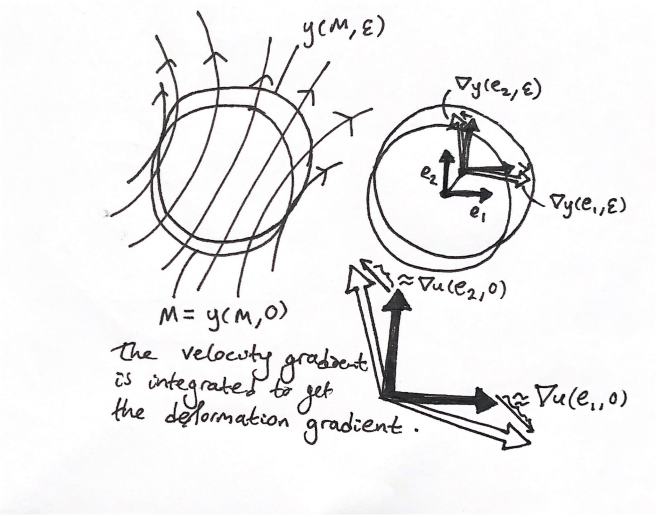
For example, given an initial disc in $\mathbb{R}^2$, we may prescribe a constant Eulerian velocity field u and see how the original disc is "mixed".



We can see that, in this case, it is meaningful to take spatial gradients in (1.19) to derive an ODE for the deformation gradient ∇y:

$$\nabla\left[\frac{d}{dt}y(x,t)\right] = \nabla u(y(x,t),t), \quad \nabla y(x,0) = I, \tag{1.20}$$

where I is the identity tensor. This is easily visualised.



We can see that the term ∇u is a "differential generator" of the deformation gradient. ∇u is called the *velocity gradient*.

1.3.4 Material points and material derivatives

The material derivative

Assuming an non-compressing flux function u which transports quantity ϕ, the differential form of the Reynolds transport theorem (1.11) becomes

$$\frac{\partial \phi}{\partial t} + \nabla \cdot (\phi u) = \frac{\partial \phi}{\partial t} + u \cdot \nabla \phi + \cancel{\phi \nabla \cdot u}. \tag{1.21}$$

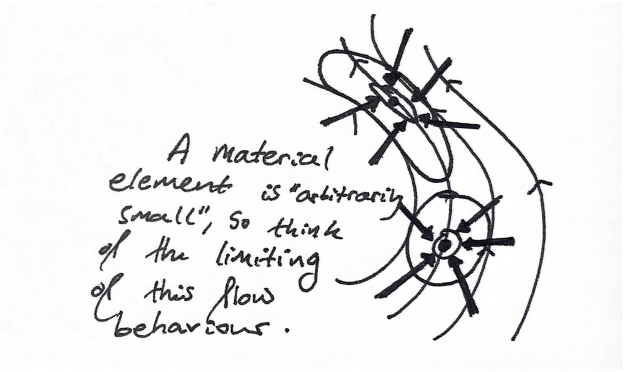
We define the *material derivative* to be

$$\frac{D}{Dt} \coloneqq \frac{\partial}{\partial t} + u \cdot \nabla. \tag{1.22}$$

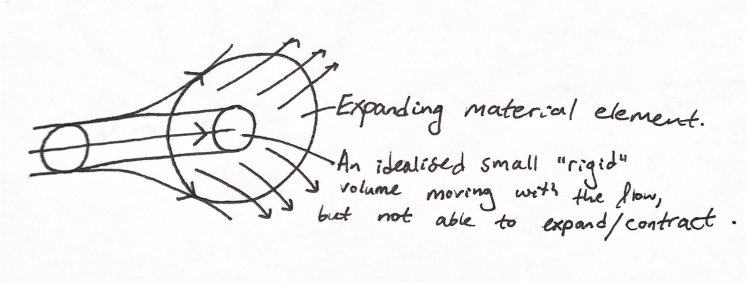
It is a convention to leave the vector field u implicit, as material derivatives are usually taken with respect to the velocity field. This derivative (1.22) will turn out to measure "per-volume" quantities.

Pieces of the continuum

A *material element* is a small piece of a continuum model which evolves with the displacement and flow. In fluid dynamics this is also called a fluid parcel, although it should not be thought of as an object placed in the fluid, but rather some tracer such as a non-diffusing dye.



In continuum mechanics a *material point* is the ideal point of the continuum, corresponding to some macroscopic averaging in physical models. At a certain point, we can imagine an arbitrarily small fluid parcel being transported by the flow. No matter how small this is, the parcel will still start to shear, expand, and contract due to the flow. The material derivative was defined as $\frac{D}{Dt} \coloneqq \frac{\partial}{\partial t} + u \cdot \nabla$. Notably this derivative contains no divergence term, and so does not measure the change of a quantity due to expansion and contraction of the arbitrarily small fluid parcel. Therefore we can think of the material derivative as acting on *per-volume* quantities.



1.4 The dynamics of the continuum

1.4.1 Conservation of mass

We will take for granted that there is some initial mass density function $\rho_0 : M \to \mathbb{R}^+$ which we would like to conserve.

The Lagrangian form of mass conservation

As the position map y evolves, for example under the action of Eulerian velocity field u in the ODE (1.19), we would like a mass density function $\rho : N \to \mathbb{R}^+$ to be greater if the position map "compresses" the material, and smaller if it "stretches" the material. Our aim is that the total mass measured in a control volume of N is the same as in its initial configuration. We can express this by the integral equation

$$\int_{\Omega_0} \rho_0(x)\,dx = \int_{y(\Omega_0,t)} \rho(y,t)\,dy, \tag{1.23}$$

where $y(\Omega_0, t)$ denotes the domain pushed forward by the position map. By the usual change of variables formula we have

$$\int_{\Omega_0} \rho_0(x)\,dx = \int_{\Omega_0} \det(\nabla y)\rho(y(x,t),t)\,dx, \tag{1.24}$$

where $J = \det(\nabla y)$ is the Jacobian, measuring the local change in the volume element due to the change of variables.

(draw this)

As (1.24) must hold identically for all Ω_0, we have the localised conservation law

$$\rho_0 = \det(\nabla y)\rho. \tag{1.25}$$

The Eulerian form of mass conservation

Conservation law (1.25) is simple. However, it is given in terms of absolute deformation gradient ∇y. Instead of asking how mass is distributed in comparison to its initial configuration, we can ask how mass is transported by u. This gives an instance of the continuity equation (1.1), where we have no source:

$$\frac{d}{dt}\int_{\Omega_0} \rho\,dx + \int_{\partial\Omega_0} \rho u \cdot \hat{n}\,dx = 0. \tag{1.26}$$

With application of Stokes' theorem we have

$$\frac{\partial \rho}{\partial t} + \nabla \cdot (\rho u) = 0. \tag{1.27}$$

Following [8], there is a more geometrical statement of this equation, which will be useful when we discuss "material points" By a divergence product rule (1.27) is equivalent to

$$\frac{\partial \rho}{\partial t} + u \cdot \nabla\rho + \rho\nabla \cdot u = 0 \quad \Rightarrow \quad \frac{D\rho}{Dt} = -\rho\nabla \cdot u, \tag{1.28}$$

where $\frac{D}{Dt}$ is the material derivative as defined in (1.22).

1.4.2 Conservation of linear momentum

If we conserve the linear momentum ρu, a "quantity of motion", under the flow of u, then we get a continuity equation

$$\frac{d}{dt}\int_{\Omega_0(t)} \rho u \, dx = \int_{\Omega_0(t)} \rho g \, dx + \int_{\partial\Omega_0(t)} \hat{t} \, dx, \tag{1.29}$$

a specific realisation of the Lagrangian continuity equation (1.10). The Lagrangian perspective is convenient as it allows us to separate interior and boundary forces on a moving piece of material. The term g is a regular body force per unit mass, where ρg corresponds to the source term s in (1.10). The boundary term involving $\hat{t}$, however, has no analogue in the scalar continuity equation (1.10). This vector term $\hat{t}$ is called the *traction* in continuum mechanics, and measures a local force exerted across the boundary of the control volume due to the immediately adjacent material.

The Euler-Cauchy stress principle

Clearly, in accord with Newton, we would like that two Ω_0 and Ω_0' which share a boundary element should have equal and opposite tractions across that boundary element. Since the normal $\hat{n}$ represents a boundary element, and is negative for the opposite element, if $\hat{t}$ is linear in $\hat{n}$ we have this required property. We can then let (1.29) become

$$\frac{d}{dt}\int_{\Omega_0(t)} \rho u \, dx = \int_{\Omega_0(t)} \rho g \, dx + \int_{\partial\Omega_0(t)} \sigma : \hat{n} \, dx \tag{1.30}$$

where σ is termed the *Cauchy stress tensor*. This is the integral form of the *Cauchy momentum equation*, the standard form of $F = ma$ in continuum mechanics.

Differentializing the Cauchy momentum equation

By application of the Reynolds transport theorem (1.13) to (1.30) we get

$$\int_{\Omega_0(0)} \frac{\partial(\rho u)}{\partial t} \, dx + \int_{\partial\Omega_0(0)} \rho u (u \cdot \hat{n}) \, dx = \int_{\Omega_0(0)} \rho g \, dx + \int_{\partial\Omega_0(0)} \sigma : \hat{n} \, dx. \tag{1.31}$$

Differentializing (1.31), by our previously derived tensor identities, gives

$$\frac{\partial(\rho u)}{\partial t} + \nabla \cdot (\rho u \otimes u) = \rho g + \nabla \cdot \sigma. \tag{1.32}$$

This is called the *conservative form* of the Cauchy momentum equation. We can derive another, possibly more convenient form of (1.32) using the fact that ρ is conserved and has no source. Here, this will be derived purely algebraically, although the final form of the equation has a useful interpretation. Expanding the partial derivative

$$\frac{\partial(\rho u)}{\partial t} = \rho \frac{\partial u}{\partial t} + u \frac{\partial \rho}{\partial t}$$

is simple. The tensor divergence $\nabla \cdot (\rho u \otimes u)$ is defined such that

$$\int_{\Omega_0} \nabla \cdot (\rho u \otimes u) \, dx = \int_{\partial\Omega_0} (\rho u \otimes u) : \hat{n} \, dx = \int_{\partial\Omega_0} \rho u (u \cdot \hat{n}) \, dx$$

for arbitrary control volumes Ω_0. As Ω_0 becomes small, we can separately assume u and ρu are constant to derive

$$\int_{\partial\Omega_0} \rho u (u \cdot \hat{n}) \, dx = u \int_{\partial\Omega_0} (\rho u) \cdot \hat{n} \, dx + \rho u \cdot \int_{\partial\Omega_0} u \hat{n} \, dx \quad + \quad \cdots$$

where a trailing term becomes neglible for a small control volume. This gives a "tensor product rule" for the divergence,

$$\nabla \cdot (\rho u \otimes u) = u\nabla \cdot (\rho u) + \rho u \cdot \nabla u. \tag{1.33}$$

Equation (1.32) then becomes

$$\rho \frac{\partial u}{\partial t} + u\frac{\partial \rho}{\partial t} + u\nabla \cdot (\rho u) + \rho u \cdot \nabla u = \rho g + \nabla \cdot \sigma.$$

Noting that $\frac{\partial \rho}{\partial t}$ is already given by continuity equation (1.27)

$$\frac{\partial \rho}{\partial t} = -\nabla \cdot (\rho u),$$

as mass is transported by u and has no source, we get

$$\rho \frac{\partial u}{\partial t} - \cancel{u\nabla(\rho u)} + \cancel{u\nabla \cdot (\rho u)} + \rho u \cdot \nabla u = \rho g + \nabla \cdot \sigma.$$

Finally, the material derivative as defined in section (ref) is helpful in simplifying the above to

$$\rho \frac{Du}{Dt} = \rho g + \nabla \cdot \sigma. \tag{1.34}$$

This form of (1.32) is called the *convective form* of the Cauchy momentum equation, and is more obviously a form of $F = ma$. Recall that the material derivative is defined as

$$\frac{D}{Dt} \coloneqq \frac{\partial}{\partial t} + u \cdot \nabla,$$

which measures the rate of change of a pointwise quantity from the perspective of a particle moving with the flow field u. The equation (1.34) then says that, if the continuum consists of idealised points each with a certain linear momentum (in the particle sense), deflection of their inertial path is due only to the application of a body force ρg at this point, and a total traction force exerted by the surrounding material.

1.4.3 Constitutive relations

The conservation equations derived are conservation of mass (1.27) and conservation of linear momentum (1.34). Here the equations are given in their typical differential form for flow problems, with the Eulerian perspective of mass conservation, and the convective form of linear momentum conservation:

$$\begin{aligned} \frac{D\rho}{Dt} + \rho\nabla \cdot u = 0 \quad &\text{(Conservation of mass)}, \\ \rho\frac{Du}{Dt} = \rho g + \nabla \cdot \sigma \quad &\text{(Conservation of linear momentum)}. \end{aligned} \tag{1.35}$$

The unknowns include mass density ρ and velocity u, described by $1 + d$ scalar functions where d is the dimension of the domain. We can see there are $1 + d$ equations by expanding u in terms of components in some basis.

$$\begin{aligned} \rho\frac{Du_i}{Dt} = \rho g_i + (\nabla \cdot \sigma)_i \quad &\text{(Conservation of linear momentum components)}, \\ i = 1, \cdots, d. \end{aligned}$$

When g and σ are known, this system is well-formed, but not very interesting. This effectively models a continuum of non-interacting material points, with a linear-momentum-introducing source function $\rho g + \nabla \cdot \sigma$. However, we usually let g be known (as this is a per-mass external body force, for example gravity), and the Cauchy stress tensor σ be unknown. This gives d^2 more unknowns, so the system is heavily underdetermined. In effect, the system (1.35) is underdetermined because we don't have a specific material in mind.

Constitutive relations

A specification of the Cauchy stress tensor σ is called a *constitutive relation*, as σ depends on the "material constitution". A constitutive relation specifies how the material configuration induces forces on the material, or rather, how kinematics is related to dynamics. With σ specified, the Cauchy momentum equation (1.34) becomes well-posed (however, in general, σ may depend on new variables such as temperature, requiring further equations).

Completing the equations

We have so far been working quite generally with the forms of continuum mechanics models and their constraints. Although we have made some assumptions (for example, we require σ to be a purely local function, an idealisation due to Cauchy [ref]), we must so far leave our systems underdetermined. In the next chapter, we will investigate an important constitutive relation for the stress σ, forming what is called a *Navier-Stokes fluid*, giving a complete system of equations called the *Navier-Stokes equations*.

Chapter 2

The Navier-Stokes equations

2.1 Introduction

The incompressible Navier-Stokes equations model the motion of a common kind of viscous fluid called a *Newtonian fluid*. They are:

- The Cauchy momentum equation (1.34) for constant mass density ρ and velocity u,
- an incompressibility constraint $\nabla \cdot u = 0$ and unknown pressure p,
- and a concrete constitutive relation for the deviatoric stress τ.

In anticipation, their common form is

$$\rho \frac{Du}{Dt} = -\nabla p + \nabla \cdot \tau + \rho g, \quad \nabla \cdot u = 0, \tag{2.1}$$

where τ is defined by Stokes' constitutive relation

$$\tau = \mu \left(\nabla u + \nabla u^T \right), \tag{2.2}$$

where μ is called the *viscosity*, and ∇u is the velocity gradient defined in section 1.3.3, measuring the local deformation of a small control volume under the flow of u. We will assume their domain is a subset of $\mathbb{R}^d$, where typically $d = 2$ or 3, although the Navier-Stokes equations can be solved in curved domains (see [23]). Alongside a domain and appropriate initial and boundary conditions, the Navier-Stokes equations (2.1) form a concrete flow problem which can be solved numerically, or in special situations analytically.

2.2 The equations of fluid motion

We begin with the standard conservation equations of mass and linear momentum, (1.35):

$$\begin{aligned} \frac{D\rho}{Dt} + \rho \nabla \cdot u = 0 \quad &\text{(Conservation of mass)}, \\ \rho \frac{Du}{Dt} = \rho g + \nabla \cdot \sigma \quad &\text{(Conservation of linear momentum)}. \end{aligned} \tag{2.3}$$

As discussed in section 1.4.3, we need to specify the Cauchy stress tensor σ such that the system is well-formed. It would be helpful to restrict the possible form of σ. In the next section we will show that σ's antisymmetric part describes a couple force, a force which induces an angular momentum (a "spin") in a small control volume, but which doesn't contribute to linear momentum. Therefore, if we do not want couple forces, we want σ to be symmetric.

2.2.1 Conservation of angular momentum

Angular momentum is traditionally presented in terms of rigid bodies, bodies subject to a distance-preserving constraint between material points. It is the moment of linear momentum.

The discussion in section 1.3.4 indicates that we can think of a very small rigid body at a material point, subject to the flow. This will be subject to a "spin force".

Let the material point be $c \in \mathbb{R}^d$, which will act as a "centre of mass', and let $c \in \Omega_0$. Define $\bar{x} \coloneqq x - c$. Define the moment of linear momentum as

$$\int_{\Omega_0} \bar{x} \wedge (\rho u)\ dx. \tag{2.4}$$

The symbol $\wedge$ indicates the cross product, whose value should be thought of as a pseudo-vector or "plane with magnitude". We will call this the angular momentum of the control volume Ω_0. We repeat here an integral form of linear momentum conservation, which already must hold:

$$\int_{\Omega_0} \frac{\partial(\rho u)}{\partial t}\, dx + \int_{\partial\Omega_0} \rho u(u \cdot \hat{n})\, dx = \int_{\Omega_0} \rho g\, dx + \int_{\partial\Omega_0} \sigma\hat{n}\, dx. \tag{2.5}$$

It is simple to derive an angular momentum conservation equation, just by taking moments of each vector quantity:

$$\int_{\Omega_0} \bar{x} \wedge \frac{\partial(\rho u)}{\partial t}\, dx + \int_{\partial\Omega_0} \bar{x} \wedge (\rho u)\,(u \cdot \hat{n})\ dx = \int_{\Omega_0} \bar{x} \wedge (\rho g)\ dx + \int_{\partial\Omega_0} \bar{x} \wedge (\sigma\hat{n})\ dx. \tag{2.6}$$

(This precludes the introduction of surface and body couples which induce no linear momentum but do induce angular momentum [8]. We ignore these torques.)
If (2.5) holds, should (2.6) hold automatically?

By Stokes' theorem, the final term in (2.5) can be written as

$$\int_{\partial\Omega_0} \sigma\hat{n}\, dx = \int_{\Omega_0} \nabla \cdot \sigma\, dx.$$

(draw the differentialization happening at each point, contracting a small control volume).

With the help of the above picture, we can try to derive a per-point form for the final term in (2.6),

$$\int_{\partial\Omega_0} \bar{x} \wedge (\sigma\hat{n})\ dx = \int_{\Omega_0} \cdots ? \cdots\ dx.$$

At a point c in Ω_0, we contract an even smaller control volume Ω_c around that point, in order to express the boundary integral over $\partial\Omega_0$ in terms of smaller boundary integrals on the interior. As this takes a limit, we can separately assume that $\bar{x} = x - c$ and σ are constant in Ω_c, giving the "product rule"

$$\int_{\partial\Omega_c} \bar{x} \wedge (\sigma\hat{n})\ dx \quad \to \quad \bar{x} \wedge \nabla \cdot \sigma + (\text{some term keeping } \sigma \text{ constant}).$$

We can reason geometrically to find the final term. We can split σ into its symmetric part S and antisymmetric part N:

$$\sigma = \frac{1}{2}\left(\sigma + \sigma^T\right) + \frac{1}{2}\left(\sigma - \sigma^T\right) = S + N.$$

Antisymmetric matrices have a lot to do with rotations: A special property of antisymmetric matrices is

$$\langle x, Nx\rangle = \left\langle x, N^T x\right\rangle = \langle x, -Nx\rangle \quad \Rightarrow \quad \langle x, Nx\rangle = 0.$$

In fact the antisymmetric matrices are exactly those which generate rotations (formally, $\exp(N)$ is orthogonal, where exp is the matrix exponential). If σ is kept constant over $\partial\Omega_c$, we can visualise the tractions contributed by the symmetric and antisymmetric parts of σ:

(draw this)

By the above diagram we can conclude that, letting $\sigma = S + N$ be constant,

$$\int_{\partial\Omega_c} \bar{x} \wedge ((S+N)\,\hat{n})\; dx = \int_{\partial\Omega_c} \bar{x} \wedge (N\hat{n})\; dx \quad \rightarrow \quad \widehat{N}$$

where $\widehat{N}$ is defined in $\mathbb{R}^3$ as the axis-angle vector representation of the differential rotation corresponding to N:

$$\begin{gathered} Nv = \widehat{N} \wedge v, \quad v \in \mathbb{R}^3 \\ N = \begin{bmatrix} 0 & -\omega_z & \omega_y \\ \omega_z & 0 & -\omega_x \\ -\omega_y & \omega_x & 0 \end{bmatrix} \Rightarrow \widehat{N} = \begin{bmatrix} \omega_x \\ \omega_y \\ \omega_z \end{bmatrix}. \end{gathered} \tag{2.7}$$

We now have

$$\int_{\partial\Omega_c} \bar{x} \wedge (\sigma\hat{n})\; dx \quad \rightarrow \quad \bar{x} \wedge \nabla\cdot\sigma + \widehat{N}, \tag{2.8}$$

which gives

$$\int_{\partial\Omega_0} \bar{x} \wedge (\sigma\hat{n})\; dx = \int_{\Omega_0} \bar{x} \wedge \nabla\cdot\sigma + \widehat{N}\, dx. \tag{2.9}$$

This shows that for the tractions measured across the boundary, although their contributions to the linear momentum of the control volume conserve it, there is another contribution to the angular momentum, which is the "spin part" of σ. To show this more directly, localise the linear conservation equation (2.5):

$$\int_{\Omega_0} \frac{\partial(\rho u)}{\partial t} + \nabla\cdot(\rho u \otimes u) - \rho g - \nabla\cdot\sigma\, dx = 0. \tag{2.10}$$

The corresponding differential form of (2.6) is

$$\int_{\Omega_0} \bar{x} \wedge \left[\frac{\partial(\rho u)}{\partial t} + \nabla\cdot(\rho u \otimes u) - \rho g - \nabla\cdot\sigma\right] dx = \int_{\Omega_0} \widehat{N}\, dx. \tag{2.11}$$

As the conservation law (2.10) must hold for all Ω_0, we can see that for (2.10) to imply (2.11) (for linear momentum conservation to imply angular momentum conservation) we need

$$\widehat{N} = 0 \quad \Rightarrow \quad \sigma = \sigma^T. \tag{2.12}$$

So, without the introduction of any explicit angular momentum sources (body and surface couples), the Cauchy stress tensor σ is required to be symmetric, reducing the d^2 unknowns to $d(d+1)/2$ unknowns.

(May be $\widehat{N}/2$, maybe a sign error. Check Leal p68.)

2.2.2 Conservation of energy

$$\begin{aligned} \frac{D\rho}{Dt} + \rho\nabla\cdot u = 0 \quad &\text{(Conservation of mass)},\\ \rho\frac{Du}{Dt} = \rho g + \nabla\cdot\sigma \quad &\text{(Conservation of linear momentum)},\\ \sigma = \sigma^T \quad &\text{(Conservation of angular momentum)}. \end{aligned}$$

2.3 Scaling and dimension

2.3.1 The Reynolds number

2.4 Stokes flow and the meaning of pressure

If we assume that the advective term $u\cdot\nabla u$ in the incompressible Navier-Stokes equations is "small", we can ignore it and derive the linear *unsteady Stokes equations*:

$$\rho\frac{\partial u}{\partial t} = \mu\Delta u + \rho g - \nabla p, \quad \nabla\cdot u = 0. \tag{2.13}$$

We are assuming validity for low Reynolds number $Re \ll 1$, where convective behaviour is neglible compared to the viscous forces, which for a Navier-Stokes fluid "diffuse" the linear momentum. Setting the left-hand-side of (2.13) to zero results in the *steady Stokes equations*

$$\mu\Delta u + \rho g - \nabla p = 0, \quad \nabla\cdot u = 0. \tag{2.14}$$

Time-dependent equation (2.13) can be thought of as a "gradient descent" to find the steady Stokes flow (2.14). The steady Stokes equation is a constrained vector Poisson equation, where we have introduced pressure p explicitly. It is well-known, by Dirichlet's principle, that we can think of a weak solution to the unconstrained vector Poisson equation as a minimiser of the Dirichlet energy,

$$\underset{u}{\text{minimize}} \quad E(u) = \frac{\mu}{2}\left\langle \nabla u, \nabla u\right\rangle - \left\langle u, \rho g\right\rangle. \tag{2.15}$$

We can validate this by computing the Euler-Lagrange equations:

$$\frac{\delta E}{\delta u} = \frac{\partial \mathcal{L}}{\partial u} - \frac{d}{dx}\frac{\partial \mathcal{L}}{\partial u_x} = -\rho g - \mu\Delta u = 0.$$

We now introduce the incompressibility constraint $\nabla\cdot u = 0$, giving the constrained minimization

$$\begin{aligned} \underset{u}{\text{minimize}} \quad & E(u) = \frac{\mu}{2}\left\langle \nabla u, \nabla u\right\rangle - \left\langle u, \rho g\right\rangle\\ \text{subject to} \quad & \nabla\cdot u = 0. \end{aligned} \tag{2.16}$$

It is not immediately obvious how to form the constrained Euler-Lagrange equations here, as $\nabla\cdot$ is a differential operator. We cannot just write

$$\text{“}\frac{\delta E}{\delta u} = \lambda\nabla\,\text{.”}$$

for scalar function λ, as we can with a pointwise linear constraint such as $u\cdot v = 0$ for some vector field v. However, this is just a problem of notation. The evaluation of energy change with perturbations is defined as

$$\left\langle \frac{\delta E}{\delta u}, \delta u\right\rangle = \int_\Omega \frac{\delta E}{\delta u}\cdot\delta u\, dx.$$

We want this measure of energy change to be purely a divergence measure, up to a scalar multiplier λ:

$$\int_\Omega \frac{\delta E}{\delta u} \cdot \delta u \, dx = \int_\Omega \lambda \nabla \cdot \delta u \, dx. \tag{2.17}$$

This means that virtual displacements with $\nabla \cdot \delta u = 0$ will not cause an energy change, which is the condition that we want for a stationary point. We can now apply integration by parts to (2.17), assuming that δu vanishes on the boundary of the domain, to get

$$\int_\Omega \frac{\delta E}{\delta u} \cdot \delta u \, dx = -\int_\Omega \nabla \lambda \cdot \delta u \, dx. \tag{2.18}$$

We can now reasonably apply the localisation step to get the constrained Euler-Lagrange equations

$$\frac{\delta E}{\delta u} = -\nabla \lambda \quad \equiv \quad \mu \Delta u + \rho g - \nabla \lambda = 0. \tag{2.19}$$

Along with the constraint $\nabla \cdot u = 0$, this is just the steady Stokes equations (2.14), where $\lambda = p$! We can see that the pressure p is actually a Lagrange multiplier, which measures a virtual force that responds to virtual displacements which would break the constraint of incompressibility. In fact, we may think of this as a derivation of the pressure.

Alternative direct derivation in terms of a modified energy

Previously, we emphasized the meaning of the Lagrange multiplier. One utility of Lagrange's methods is their automated calculational power. It is standard to express that a solution to the optimization problem (2.16), with a differentiable equality constraint, is a stationary point of the modified energy

$$L(u, \lambda) \coloneqq \frac{\mu}{2} \langle \nabla u, \nabla u \rangle - \langle u, \rho g \rangle - \langle \lambda, \nabla \cdot u \rangle . \tag{2.20}$$

We can take an evaluated first variation with respect to u to get

$$\left\langle \frac{\delta L}{\delta u}, \delta u \right\rangle = \langle -\rho g - \mu \Delta u, \delta u \rangle - \langle \lambda, \nabla \cdot \delta u \rangle ,$$

which by integration by parts becomes

$$\left\langle \frac{\delta L}{\delta u}, \delta u \right\rangle = \langle -\rho g - \mu \Delta u + \nabla \lambda, \delta u \rangle . \tag{2.21}$$

We then get

$$\begin{aligned} \frac{\delta L}{\delta u} &= -\rho g - \mu \Delta u + \nabla \lambda = 0, \\ \frac{\delta L}{\delta \lambda} &= -\nabla \cdot u = 0, \end{aligned} \tag{2.22}$$

which are the steady Stokes equations (2.14) with pressure $p = \lambda$.

2.4.1 Application to hydrostatics

For example, we may imagine the steady Stokes equations modelling a calm sea with a flat seabed. We can let the body force be gravity described by a potential ϕ:

$$\rho g = -\nabla \phi.$$

If we make a perturbed displacement of the velocity field at the bottom of the ocean, supposing that some volume of water is beginning to expand, we are working against gravity as well as our virtual force, pressure.

(draw this)

Chapter 3

Two Galerkin methods for Poisson's equation

3.1 Introduction

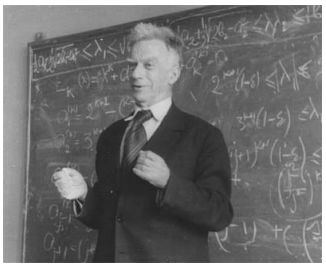
Continuum mechanics provides the foundation for understanding of fluid motion, and the Navier-Stokes equations for a Newtonian fluid are the primary model for the prediction of fluid behaviour in engineering. The domain and initial/boundary conditions can be arbitrarily complex. For example, fluid motion is often simulated through a digital surface model of a real-world object or system created in computer-aided design software. On the other side of the coin, fluid models are very often used in modern film, both live action and animated, and all relevant geometry can be adjusted by artists for a desired visual effect.

Peter Dirichlet
(1805–1859)

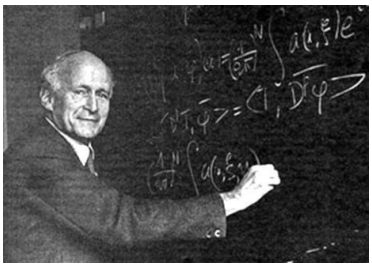
David Hilbert
(1862–1943)

Walther Ritz
(1878–1909)

Boris Galerkin
(1871–1945)



Sergei Sobolev (1908–1989)



Laurent Schwartz (1915–2002)

Figure 3.1: Some key figures whose work precipitated modern developments in PDEs.

In applications, the Navier-Stokes equations are primarily solved on a computer, and a computer works with both finite data and finite precision. We will solve these equations by a *Galerkin method*. Galerkin methods are particularly amenable to computer implementation, so much so that the whole process of geometric modelling, boundary condition handling, residual minimisation, and solution reconstruction, can be automated ([41], [20], [19]). We begin by deriving two instances of Galerkin methods for Poisson's equation, performed on an irregular 2D domain to emphasize that the methods trivially extend to complex

geometries. These methods are instances of a *finite volume method* and a *finite element method*. Computer implementations are presented, which generate visualisations and quality metrics for the results.

The Galerkin method

The Galerkin method was established by engineer and mathematician Boris Galerkin (1871–1945) in a 1915 paper containing an approximation method for the biharmonic equation of classical beam theory [39]. Briefly, the principle of Galerkin is to choose a space of approximations, today called "test functions", and solve for the test function which minimises a residual error with respect to some function space norm. This norm is induced by a choice of what are now called the "trial functions". If the trial functions are the same as the test functions, the residual is minimised with respect to the Euclidean norm.

As well as proving highly useful in numerical applications, the methods of Galerkin (and preceding work by mathematicians such as Walther Ritz) proved important to further developments in the mathematical theory of PDEs. A 1934 talk named "Generalized Solutions to the Wave Equation", by mathematician Sergei L. Sobolev (1908–1989), precipitated the development of generalised functions by Laurent Schwartz (1915-2002), integral to the modern study of PDEs ([40], [38]). These developments are clarifications on the question: What do we mean when we speak of a solution to a PDE from physics? Galerkin's method is much closer to the answer than, for example, the finite difference method.

3.2 The equation to solve: Poisson's equation

Among the fundamental PDEs are the heat equation

$$\frac{\partial h}{\partial t} = \Delta h + g$$

and Poisson's equation

$$-\Delta h = g. \tag{3.1}$$

The focus here is on the Dirichlet problem

$$-\Delta h = g, \quad h|_\Gamma = h_\Gamma, \tag{3.2}$$

where Γ is the boundary of the domain, and the domain is 2D. Poisson's equation can be thought of as the steady-state version of the heat equation. The solution of a Poisson problem will be a crucial component of the solution of the Stokes equations. It seems ideal to start by discussing discretization methods for Poisson's equation (3.1) in particular, as it is likely the simplest non-trivial PDE. Poisson's equation (3.1), importantly, is based on an integral conservation law. An integral form will give clearer routes to discretizations which have geometric meaning. For example, the general integral conservation law (1.1),

$$\frac{d}{dt}\int_{\Omega_0} \phi \, dx = \int_{\Omega_0} s \, dx + \oint_{\partial\Omega_0} \phi j \cdot (-\hat{n}) \, dx,$$

is a geometric statement about fluxes of quantity ϕ by j, quantified over arbitrary control volumes Ω_0. There are an infinite number of control volumes, and therefore an infinite number of equations which must hold, and there is an infinite dimensional space of solutions to choose from. A key idea, then, is to choose a finite number of equations and a finite dimensional subspace of possible approximate solutions. A supposed solution will be represented by the coefficients of some choice of basis functions for the subspace. Each equation will be checked exactly against this supposed solution. To continue with this idea, we must write Poisson's equation (3.1) in integral form.

Deriving the heat and Poisson equation through diffusion processes

The Poisson equation (3.1) can be thought of as the steady state of some diffusion process with a source term g, although this need not be its literal physical interpretation. A diffusion process "levels out" some quantity, such as temperature or some chemical concentration. A diffusion could intuitively be thought of as a progressive "blurring", such as in a camera defocus, and in fact many common image processing techniques use diffusion PDEs from physics [17]. We will stick with the notion of "temperature" h as the diffused quantity. *Fick's law of diffusion* is a constitutive relation giving the bulk flux of temperature h as proportional to the negative gradient:

$$hj = -\mu \nabla h,$$

where μ is called the diffusion coefficient. This is one way of saying that the temperature tends to level out. If we form a continuity equation (1.1) for temperature, with source s, we get

$$\frac{d}{dt}\int_{\Omega_0} h\,dx = \int_{\Omega_0} s\,dx + \oint_{\partial\Omega_0} \mu \nabla h \cdot \hat{n}\,dx, \tag{3.3}$$

which by application of Stokes' theorem becomes

$$\frac{dh}{dt} = s + \nabla \cdot (\mu \nabla h)\,. \tag{3.4}$$

If we further assume that the diffusion coefficient μ is constant, we get

$$\frac{dh}{dt} = s + \mu \nabla \cdot \nabla h = s + \mu \Delta h, \tag{3.5}$$

which is the standard heat equation. The steady-state heat equation is then

$$-\Delta h = g, \tag{3.6}$$

where we let $g = s/\mu$ in the above. This completes the derivation of Poisson's equation (3.1). In integral form, "undoing" the application of Stokes' theorem above, Poisson's equation is

$$\oint_{\partial\Omega_0} -\nabla h \cdot \hat{n}\,dx = \int_{\Omega_0} g\,dx \quad \text{for all control volumes } \Omega_0. \tag{3.7}$$

Form (3.7) clearly shows that we are calculating a steady state, as we are solving for h such that the amount of heat that leaves Ω_0 is the amount introduced into Ω_0 by the source function. The form (3.7), rather than (3.1), will be the starting point for deriving Galerkin methods.

So how do we discretize Poisson's equation?

Thinking about the differential equation (3.9) leads to the *finite difference method*, historically the first and still very important in applications.

An aside: The finite difference method.

It is a theorem of Gauss that in Euclidean space $\mathbb{R}^3$ we have

$$\nabla \cdot v = \frac{\partial v_x}{\partial x} + \frac{\partial v_y}{\partial y} + \frac{\partial v_z}{\partial z}, \tag{3.8}$$

and we get (3.1) in the form

$$-\left(\frac{\partial^2 h}{\partial x^2} + \frac{\partial^2 h}{\partial y^2} + \frac{\partial^2 h}{\partial z^2}\right) = g. \tag{3.9}$$

By forming secant approximations of the derivatives over a regular grid, a linear system is formed, and the approximate solution is solved for as a function of this grid. However, it may be hard to give a real interpretation of what the solution samples at grid points say about the solution everywhere. They could be coefficients of, for example, hat basis functions. Yet a finite difference discretization of Poisson equation (3.1) may not take this into account at all.

We will not be using the finite difference method [1]. Starting with the integral form (3.7), there are many routes to take to discretization. Notice that there are two variables in (3.7), although one is bound over universal quantification: the function h and the control volume Ω_0. The general Galerkin method very naturally appears when the "space" Ω_0 resides in and the space h resides in are reduced such that the equations are both finite dimensional and well-determined. Firstly, let's derive the conceptually simplest discretization method of these spaces, the finite volume method.

3.3 Discretizing Poisson's equation by finite volumes

Equation (3.7) is quantified over an infinite number of control volumes Ω_0. These control volumes are in the interior of Ω, as the solution at the boundary Γ is specified by the Dirichlet boundary condition $h|_\Gamma = h_\Gamma$, and there is no flux information across Γ. A simple idea is to choose a finite number of control volumes $\Omega_1, \cdots, \Omega_n$ (hence "finite volumes"), and check that the flux integral holds over each of these. We will then have n equations on h. As this system will be underdetermined (h has infinite degrees of freedom), we must restrict h to a finite dimensional space of approximations Φ^*.

We begin by discussing the general construction of a finite volume method, but this will soon be made concrete with a simple, specific example of a finite volume discretization.

3.3.1 The Dirichlet boundary condition and the test space

The Dirichlet boundary condition $h|_\Gamma = h_\Gamma$ specifies the boundary values for any solution. However, the finite dimensional space Φ^* may not be able to exactly represent the boundary function h_Γ, and therefore the first step is to approximate h_Γ (for which there are many options, for example projection in the L^2-norm or interpolation across a finite sampling of boundary points [22]). Choosing some $\phi_\Gamma \in \Phi^*$ with

$$\phi_\Gamma|_\Gamma \approx h_\Gamma,$$

we can define an "interior" function space

$$\Phi = \{h^* \in \Phi^* \mid h|_\Gamma = 0\},$$

and the approximate solution $\tilde{h}$ will then be

$$\tilde{h} = \phi + \phi_\Gamma$$

for some $\phi \in \Phi$. In the finite volume and finite element literature the space Φ is called the test space. Since we selected n control volumes $\Omega_1, \cdots, \Omega_n$, the approximating space Φ^* must be chosen such that Φ is n-dimensional. We can choose a basis for the test space,

$$\Phi = \operatorname{span}\{\phi_1, \cdots, \phi_n\}.$$

The problem is now to find a vector of coefficients of these test functions, $\hat{h} = (h_1, \cdots, h_n)^T$, and reconstruct the solution as

$$\tilde{h} = \phi + \phi_\Gamma,$$

[1] However, even if the spatial discretization is based on integral conservation laws, typically time-dependent problems are discretized with finite differences in the time dimension.

where ϕ is the "interior variation"

$$\phi = \Phi \cdot \hat{h} \coloneqq h_1\phi_1 + \cdots + h_n\phi_n.$$

3.3.2 Forming a linear system

The integral-conservation-law form (3.7) of Poisson's equation (3.1) now directly indicates a method of approximation. Letting h be approximated by $\tilde{h} = \phi + \phi_\Gamma$ (where $\phi \in \Phi$ is unknown and $\phi_\Gamma \in \Phi^*$ is known), and restricting the flux integrals to the control volumes $\Omega_1, \cdots, \Omega_n$, the conservation law (3.7) becomes

$$\oint_{\partial\Omega_j} -\nabla\left(\phi + \phi_\Gamma\right) \cdot \hat{n}\, dx = \int_{\Omega_j} g\, dx \qquad j = 1, \cdots, n. \tag{3.10}$$

The "interior variation" $\phi = \Phi \cdot \hat{h}$ is the only unknown, so we can move all knowns to the right-hand side and expand ϕ in terms of the basis test functions $\phi_1, \cdots, \phi_n$:

$$\oint_{\partial\Omega_j} -\nabla\left(\sum_{i=1}^{n} h_i\phi_i\right) \cdot \hat{n}\, dx = \int_{\Omega_j} g\, dx + \oint_{\partial\Omega_j} \nabla\phi_\Gamma \cdot \hat{n}\, dx \qquad j = 1, \cdots, n. \tag{3.11}$$

By linearity, to emphasize the separate integrals that need to be computed, the above equation can be written as

$$\sum_{i=1}^{n} h_i \oint_{\partial\Omega_j} -\nabla\phi_i \cdot \hat{n}\, dx = \int_{\Omega_j} g\, dx + \oint_{\partial\Omega_j} \nabla\phi_\Gamma \cdot \hat{n}\, dx \qquad j = 1, \cdots, n. \tag{3.12}$$

It is seen here that there must be some restrictions on the ϕ_i and ϕ_Γ. Formally, they must be in the Sobolev space $H^1(\Omega)$ (— NOTE: This is probably wrong, as the integrals are over internal cell boundaries and the boundary Γ. Need to read about H^{div}) This simply means that they must have a gradient defined "almost everywhere". It does not matter if the gradient is not defined at isolated lower-dimensional subsets, as these make no contribution to the integral. The system (3.12) can be written in matrix form as

$$\begin{aligned} A\hat{h} &= \begin{bmatrix} \oint_{\partial\Omega_1} -\nabla\phi_1 \cdot \hat{n}\, dx & \cdots & \oint_{\partial\Omega_1} -\nabla\phi_n \cdot \hat{n}\, dx \\ \vdots & & \vdots \\ \oint_{\partial\Omega_n} -\nabla\phi_1 \cdot \hat{n}\, dx & \cdots & \oint_{\partial\Omega_n} -\nabla\phi_n \cdot \hat{n}\, dx \end{bmatrix} \begin{bmatrix} h_1 \\ \vdots \\ h_n \end{bmatrix} \\ &= \begin{bmatrix} \int_{\Omega_1} g\, dx + \oint_{\partial\Omega_1} \nabla\phi_\Gamma \cdot \hat{n}\, dx \\ \vdots \\ \int_{\Omega_n} g\, dx + \oint_{\partial\Omega_n} \nabla\phi_\Gamma \cdot \hat{n}\, dx \end{bmatrix} = \hat{f}. \end{aligned} \tag{3.13}$$

This system is solved for the coefficients of combination $\hat{h} = (h_1 \cdots h_n)^T$, and the approximate solution is constructed as $\tilde{h} = \Phi \cdot \hat{h} + \phi_\Gamma$. If the control volumes $\Omega_1, \cdots, \Omega_n$ and test space $\Phi = \text{span}\{\phi_1, \cdots, \phi_n\}$ are chosen well, this linear system will be nonsingular and hopefully well-conditioned. If the chosen control volumes join together in the interior of Ω like pieces of a puzzle, this gives a conservative system of balanced fluxes, but it is another question whether the approximation $\tilde{h}$ is good.

3.3.3 A piecewise linear finite volume triangulation scheme

Possibly the simplest scheme is to triangulate Ω as $\bigcup_i T_i$, such that there are n interior vertices $p_1, \cdots, p_n$, and n_Γ boundary vertices $p_1^\Gamma, \cdots, p_{n_\Gamma}^\Gamma$, and n_T triangles $T_1, \cdots, T_{n_T}$. An example triangulation of an ellipse-shaped domain is shown in figure 3.2.

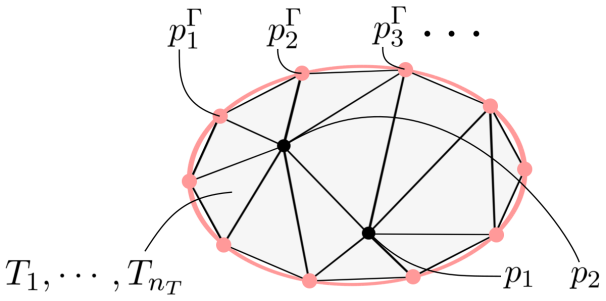


Figure 3.2: The domain Ω is partitioned into $n_T = 12$ triangular cells $T_1 \cdots T_{12}$. The boundary is sampled at $n_\Gamma = 10$ points $p_1^\Gamma, \cdots, p_{10}^\Gamma$. There are $n = 2$ interior points, p_1 and p_2.

The test space

Firstly, there is a clear choice of function space Φ^*, which satisfies the H^1 condition (NOTE: Is H^1 necessary? Need to check conditions for finite volumes), consisting of piecewise linear "hat" basis functions at each interior or boundary vertex,

$$\Phi^* = \text{span}\left\{\text{Hat}(p_1), \cdots, \text{Hat}(p_n), \text{Hat}(p_1^\Gamma), \cdots, \text{Hat}(p_{n_\Gamma}^\Gamma)\right\}.$$

The hat function at a vertex has value 1 at that vertex and value 0 at its neighbours. The test space (which is zero on the boundary Γ) is then

$$\Phi = \text{span}\left\{\text{Hat}(p_1), \cdots, \text{Hat}(p_n)\right\} = \text{span}\left\{\phi_1, \cdots, \phi_n\right\},$$

where $\phi_i \coloneqq \text{Hat}(p_i)$. One of these basis functions for the ellipse-shaped domain is shown in figure 3.3.

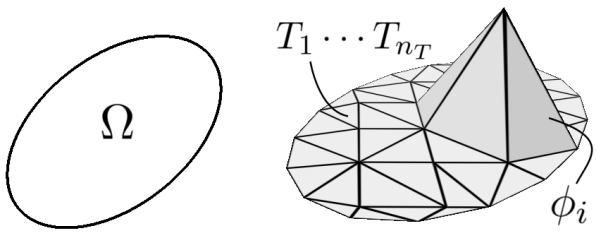


Figure 3.3: The domain Ω is partitioned into triangular cells $T_1 \cdots T_{n_T}$. One example of a hat basis function ϕ_i is shown, where the vertical axis is the basis function value.

Approximating the boundary values

One simple approximation of $h|_\Gamma = h_\Gamma$ is

$$\phi_\Gamma = h_\Gamma(p_1^\Gamma)\mathrm{Hat}(p_1^\Gamma) + \cdots + h_\Gamma(p_{n_\Gamma}^\Gamma)\mathrm{Hat}(p_{n_\Gamma}^\Gamma), \tag{3.14}$$

which is a piecewise linear interpolation when restricted to the boundary Γ. The approximation ϕ_Γ for some boundary function h_Γ is shown in figure 3.4.

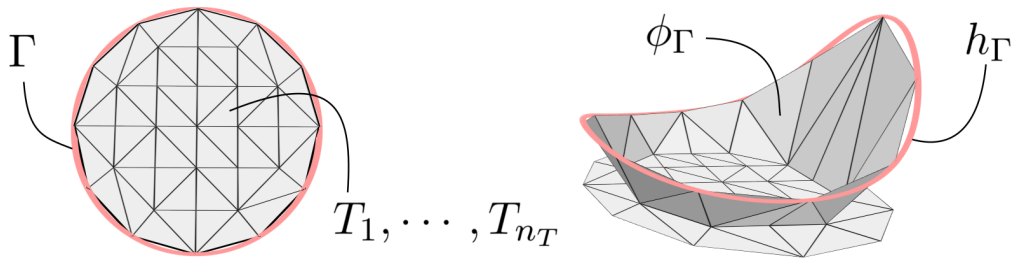


Figure 3.4: A circle domain Ω with boundary Γ is tessellated by triangles $T_1, \cdots, T_{n_T}$. The boundary function h_Γ is approximated by ϕ_Γ, a linear combination of hat functions centred at the boundary vertices.

This approximation ϕ_Γ is added to the "interior variation" $\phi = \Phi \cdot \hat{h}$ to construct a possible solution. When a linear solver is given the linear system (3.13), it will find the interior variation ϕ which minimises some residual norm. An iterative solver, for example, can then be thought of as progressively varying a thin membrane attached to the boundary curve, until the residual error is sufficiently small. This construction is shown in figure 3.5.

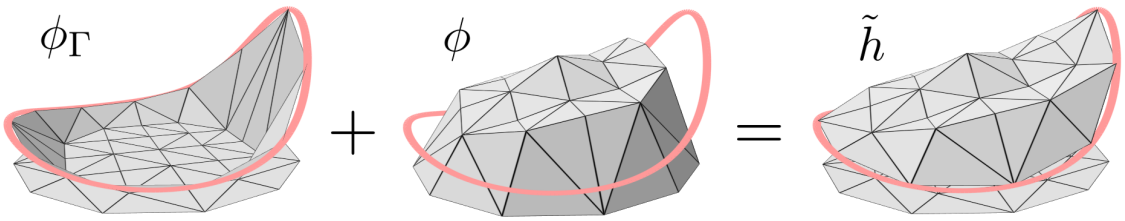


Figure 3.5: The fixed boundary approximation ϕ_Γ is added to a variable interior variation ϕ to give a possible approximate solution $\tilde{h}$. This possible solution can be checked by integration against the trial control volumes $\Omega_1, \cdots, \Omega_n$.

Choosing a finite number of control volumes

If we choose some characteristic "triangle centre" for each triangle, then we can associate to the interior vertices $p_1, \cdots, p_n$ a set of domains $\Omega_1, \cdots, \Omega_n$. Each Ω_i is determined by the polygon which joins the centres of the triangles incident to p_i. Two common choices for the triangle centre are the barycentre, which is the average position of the three vertices, and the circumcentre, which is the centre of the unique circle passing through the three vertices. The circumcentre scheme gives what are called "Voronoi cells" due to their relation to Voronoi diagrams in computational geometry [31]. The resulting cell decompositions are displayed in figure 3.6.

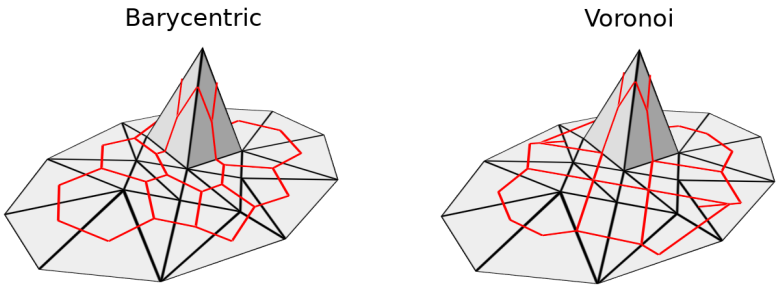


Figure 3.6: The domain Ω is partitioned into triangular cells $T_1 \cdots T_{n_T}$. Flux integrals are taken over n polygonal cells, one for each internal vertex p_i, for example using triangle barycentres or circumcentres.

Notice that, in the circumcentre case, the thin triangles do not contain their circumcentre, and therefore each cell boundary is not necessarily contained in a single fan of triangles. Notice also that the Voronoi polygon passes through the midpoints of each edge incident to p_i. This midpoint property will soon prove useful in computing integrals, but the resulting exposition will be clearer if we can also assume that the cell boundary *is* contained in a single fan of triangles. Therefore we define the *mixed Voronoi cell* [14], where a wayward circumcentre is pulled back to the opposite edge.

(draw this)

The scheme as described so far, consisting of a triangle mesh, hat test functions, and barycentric, Voronoi, or mixed Voronoi control volumes, has found some success, especially in the domain of geometry processing ([14], [15]). We will work with the mixed Voronoi cells.

3.3.4 Computing the linear system by closed form integration

Now that we have chosen a specific finite volume discretization, given a domain triangulation we can compute the coefficients of the linear system (3.13) and solve for $\hat{h}$, giving the approximate solution $\tilde{h} = \phi_\Gamma + \Phi \cdot \hat{h}$. By fortuitous circumstances, these integrals actually have simple closed forms. (This is with the exception of the heat source integrals $\int_{\Omega_j} g\, dx$, which may require numerical integration if the heat source g is not restricted to Φ^*.)

It will be more convenient to start again with the trial flux integral

$$\oint_{\partial\Omega_j} -\nabla\tilde{h} \cdot \hat{n}\, dx, \tag{3.15}$$

rather than with the expanded linear system (3.13). By linearity, the ∇ in $\nabla\tilde{h}$ "falls through" the expansion of $\tilde{h}$:

$$\nabla\tilde{h} = \nabla\phi_\Gamma + \nabla\phi = \sum_{i=1}^{n_\Gamma} h_\Gamma(p_i^\Gamma)\nabla\phi_i^\Gamma + \sum_{i=1}^{n} h_i \nabla\phi_i. \tag{3.16}$$

By linearity the integral (3.15) becomes

$$\sum_{i=1}^{n_\Gamma} \left[h_\Gamma(p_i^\Gamma) \oint_{\partial\Omega_j} -\nabla\phi_i^\Gamma \cdot \hat{n}\, dx \right] + \sum_{i=1}^{n} \left[h_i \oint_{\partial\Omega_j} -\nabla\phi_i \cdot \hat{n}\, dx \right]. \tag{3.17}$$

Now consider a certain flux integral (3.15) over control volume Ω_j. We will reexpress the above integral in "local" terms.

Some local indexing definitions.

Let $v = p_j$ be the interior vertex corresponding to the control volume Ω_j, and denote by $v_1, \cdots, v_k$ its adjacent vertices (either in the interior or on the boundary), and by $T_1^v \cdots T_k^v$ its adjacent triangles. Triangle T_l^v joins v, v_l, v_{l+1} where addition wraps around from k to 1. As it will be necessary to relate these local indices (denoted l) to the global vertex indices (denoted i), define an index map by

$$I(l) = \text{the global index } i \text{ of the } p_i \text{ corresponding to adjacent vertex } v_l,$$

which is valid only when v_l is an interior vertex.

The vertex v corresponds to a single trial flux integral over the surrounding Voronoi cell Ω_j. As the hat functions have compact support, Ω_j only intersects the domains of the hat function ϕ_j and those corresponding to the adjacent vertices. Name these basis functions $\phi_j = \phi_v$ and $\phi_{v_1}, \cdots \phi_{v_k}$. Then the local form of the flux integral (3.15) is

$$\oint_{\partial\Omega_j} -\nabla\tilde{h} \cdot \hat{n}\, dx = h_j \oint_{\partial\Omega_j} -\nabla\phi_v \cdot \hat{n}\, dx + \sum_{l=1}^{k} h_{v_l} \oint_{\partial\Omega_j} -\nabla\phi_l^v \cdot \hat{n}\, dx, \tag{3.18}$$

where h_{v_i} is defined as

$$h_{v_i} = \begin{cases} h_\Gamma(p_i^\Gamma) & \text{if } v_i \text{ is a boundary vertex,} \\ h_{I(v_i)} & \text{if } v_i \text{ is an interior vertex.} \end{cases}$$

The ϕ_v and $\phi_{v_1}, \cdots, \phi_{v_k}$ are hat functions which are linear when restricted to each triangle $T_1^v, \cdots, T_k^v$. Therefore the gradients $\nabla\phi_v$ and $\nabla\phi_{v_1}, \cdots, \nabla\phi_{v_k}$ are constant on those triangles. We can break the closed flux integrals in (3.18) into flux integrals restricted to each triangle adjacent to v, giving

$$\oint_{\partial\Omega_j} -\nabla\tilde{h} \cdot \hat{n}\, dx = \sum_{t=1}^{k} \left[\int_{\Omega_j \cap T_t^v} -\nabla\phi_v \cdot \hat{n}\, dx + \sum_{l=1}^{k} h_{v_l} \int_{\Omega_j \cap T_t^v} -\nabla\phi_l^v \cdot \hat{n}\, dx \right]. \tag{3.19}$$

We can simplify matters by focusing on just one adjacent triangle $T_t^v = T$, between vertices v, v_t, v_{t+1}. This triangle T only overlaps with the domains of the hat functions centred at v, v_t, v_{t+1}, so we now only need to compute

$$\int_{\Omega_j \cap T} -\nabla\phi_v \cdot \hat{n}\, dx \quad \text{and} \quad \int_{\Omega_j \cap T} -\nabla\phi_{v_t} \cdot \hat{n}\, dx \quad \text{and} \quad \int_{\Omega_j \cap T} -\nabla\phi_{v_{t+1}} \cdot \hat{n}\, dx.$$

This is simple because the gradients are constant on $\Omega_j \cap T_t^v$.

(figure)

The Voronoi polygon joins the midpoints $m' = \frac{1}{2}(v + v_t)$ and $m'' = \frac{1}{2}(v + v_{t+1})$, taking a detour to a point within T.

(figure)

However, since the gradient vector fields are constant, they are conservative, and we can simplify the domain $\Omega_j \cap T$ to the line segment L between the midpoints m' and m''. Since the gradients are constant on T, these integrals are the length of L multiplied by one "sample" flux across L. This can be done easily, by noticing L is parallel with and half the length of the line segment between v_t and v_{t+1}. We can then compute the integrals as:

$$\frac{1}{2}\nabla\phi_v \cdot (v_{t+1} - v_t)^\perp, \quad \text{and} \quad \frac{1}{2}\nabla\phi_{v_t} \cdot (v_{t+1} - v_t)^\perp, \quad \text{and} \quad \frac{1}{2}\nabla\phi_{v_{t+1}} \cdot (v_{t+1} - v_t)^\perp, \tag{3.20}$$

where $(v_{t+1} - v_t)^\perp$ is the vector $v_{t+1} - v_t$ rotated 90 degrees anticlockwise. We now only need the constant gradients on T. This requires some simple geometry. As this construction is the same for each vertex, consider an anticlockwise ordering of v, v_t, v_{t+1}, named v', v'', v'''.

(figure of a rectangle drawn around the triangle, and the perpendicular direction, with width and height labels)

We must normalize this vector and then divide by the height of this rectangle, as the slope is inversely proportional to the rectangle height. This gives the gradient

$$\nabla\phi_{v'} = \frac{(v''' - v'')^\perp}{\|v''' - v''\|} \Big/ h = \frac{(v''' - v'')^\perp}{wh}.$$

Finally, $\text{Area}(T) = wh/2$, so we can write the gradients as

$$\nabla\phi_v = \frac{(v_{t+1} - v_t)^\perp}{2\text{Area}(T)}, \quad \text{and} \quad \nabla\phi_{v_t} = \frac{(v - v_{t+1})^\perp}{2\text{Area}(T)}, \quad \text{and} \quad \nabla\phi_{v_{t+1}} = \frac{(v_t - v)^\perp}{2\text{Area}(T)}. \tag{3.21}$$

Plugging the gradients (3.21) into the closed form integrals (3.20), we get

$$\begin{aligned}
\int_L -\nabla\phi_v \cdot \hat{n}\, dx &= \frac{1}{4\text{Area}(T)}\|v_{t+1} - v_t\|^2, \\
\int_L -\nabla\phi_{v_t} \cdot \hat{n}\, dx &= \frac{1}{4\text{Area}(T)}(v - v_{t+1}) \cdot (v_{t+1} - v_t), \\
\int_L -\nabla\phi_{v_{t+1}} \cdot \hat{n}\, dx &= \frac{1}{4\text{Area}(T)}(v_t - v) \cdot (v_{t+1} - v_t),
\end{aligned} \tag{3.22}$$

where the 90-degree rotation has been removed, as it preserves the dot product. With these closed forms, we can express the original integral as

$$\begin{aligned}
&\oint_{\partial\Omega_j} -\nabla\tilde{h} \cdot \hat{n}\, dx \\
&= \sum_{t=1}^{k} \frac{1}{4\text{Area}(T_t^v)} \left[h_j\|v_{t+1} - v_t\|^2 + h_{v_t}(v - v_{t+1}) \cdot (v_{t+1} - v_t) + h_{v_{t+1}}(v_t - v) \cdot (v_{t+1} - v_t) \right].
\end{aligned} \tag{3.23}$$

This directly indicates an effective algorithm for constructing the linear system.

Integrating the source function

One more thing is needed, however. The right-hand side vector of (3.13) contains source integrals

$$\int_{\Omega_j} g\, dx,$$

so we require some numerical integration. The simplest choice is a "rectangle rule", where we sample the source function g at the interior vertex p_j, then assume that g is constant over Ω_j. This gives the approximation

$$\int_{\Omega_j} g\, dx \approx g(p_j)\text{Area}(\Omega_j). \tag{3.24}$$

The computation of the mixed Voronoi cell area $\text{Area}(\Omega_j)$ can be achieved with some routine geometric calculations. Iterating over each triangle incident to p_j, compute the

cell segment area as the area of two triangles, the second triangle depending on whether the computed circumcentre lies inside or outside the triangle.

(draw this)

3.3.5 The resulting matrix assembly algorithm

Using the expression (3.23), the linear system (3.13) can be constructed by iterating over each interior vertex v, then each adjacent triangle, and computing these closed form integrals. Each adjacent triangle corresponds to two vertices incident to v, and for each of these vertices, either a value is placed in either the system matrix or the right-hand side is adjusted, depending on whether the triangle vertex is in the interior or on the boundary. Also, while iterating over the interior vertices, the corresponding source integral is approximated by (3.24), and the result is added to the right-hand side vector. Pseudocode for this algorithm is given below.

Routine *Poisson_FVM_Matrix_Assembly*:

Data: Domain Ω with boundary Γ.
Heat source function $g : \Omega \to \mathbb{R}$.
Dirichlet boundary function $h_\Gamma : \Gamma \to \mathbb{R}$.
Triangulation of Ω consisting of:
Interior vertices $p_1, \cdots, p_n$.
Boundary vertices $p_1^\Gamma, \cdots, p_{n_\Gamma}^\Gamma$.
Triangles $T_1, \cdots, T_{n_T}$.

Result: The coefficients $(A, \hat{f})$ of the system $A\hat{h} = \hat{f}$ (3.13), which can be solved by the caller for $\hat{h}$ in order to reconstruct an approximate solution $\tilde{h} : \Omega \to \mathbb{R}$ of the boundary value problem (3.2),

$$-\Delta h = g,\ h|_\Gamma = h_\Gamma.$$

$A \leftarrow n \times n$ zero matrix;
$\hat{f} \leftarrow$ length n zero vector;
for $j = 1, \cdots, n$ **do**
  Let v denote the interior vertex p_j.
  Let $v_1, \cdots, v_k$ denote the vertices adjacent to v.
  Let $T_1^v, \cdots, T_k^v$ denote the triangles adjacent to v.
  (Triangle T_t joins vertices v, v_t, v_{t+1}.)
  for $t = 1, \cdots, k$ **do**
    $C \leftarrow \frac{1}{4\text{Area}(T)}$;
    $A[j,j] \leftarrow A[j,j] + C\|v_{t+1} - v_t\|^2$;
    if v_t *is on the boundary* **then**
      $\hat{f}[I(t)] \leftarrow \hat{f}[I(t)] - h_\Gamma(v_t)C(v - v_{t+1}) \cdot (v_{t+1} - v_t)$;
    else
      $A[j, I(t)] \leftarrow A[j, I(t)] + C(v - v_{t+1}) \cdot (v_{t+1} - v_t)$;
    end
    if v_{t+1} *is on the boundary* **then**
      $\hat{f}[I(t+1)] \leftarrow \hat{f}[I(t+1)] - h_\Gamma(v_{t+1})C(v_t - v) \cdot (v_{t+1} - v_t)$;
    else
      $A[j, I(t+1)] \leftarrow A[j, I(t+1)] + C(v_t - v) \cdot (v_{t+1} - v_t)$;
    end
  end
  $\hat{f}[j] \leftarrow \hat{f}[j] + \text{Control\_Volume\_Area}(\Omega_j) \cdot g(v)$;
end
return $(A, \hat{f})$;

Algorithm 1: Pseudocode for the described finite volume matrix assembly process for the Poisson boundary value problem (3.2).

3.3.6 The resulting finite volume algorithm

We now have an effective algorithm for approximating Poisson's equation:

1. Partition the 2D domain Ω into triangles $T_1, \cdots, T_{n_T}$. This determines the hat basis functions $\phi_1, \cdots, \phi_n$ and the mixed Voronoi cells $\Omega_1, \cdots, \Omega_n$.

2. Form the matrix and right-hand side in (3.13) with the algorithm **??**.

3. Solve the resulting linear system for $\hat{h}$, and construct the solution as $\tilde{h} = \phi_\Gamma + \Phi \cdot \hat{h}$.

Each of these steps is conceptually well-separated into the domains of mesh generation, matrix assembly, and numerical linear algebra. The above pseudocode **??** is a matrix assembly algorithm. It assumes that there is already a domain triangulation, and constructs the linear system (3.13). Although we have addressed the problem of matrix assembly, specific to this particular finite volume method, mesh generation and numerical linear algebra can be delegated to external libraries. The full solve algorithm is given below.

Routine *Solve_Poisson_FVM*:
Data: Domain Ω with boundary Γ.
Heat source function $g : \Omega \to \mathbb{R}$.
Dirichlet boundary function $h_\Gamma : \Gamma \to \mathbb{R}$.
Result: Approximate solution $\tilde{h} : \Omega \to \mathbb{R}$ of the boundary value problem (3.2),
$$-\Delta h = g,\ h|_\Gamma = h_\Gamma.$$
// Mesh generation
triangulation $\leftarrow$ Generate_Triangulation(Ω);
// Matrix assembly
$(A, \hat{f}) \leftarrow$ Poisson_FVM_Matrix_Assembly(Ω, g, h_Γ, triangulation);
// Linear solve
Solve $A\hat{h} = \hat{f}$ for $\hat{h}$.
// Solution reconstruction
$\tilde{h} \leftarrow \sum_{j=1}^{n} h_j \mathrm{Hat}(p_j) + \sum_{j=1}^{n_\Gamma} h_\Gamma(p_j^\Gamma)\mathrm{Hat}(p_j^\Gamma)$;
return $\tilde{h}$;

The routine **Generate_Triangulation** is provided by a mesh generator, and the **Solve** routine is provided by a numerical linear algebra library.

Mesh generation

Mesh generation is a huge field. However, there are only two essentials for the handling of 2D domains and piecewise linear triangulations — a mesh data structure and a triangulator. Typically a mesh data structure will be provided by a separate library, and the data organization (e.g., vertex, triangle, and adjacency information) will be designed to facilitate the kinds of mesh traversals performed during matrix assembly. This is typically some variant of a "halfedge" data structure [14], implemented in libraries such as Geometry Central [35], the Polygon Mesh Processing library [36], and OpenMesh [37].

A mesh data structure is a foundational component of mesh generation systems. For 2D domains, a somewhat regular sampling of points on the boundary and interior, followed by triangulation of these points, can suffice for a piecewise-linear finite volume mesh. The Triangle [29] library, for example, contains a single very efficient and robust C routine (consisting of 16k lines of highly optimized code) for performing Delaunay triangulations [31] on 2D domains, with the specific goal of creating robust finite element meshes.

Numerical linear algebra

The solution of large linear systems is a vast topic. PDE solvers are typically clients of standard, robust linear and non-linear solver libraries, such as Argonne National Laboratory's PETSc libraries [32] and the smaller-scale C++ libraries Armadillo [33] and Eigen [34]. A PDE solver will typically pass either a full matrix to the linear solver (dense or sparse), or provide the linear solver with callback routines that give it access to the system coefficients when they are needed.

— Discuss sparsity here.

3.3.7 An implementation in C++

I have implemented algorithm **??** in C++. This code uses a simple mesh generator based on the C library Triangle [29]. Triangle is given a list of points and boundary points, and generates a Delaunay triangulation [31]. This triangulation is then converted to a halfedge mesh data structure based on the ideas in the Geometry Central library [35]. The matrix is constructed as a flat array of indices and coefficients, which are then converted to CRS (compressed row storage) form. The system is solved with a sparse LU factorisation method provided by the C++ Eigen library [34]. With these interfaces, the resulting C++ was almost a direct transliteration of the pseudocodes **??** and **??**.

The implemented matrix assembly algorithm has linear space and time complexity in the number of mesh vertices, and all data is generated and accessed in flat buffers, giving good cache performance. The only superlinear stage is the system solve, which is handled by Eigen [34]. All computations have been performed with double precision floating point.

3.3.8 Validating the method

In order to validate this algorithm, and in particular its C++ implementation described above, some simple test cases can be generated, using the "method of manufactured solutions" [21]. Firstly, suppose the exact solution is

$$h = x^2 - y^2.$$

From the chosen solution h, we can derive the boundary condition and heat source that would give that solution. Of course, the boundary condition must be $h_\Gamma = h|_\Gamma$. We can solve for the source term g by plugging h into Poisson's equation (3.1), giving

$$-\Delta\left(x^2 - y^2\right) = g \quad \Rightarrow \quad g = -(2-2) = 0.$$

This test case has been chosen as it degenerates to an instance of the simpler Laplace's equation, where there is no heat source:

$$-\Delta h = 0, \quad h|_\Gamma = x^2 - y^2 \quad \text{(The test problem)}. \tag{3.25}$$

For clarity, we use a simple square domain $\Omega = [-1, 1]^2$, and a uniform grid parameterized by value r, which measures the average length of a triangle edge. [2]. Each square in this grid corresponds to two right-angled triangles in the triangulation [3]. To measure convergence, we use the L^2-norm (although others could be used), defined by

$$\left\|\tilde{h} - h\right\|_{L^2(\Omega)} = \sqrt{\int_{-1}^{1}\int_{-1}^{1} (\tilde{h}(x,y) - h(x,y))^2 \, dx \, dy}. \tag{3.26}$$

[2] Traditionally this kind of parameter is denoted h, which we are already using for the heat function.

[3] This is a rather uninteresting mesh which doesn't take advantage of Galerkin methods' ability to model complex geometries, but it is a difficult problem to parameterise a complex mesh by some r such that the error metric acts "nicely".

To validate the convergence of the method, simply decrease r, generate the mesh, solve the test equation, compute the L^2 error (3.26), and repeat. A log-log plot of the error for the simple test problem is shown in figure 3.7. It can be seen clearly that the convergence for test problem (3.25) is quadratic.

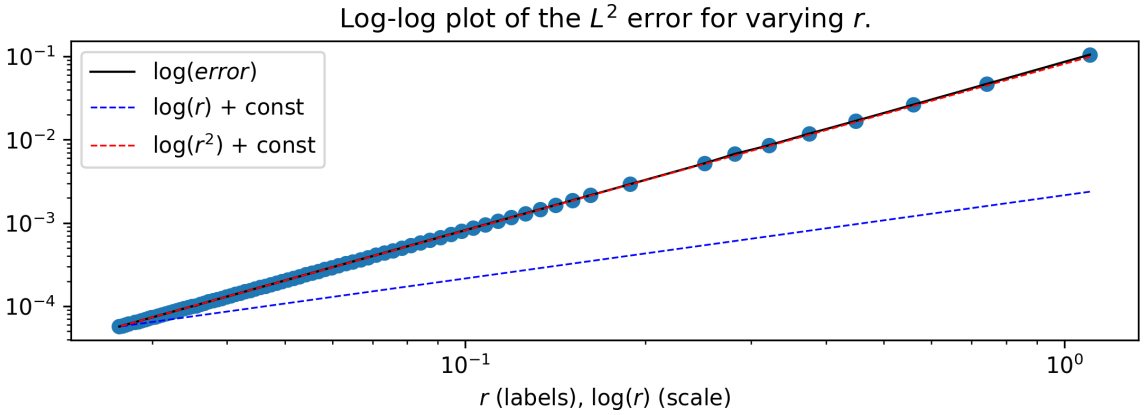


Figure 3.7: The L^2-norm error for the test problem (3.25), as parameter r varies.

We should also try a test problem which does not degenerate to Laplace's equation. Suppose instead that the exact solution is

$$h = e^{-2(x^2+y^2)}.$$

The source term g is solved for as

$$-\Delta\left(e^{-2(x^2+y^2)}\right) = g \quad \Rightarrow \quad g = \left(8 - 16x^2 - 16y^2\right) e^{-2(x^2+y^2)},$$

giving the new test problem,

$$-\Delta h = \left(8 - 16x^2 - 16y^2\right) e^{-2(x^2+y^2)}, \quad h|_\Gamma = e^{-2(x^2+y^2)} \quad \text{(The test problem)}. \quad (3.27)$$

The log-log error plot is displayed in figure (3.8). In this case, there is still a clear quadratic convergence, but with some oscillation in quality as r decreases. This is likely due to the coarse numerical integration of the heat source (which is using the rectangle rule for simplicity). A visualization of these convergence results for problem (3.27) is given in figure 3.9. To emphasize the generality of the algorithm, figure 3.10 displays convergence for problem (3.25) on an irregularly-meshed disk.

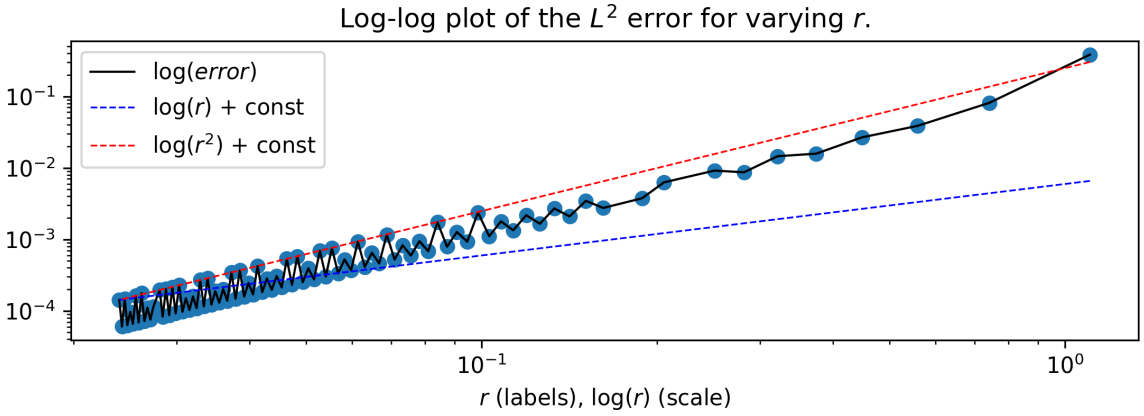


Figure 3.8: The L^2-norm error for the test problem (3.27), as parameter r varies.

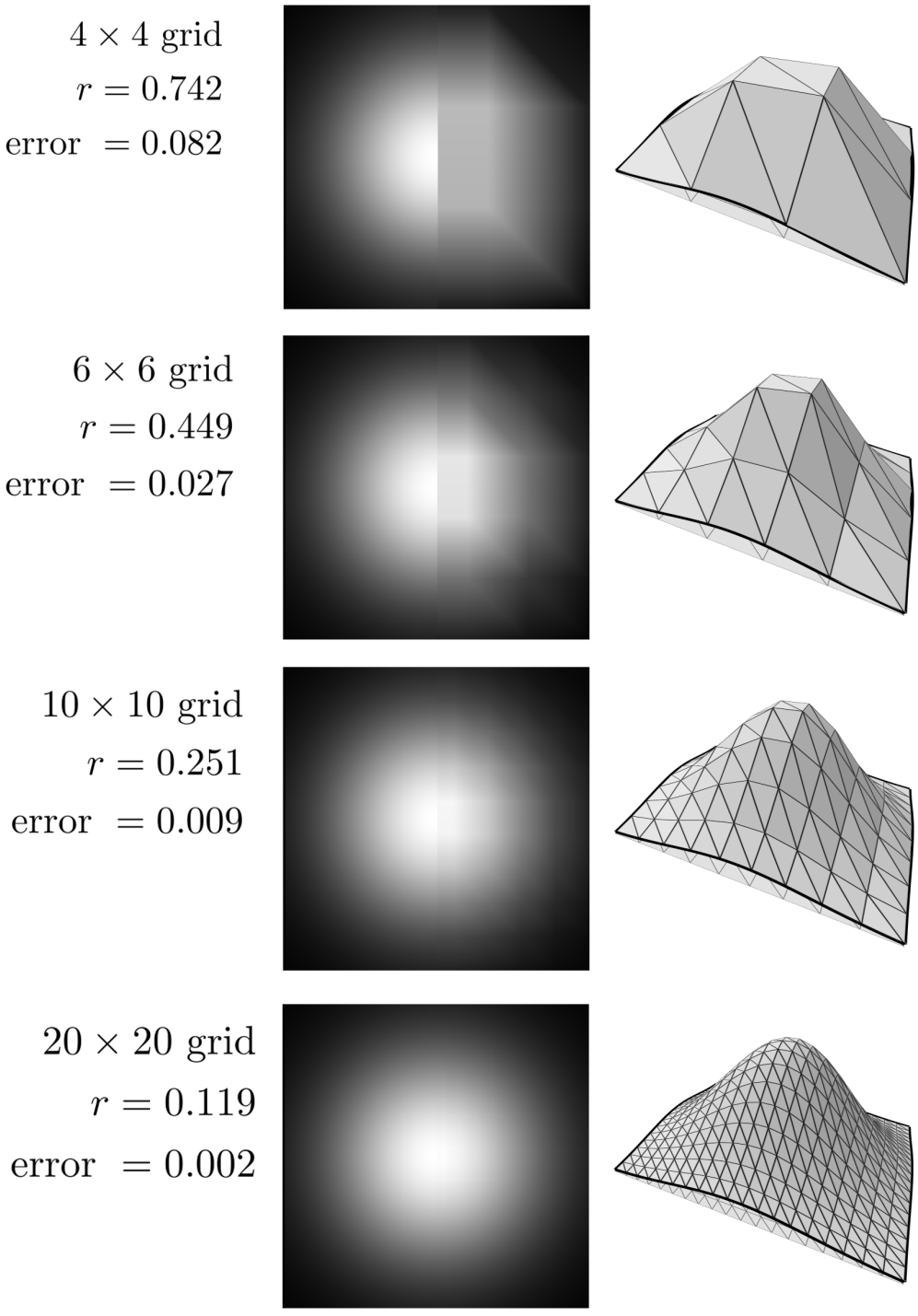


Figure 3.9: Approximate solutions to the test problem (3.27) with varying r. The middle squares contain plots of the exact solution (on the left rectangle), and the approximate solution (on the right rectangle). Black is $h = 0$, and white is $h = 1$. On the right, a 3D view of the solution and mesh is shown.

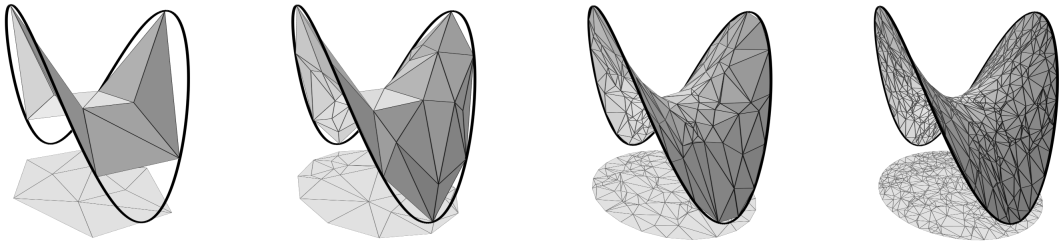


Figure 3.10: Approximate solutions to the test problem (3.25) with varying r. The domain is a unit disk, and r is reduced by introducing random points. The finite volume algorithm **??** works trivially with curved domains and irregular meshes.

3.4 From finite volumes to general Galerkin methods

We will soon derive another instance of a Galerkin method, called a *finite element method*. This approach is directly related to the "finite volumes" described above. The finite element method, however, requires a construction called the *weak form* of a PDE, which can be directly motivated from the preceding discussion on finite volumes. While each finite volume has a clear geometric meaning (as a small cell around which a certain test flux integral is taken), the geometric meaning of a "finite element" requires slightly more thought, although it will be seen to be the same idea in disguise — the resulting linear system will be formed from averaged, or "blurred" flux integrals, rather than single flux integrals.

3.4.1 Generalising the flux integrals by combining equations

The matrix equation (3.13) consists of linear equations

$$\oint_{\partial\Omega_1} -\nabla\left(\sum_{i=1}^{n} h_i\phi_i\right)\cdot\hat{n}\,dx = \int_{\Omega_1} g\,dx \quad (\text{for } j=1),$$

and so on. We cannot compute flux integrals over all arbitrary control volumes, but we can take a number of "trial" flux integrals over the finite number of cells $\Omega_1,\cdots,\Omega_n$. As with any linear system, we can take linear combinations of these equations to get more equations which must hold on a solution. For example,

$$5\left[\oint_{\partial\Omega_1} -\nabla\left(\sum_{i=1}^{n} h_i\phi_i\right)\cdot\hat{n}\,dx\right]-2\left[\oint_{\partial\Omega_2} -\nabla\left(\sum_{i=1}^{n} h_i\phi_i\right)\cdot\hat{n}\,dx\right]=5\left[\int_{\Omega_1} g\,dx\right]-2\left[\int_{\Omega_2} g\,dx\right] \tag{3.28}$$

must hold for a solution $\hat{h}$ to (3.13). Equation (3.28) is not a flux integral, but a sum of flux integrals. Now suppose instead we are working with the original integral form of Poisson's equation, (3.7):

$$\oint_{\partial\Omega_0} -\nabla h\cdot\hat{n}\,dx = \int_{\Omega_0} g\,dx \quad \text{for all control volumes } \Omega_0.$$

Then, in the same way, we can take linear combinations of the equations (3.7) for some collection of control volumes, to get another equation which the exact solution h must satisfy. In addition we can take integral combinations, whenever that integral is well-defined. For example, consider a parameterised family of control volumes, $\{\Omega_{0,t} \mid 0<t<1\}$, defined by

$$\Omega_{0,t} = \text{the disk of radius } t \text{ around some fixed point } c.$$

Then the equation

$$\int_{t=0}^{t=1} \left[\oint_{\partial\Omega_{0,t}} -\nabla h \cdot \hat{n}\, dx \right] dt = \int_{t=0}^{t=1} \left[\int_{\Omega_{0,t}} g\, dx \right] dt \tag{3.29}$$

must also hold for the exact solution h. We therefore have an alternative route to discretization of the integral equation (3.7): instead of choosing a finite number of flux equations over single control volumes, we choose a finite number of integrals of flux equations over families of control volumes.

3.4.2 The weak form of Poisson's equation

The averaged flux integral equation (3.29), expressed as a nested integral, can be restated as a single integral over the whole domain:

$$\int_\Omega \nabla h \cdot \nabla v\, dx = \int_\Omega gv\, dx, \tag{3.30}$$

where v is a scalar function defined as

$$v(x) = \begin{cases} 1-t & \text{if } x \in \Omega_{0,t} \\ 0 & \text{otherwise.} \end{cases} \tag{3.31}$$

This deserves some explanation, which will be given at two levels of informality. The concentric family of control volumes $\Omega_{0,t}$ appear as level sets $v = 1 - t$. Since this cone has unit slope, the normal $\hat{n}$ of the level set $v = 1 - t$ is $-\nabla v$. Since the left-hand side of (3.30) is an integral over the fluxes through each level set, the nested integral is subsumed into one integral over the union of level sets. The outer integral of the right-hand side of (3.30) consists of a "stack of integrations" of g against concentric disks, which is equivalent to integrating g against a "stack of concentric disks", or rather, the cone v (3.31).

(draw this)

At another level of informality, with more notation, we can define the indicator function

$$\chi(\Omega_{0,t})(x) = \begin{cases} 1 & \text{if } x \in \Omega_{0,t} \\ 0 & \text{otherwise,} \end{cases} \tag{3.32}$$

and define the gradient of the indicator function by

$$\nabla\chi(\Omega_{0,t})(x) = \begin{cases} -\hat{n} \cdot \infty & \text{if } x \in \partial\Omega_{0,t} \\ 0 & \text{otherwise.} \end{cases} \tag{3.33}$$

The ∞ is an informal way of stating that a 2D integral will give a non-zero value on this boundary, which is a set of measure zero — with typical functions, any isolated set of measure zero is ignored [4]. If there should be any definition of a "generalised gradient", this seems reasonable — if the indicator function $\chi(\Omega_{0,t})$ is approximated by a sequence of smooth functions $f_1, f_2, \ldots$, the gradient ∇f_i will approach (3.33).

(draw this)

[4] This idea (and the related Dirac delta "function") is formalised in the theory of distributions, or generalised functions, introduced by Laurent Schwartz in the late 1940s.

We then have the equalities

$$\int_{t=0}^{t=1} \left[\oint_{\partial\Omega_{0,t}} -\nabla h \cdot \hat{n}\, dx \right] dt = \int_{t=0}^{t=1} \left[\int_{\Omega} \nabla h \cdot \nabla \chi(\Omega_{0,t})\, dx \right] dt$$
$$= \int_{\Omega} \nabla h \cdot \nabla \left[\int_{t=0}^{t=1} \chi(\Omega_{0,t})\, dt \right] dx,$$

where the final equality might assume some regularity (— Fubini's theorem?). We then define

$$v \coloneqq \int_{t=0}^{t=1} \chi(\Omega_{0,t})\, dt, \tag{3.34}$$

which is simplified to (3.31). Note that ∇v is not defined everywhere (∇v does not exist at c or on the unit circle around c). However, this is fine, as these are isolated sets of measure zero, which can be ignored as making no contribution to the integral. Formally, v is in the Sobolev space $H_0^1(\Omega)$, where the 0 subscript denotes that v must vanish on the boundary Γ. (The integral of fluxes cannot contain a flux measurement across Γ, as the solution there is specified by the Dirichlet boundary condition — there is no flux information on Γ.)

Now we can forget our specific definition of v, and define the *weak form* of Poisson's equation to be

$$\int_{\Omega} \nabla h \cdot \nabla v\, dx = \int_{\Omega} gv\, dx \quad \text{for all } v \in H_0^1(\Omega). \tag{3.35}$$

The weak form will be the starting point for deriving general Galerkin methods.

3.4.3 General Galerkin methods for Poisson's equation

A general Galerkin method will discretize the test space as usual, approximating h by

$$\tilde{h} \in \Phi = \operatorname{span}\{\phi_1, \cdots, \phi_n\},$$

and will also approximate the "trial space" $H_0^1(\Omega)$ [5] as

$$v \in \Psi = \operatorname{span}\{\psi_1, \cdots, \psi_n\}.$$

Since the weak form equations (3.35) are linear in v, we need only check them for $v = \psi_1, \cdots, \psi_n$. The resulting discrete system of equations for a general Galerkin method is

$$\begin{aligned} \int_{\Omega} \nabla \tilde{h} \cdot \nabla \psi_j\, dx &\equiv \int_{\Omega} \nabla(\phi_\Gamma + \Phi \cdot \hat{h}) \cdot \nabla \psi_j\, dx \\ &\equiv \int_{\Omega} \nabla \phi_\Gamma \cdot \nabla \psi_j\, dx + \sum_{i=1}^{n} h_i \int_{\Omega} \nabla \phi_i \cdot \nabla \psi_j\, dx \\ &= \int_{\Omega} g\psi_j\, dx, \qquad j = 1, \cdots, n. \end{aligned} \tag{3.36}$$

Remember that we start with the discrete solution space Φ^*, then define the test space Φ to be Φ^* restricted to those functions vanishing on Γ. The term $\phi_\Gamma \in \Phi^*$ is a "base" term which approximates the boundary condition h_Γ on Γ. We can try just about any kind of function spaces for Φ and Ψ, but we can make some broad distinctions, for example:

- If $\Psi = \Phi$, then we call this a *Bubnov–Galerkin* method.
- If $\Psi \neq \Phi$, then we call this a *Petrov–Galerkin* method.

[5] The function space Ψ is called the trial space as it determines the finite number of "trial equations" which need to hold in order to establish that $\tilde{h}$ is an approximate solution.

Further, whether Bubnov– or Petrov–Galerkin, there are two common classes of H_0^1-conforming Galerkin methods:

- If the Φ and Ψ basis functions have well-localised compact support, we get the *finite element method*.
- If the Φ and Ψ basis functions are non-localised global functions, for example harmonics of Ω, we get the *spectral method*.

Next, we will derive a simple finite element method for Poisson's equation.

Relation to FVM: The finite volume method can be considered a sharp edge case of (3.36), where Ψ_{FVM} is defined as

$$\Psi_{\text{FVM}} = \text{span}\left\{\chi(\Omega_1), \cdots, \chi(\Omega_n)\right\}$$

for some choice of control volumes $\Omega_1, \cdots, \Omega_n$. While Ψ_{FVM} is technically not H_0^1-conforming, if the gradient generalisation (3.33) is used, the result is the finite volume method.

3.5 Discretizing Poisson's equation by finite elements

We now derive two finite element methods for Poisson's equation. The first will use the same piecewise linear triangulation scheme as for the previous FVM, but with seemingly different integrals — the result will turn out to not give any improvement over algorithm **??**. However, we will subsequently generalise this finite element method to a higher order method.

3.5.1 Forming a linear system

We now start with the weak form (3.35). Firstly, we can express the resulting Galerkin linear system in matrix form as

$$\begin{aligned} A\hat{h} &= \begin{bmatrix} \int_\Omega \nabla\phi_1 \cdot \nabla\psi_j \, dx & \cdots & \int_\Omega \nabla\phi_n \cdot \nabla\psi_j \, dx \\ \vdots & & \vdots \\ \int_\Omega \nabla\phi_1 \cdot \nabla\psi_j \, dx & \cdots & \int_\Omega \nabla\phi_n \cdot \nabla\psi_j \, dx \end{bmatrix} \begin{bmatrix} h_1 \\ \vdots \\ h_n \end{bmatrix} \\ &= \begin{bmatrix} \int_\Omega g\psi_j \, dx - \int_\Omega \nabla\phi_\Gamma \cdot \nabla\psi_j \, dx \\ \vdots \\ \int_\Omega g\psi_j \, dx - \int_\Omega \nabla\phi_\Gamma \cdot \nabla\psi_j \, dx \end{bmatrix} = \hat{f}. \end{aligned} \tag{3.37}$$

This is much the same as the FVM matrix equation (3.13), with flux integrals replaced by integrals against trial functions (giving "blurred" flux integrals). Notice that the integrals in (3.37), in contrast to (3.13), are over the *entire domain*. It may be very costly in general to construct such a matrix, requiring the computation of many large integrals. The finite element method, in particular, solves this problem by "localising" basis functions of the test and trial spaces. Each basis test function ϕ_i has *compact support*, meaning that there is some compact subdomain D_i^ϕ such that

$$\phi_i(x) = \begin{cases} \phi_i(x) & \text{if } x \in D_i^\phi \\ 0 & \text{otherwise,} \end{cases}$$

and similarly each basis trial function ψ_i has some corresponding compact subdomain D_i^ψ such that

$$\psi_i(x) = \begin{cases} \psi_i(x) & \text{if } x \in D_i^\psi \\ 0 & \text{otherwise.} \end{cases}$$

This has the effect of reducing the domain of each integral in (3.37), as

$$\int_\Omega \nabla\phi_i \cdot \nabla\psi_j \, dx = \int_{D_i^\phi \cap D_j^\psi} \nabla\phi_i \cdot \nabla\psi_j \, dx.$$

With well-localised basis functions, the intersection will be

$$D_i^\phi \cap D_j^\psi = \emptyset$$

for most indices i, j, implying the matrix (3.37) will be *sparse*. In practice this allows the use of iterative or graph-based sparse matrix algorithms which can be hugely more efficient than dense matrix computations of the same size. The size of the matrix will increase quadratically with the number of nodes in the discretization, while the number of nonzeros typically increases linearly. A typical sparsity pattern for a finite element problem is shown in figure 3.11.

Figure 3.11: An example sparsity pattern for the finite element matrix of a 2D Poisson problem. White is zero and black is non-zero. The mesh has 61 vertices (16 on the boundary), and 104 triangles, resulting in a 45x45 system with 2025 entries, 197 non-zeros, and fill of 0.0973.

3.5.2 A piecewise linear finite element triangulation scheme

We can use almost the same piecewise linear triangulation scheme as in the finite volume method, but we no longer take trial flux integrals over mixed Voronoi cells. Instead, we can use the Bubnov–Galerkin method: Let $\Psi = \Phi$ (which are the hat functions) and form the system of linear equations (3.36).

Computing the linear system by closed form integration

In deriving the FVM, we had to compute closed form flux integrals over complex polygonal cells. At first sight, the integration required for our new FEM method is different, although many of the steps here are the same as in section 3.3.4. However, the integrals for the matrix system (3.36) will turn out to be *exactly* the same as those derived for our finite volume method. This is a peculiarity of the hat functions — we will then show how this, seemingly pedantic, complete derivation of this equivalent FEM scheme, can be generalised.

Again, we use the piecewise linear interpolation (3.14) of the Dirichlet boundary function h_Γ:

$$\phi_\Gamma = h_\Gamma(p_1^\Gamma)\mathrm{Hat}(p_1^\Gamma) + \cdots + h_\Gamma(p_{n_\Gamma}^\Gamma)\mathrm{Hat}(p_{n_\Gamma}^\Gamma).$$

As before, we expand the gradient $\nabla\tilde{h}$ (3.16) by linearity as

$$\nabla\tilde{h} = \nabla\phi_\Gamma + \nabla\phi = \sum_{i=1}^{n_\Gamma} h_\Gamma(p_i^\Gamma)\nabla\phi_i^\Gamma + \sum_{i=1}^{n} h_i\nabla\phi_i,$$

and in this case the (Bubnov–)Galerkin linear system (3.36) becomes

$$\int_\Omega \nabla\tilde{h}\cdot\nabla\phi_j\,dx = \sum_{i=1}^{n_\Gamma}\left[h_\Gamma(p_i^\Gamma)\int_\Omega \nabla\phi_i^\Gamma\cdot\nabla\phi_j\,dx\right] + \sum_{i=1}^{n}\left[h_i\int_\Omega \nabla\phi_i\cdot\nabla\phi_j\,dx\right]. \tag{3.38}$$

As in section 3.3.4, we now reexpress (3.38) in terms of local indices. First, note that a trial function $\psi_j = \phi_j$ has compact support, with domain $D_j^\phi = D_j^\psi = D_j$, which is the union of triangles incident to the interior vertex p_j. This domain intersects with the domains of those hat basis functions of Φ^* which are centred on p_j or an adjacent vertex (either in the interior or on the boundary). We will compute the integral for a certain trial function $\psi = \psi_j$. We repeat the local index definitions from section 3.3.4 here.

Some local indexing definitions.

Let $v = p_j$ be the interior vertex corresponding to the trial function $\psi = \psi_j = \phi_j$, and denote by $v_1, \cdots, v_k$ its adjacent vertices (either in the interior or on the boundary), and by $T_1^v \cdots T_k^v$ its adjacent triangles. Triangle T_l^v joins v, v_l, v_{l+1} where addition wraps around from k to 1. As it will be necessary to relate these local indices (denoted l) to the global vertex indices (denoted i), define an index map by

$$I(l) = \text{the global index } i \text{ of the } p_i \text{ corresponding to adjacent vertex } v_l,$$

which is valid only when v_l is an interior vertex.

The local form of the equation (3.38) is then

$$\int_\Omega \nabla\tilde{h}\cdot\nabla\psi\,dx = h_j\int_\Omega \nabla\phi_v\cdot\nabla\psi\,dx + \sum_{l=1}^{k} h_{v_l}\int_\Omega \nabla\phi_l^v\cdot\nabla\psi\,dx, \tag{3.39}$$

where, as in 3.3.4, h_{v_i} is defined as

$$h_{v_i} = \begin{cases} h_\Gamma(p_i^\Gamma) & \text{if } v_i \text{ is a boundary vertex,} \\ h_{I(v_i)} & \text{if } v_i \text{ is an interior vertex.} \end{cases}$$

Again, with the same reasoning as in section 3.3.4, due to $\nabla\phi_i = \nabla\psi_i$ being constant on each triangle, to simplify integration we break (3.39) into a sum of integrals over triangles:

$$\int_\Omega \nabla\tilde{h}\cdot\nabla\psi\,dx = \sum_{t=1}^{k}\left[\int_{T_t^v}\nabla\phi_v\cdot\nabla\psi\,dx + \sum_{l=1}^{k} h_{v_l}\int_{T_t^v}\nabla\phi_l^v\cdot\nabla\psi\,dx\right]. \tag{3.40}$$

We want to reduce an integral over each separate triangle (oblique, isosceles, etc.) to an integral over a reference triangle for which we have closed forms. This is a common pattern in the finite element method. We will transform each triangle domain T into a "reference element" $\mathcal{R}$, defined by

$$\mathcal{R} = \left\{(x,y)\in[0,1]^2 \mid x+y<1\right\}.$$

We introduce the "reference basis functions", which account for all possible ways a hat function could intersect with this triangle:

$$\Phi_1^\mathcal{R} = 1-x-y, \quad \text{and} \quad \Phi_2^\mathcal{R} = x \quad \text{and} \quad \Phi_3^\mathcal{R} = y. \tag{3.41}$$

with gradients

$$\nabla\Phi_1^{\mathcal{R}} = (-1,-1)^T, \quad \text{and} \quad \nabla\Phi_2^{\mathcal{R}} = (1,0)^T \quad \text{and} \quad \Phi_3^{\mathcal{R}} = (0,1)^T. \tag{3.42}$$

(draw $\mathcal{R}$ and reference basis functions)

Suppose we are computing the integral (3.40) around some interior vertex v, and are computing the subintegral for some incident triangle T with vertices v, v', v''. We would like to make a change of variables to the reference domain $\mathcal{R}$. The affine map $\gamma : \mathcal{R} \to T$ defined by

$$\gamma(x,y) = v + (v'-v)x + (v''-v)y \tag{3.43}$$

gives a pullback of scalar functions $\Phi_{1,2,3}^{\mathcal{R}}$ on $\mathcal{R}$ to scalar functions $\Phi_{1,2,3}$ on T. These scalar functions on T correspond exactly to those pieces of basis hat functions whose gradients we want to integrate.

(draw this affine mapping)

To derive a change of variables formula, we need to compute the Jacobian matrix of γ:

$$J = \begin{bmatrix} \frac{\partial\gamma_x}{\partial x} & \frac{\partial\gamma_x}{\partial y} \\ \frac{\partial\gamma_y}{\partial x} & \frac{\partial\gamma_y}{\partial y} \end{bmatrix} = \left[\begin{array}{c|c} v'-v & v''-v \end{array}\right].$$

If we pushforward the domain of integration from T to $\mathcal{R}$ by γ (enacting a change of variables), the resulting expression is

$$\int_T \nabla\Phi_l \cdot \nabla\Phi_m \, dx = \int_{\mathcal{R}} \det(J)(J^{-T}\nabla\Phi_l^{\mathcal{R}}) \cdot (J^{-T}\nabla\Phi_m^{\mathcal{R}}) \, dx. \tag{3.44}$$

Since J depends only on the triangle vertex positions v, v', v'', it is constant on $\mathcal{R}$, and we can easily compute its inverse transpose (using the standard 2×2 inverse formula):

$$J^{-T} = \frac{1}{\det(J)} \begin{bmatrix} v''_y - v_y & v_y - v'_y \\ v_x - v''_x & v'_x - v_x \end{bmatrix} = \frac{1}{2\text{Area}(T)} \left[\begin{array}{c|c} (v-v'')^\perp & (v'-v)^\perp \end{array}\right].$$

Since the integrand is constant, (3.44) simplifies to

$$\int_T \nabla\Phi_l \cdot \nabla\Phi_m \, dx = \text{Area}(T)(J^{-T}\nabla\Phi_l^{\mathcal{R}}) \cdot (J^{-T}\nabla\Phi_m^{\mathcal{R}}), \tag{3.45}$$

and if we define the matrix

$$K = \left[\begin{array}{c|c} (v-v'')^\perp & (v'-v)^\perp \end{array}\right],$$

the equation (3.45) is further simplified to

$$\int_T \nabla\Phi_l \cdot \nabla\Phi_m \, dx = \frac{1}{4\text{Area}(T)}(K\nabla\Phi_l^{\mathcal{R}}) \cdot (K\nabla\Phi_m^{\mathcal{R}}). \tag{3.46}$$

Plugging in the gradients , we get

$$K\nabla\Phi_1^{\mathcal{R}} = (v''-v')^\perp, \quad \text{and} \quad K\nabla\Phi_2^{\mathcal{R}} = (v-v'')^\perp, \quad \text{and} \quad K\nabla\Phi_3^{\mathcal{R}} = (v'-v)^\perp. \tag{3.47}$$

By plugging these expressions into (3.46), we get

$$\int_L \nabla\Phi_v \cdot \psi \, dx = \frac{1}{4\text{Area}(T)} \|v_{t+1} - v_t\|^2,$$
$$\int_L \nabla\Phi_{v_t} \cdot \nabla\psi \, dx = \frac{1}{4\text{Area}(T)} (v - v_{t+1}) \cdot (v_{t+1} - v_t),$$
$$\int_L \nabla\Phi_{v_{t+1}} \cdot \nabla\psi \, dx = \frac{1}{4\text{Area}(T)} (v_t - v) \cdot (v_{t+1} - v_t).$$

Note that these are the same as the closed forms (3.22) derived for the finite volume method. Now we notice that all this work has been wasted, as this finite element method gives exactly the same matrix as the previous finite volume method.

Integrating the source function

Again, one more thing is needed. We require some numerical integration for the right-hand side source terms

$$\int_\Omega g\psi_j \, dx, \quad j = 1, \cdots, n.$$

Again, we use the simplest choice, a "rectangle rule", where we sample the source function g at the interior vertex p_j, then assume that g is constant over Ω_j. Due to the assumption that g is constant, we are effectively multiplying the sample g value by the geometric volume of the hat function ψ_j, which is one fourth the area of its domain D_j^ψ. This gives the approximation

$$\int_\Omega g\psi_j \, dx \quad \approx \quad g(p_j) \frac{1}{4} \sum_{t=1}^{k} \text{Area}(T_t^{p_j}). \tag{3.48}$$

Note that $\frac{1}{4}\sum_{t=1}^{k} \text{Area}(T_t^{p_j})$, the volume of the hat function, is also equal to the area of the midpoint-connecting polygon. Approximation (3.48) is *almost* the same as the FVM source approximation (3.24), but instead of integrating over the mixed Voronoi cell, we integrate over the midpoint-connecting polygon.

(draw this)

We therefore will not bother in implementing this piecewise linear scheme. So what was the point?

3.5.3 A piecewise quadratic finite element triangulation scheme

It should have been clear from the start that the resulting integrals would be the same, due to the simple geometry of hat functions — however, it is instructive to work through the computational, routine derivation in section 3.5.2, as we can generalise from hat functions to higher order piecewise polynomial functions.

The computation of hat function integrals in sections 3.3.4 and 3.5.2 turned out to reduce to integrals of linear functions (and their gradients) over triangles. So lets focus on just one triangle T. We have a standard basis on T, consisting of those linear functions which are 1 at one vertex and 0 at the others. The hat function basis appears incidentally, as we identify coefficients of vertices shared by adjacent triangles. A key idea of finite elements extends this naturally: Form a function space over a triangle (or more generally some other cell shape such as a quadrilateral), and choose some "nodal basis". A linear function on T is determined by its value at the corners — the linear basis functions are called "nodal" due to their defining property, being 1 at one vertex and 0 at the others. When a nodal point is shared by two triangles, the coefficient values are identified. This implicitly determines a basis function of the global test and trial function spaces, Φ and Ψ.

The piecewise quadratic basis functions

As we tried linear polynomials on T, why not try quadratic polynomials? There are 6 degrees of freedom for quadratic polynomials given in barycentric coordinates, so a natural choice of nodal points are the triangle vertices and their midpoints, shown below.

(nodal points and corresponding basis functions)

Using barycentric coordinates (x, y, z) on T, where $x + y + z = 1$, the quadratic nodal basis functions are

$$\begin{aligned} \Phi_{200} &= x(1-2y-2z), \quad & \Phi_{110} &= 4xy, \\ \Phi_{020} &= y(1-2z-2x), \quad & \Phi_{011} &= 4yz, \\ \Phi_{002} &= z(1-2x-2y), \quad & \Phi_{101} &= 4zx. \end{aligned} \tag{3.49}$$

We use "barycentric indices", common in the computer-aided geometric design literature. These indices are convenient as, up to a scaling, they are the barycentric coordinates of the basis function's corresponding node. These basis functions are displayed in figure 3.12.

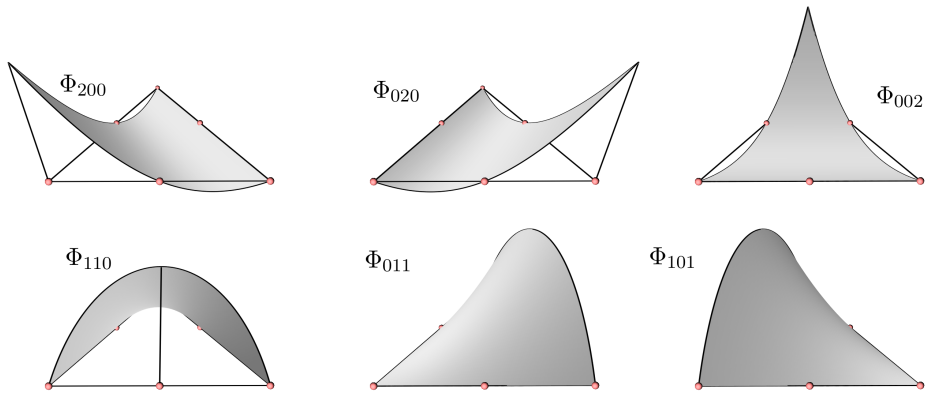


Figure 3.12: The quadratic nodal basis functions on the triangle, where the nodes are the vertices and midpoints. The barycentric coordinates of the triangle vertices (from the lower-left then anticlockwise) are $(x, y, z) = (1, 0, 0), (0, 1, 0), (0, 0, 1)$.

As the linear functions on the triangle, with incident nodes identified, implicitly defined hat functions, so do these quadratic functions on the triangle. We get two classes of global basis functions, formed by stitching together incident nodal basis functions: those associated to a vertex, and those associated to a midpoint. Examples of these basis functions are displayed in figure 3.13.

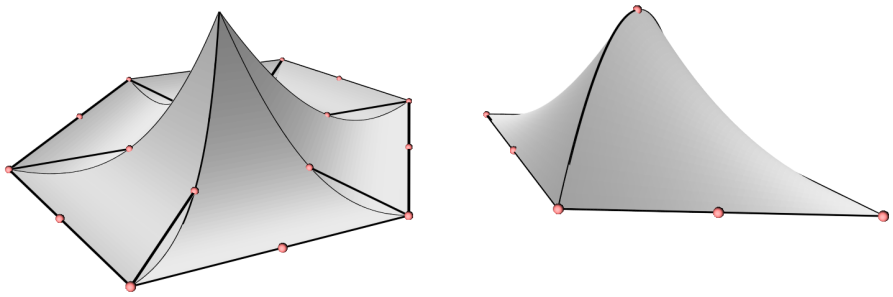


Figure 3.13: Left: The global piecewise quadratic basis function determined by an interior vertex node, with 6 incident triangles. Right: The global piecewise quadratic basis function determined by an interior midpoint node.

Much like how hat functions combine to give a piecewise linear continuous interpolation of any function, we now get a piecewise quadratic continuous interpolation, simply by combining the basis functions by the function evaluated at each nodal point. Example interpolations are shown in figure 3.14.

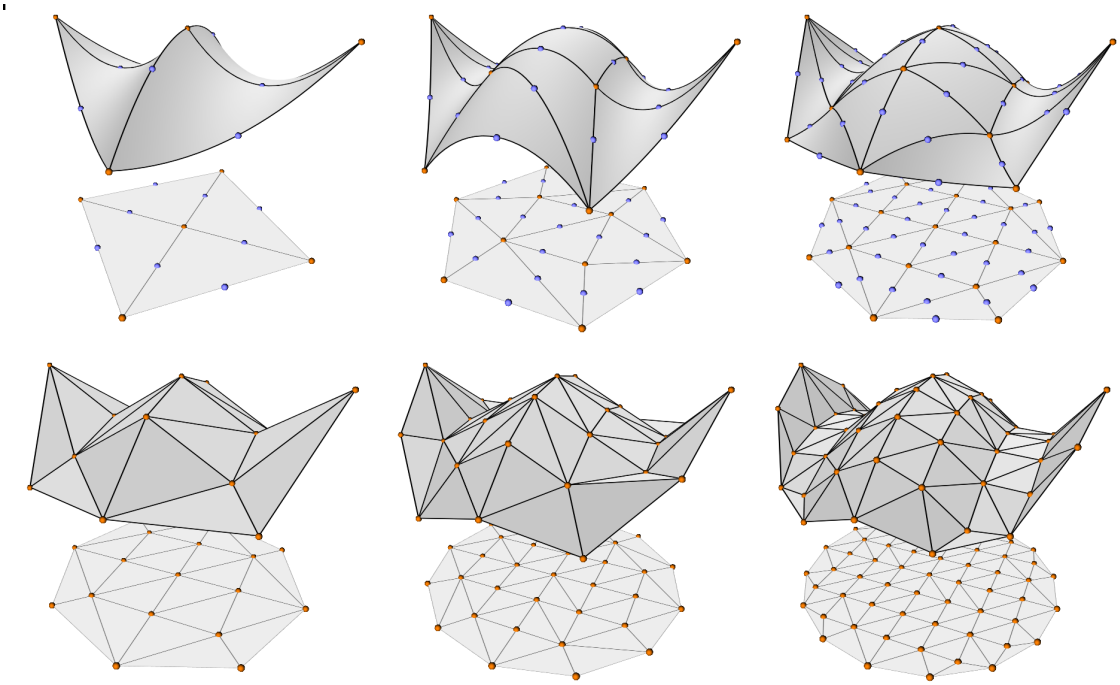


Figure 3.14: Interpolations of $f(x, y) = \frac{1}{2}(x^2 - y^2) + 1.23 + \frac{3}{10}\cos(4x)$ on the unit circle (with linear boundary approximation). First row: Piecewise quadratic interpolations of f. Second row: Piecewise linear interpolations of f. The adjacent quadratic and linear interpolations have roughly the same number of sample points — the trade-off is between the number of triangles and the order of the basis functions.

The linear system

3.5.4 Computing the linear system for the piecewise quadratic scheme

In section 3.5.2 we derived the expressions needed to form the linear system for the piecewise linear scheme. With the nodal basis functions expressed in coordinates of a reference triangle $\mathcal{R}$, by performing a change of variables we can compute closed form expressions for the integrals restricted to each triangle. This process is the same in the quadratic case, although it becomes more tedious. Thankfully, this process can be automated by using an algebraic computing language — I have used the Python library SymPy [42]. Listing 3.15 displays Python code which has been used to compute coefficients. The function **integrate_gradients**(a, b, c, i, j, k) computes a symbolic definite integral,

$$\int_{\mathcal{R}} (K\nabla\Phi^{\mathcal{R}}_{abc}) \cdot (K\nabla\Phi^{\mathcal{R}}_{ijk})\, d\hat{x} = \int_{x=0}^{x=1} \int_{y=0}^{y=1-x} (K\nabla\Phi^{\mathcal{R}}_{abc}) \cdot (K\nabla\Phi^{\mathcal{R}}_{ijk})\, dy\, dx, \tag{3.50}$$

where $\Phi^{\mathcal{R}}_{abc}$ is the nodal basis function Φ_{abc} in reference triangle coordinates, with z replaced by $1 - x - y$. As a rough outline, the resulting matrix assembly algorithm (for constructing A and $\hat{f}$), like in the pseudocode **??**, simply

- iterates over each trial basis function $\psi \in \Psi$,

- determines the basis functions $\phi^* \in \Phi^*$ which have domain overlapping ψ's domain (by some mesh traversal),
- splits the domain of integration into triangles (where the restricted basis functions are quadratic — previously they were linear),
- computes the integrals in closed form on this triangle (where these integrals could be derived using computer algebra),
- updates the matrix and RHS values, depending on whether ϕ^*'s node is on the boundary or not.

The new matrix assembly algorithm is given in listing **??**. The fully expanded algorithm, with mesh traversals and closed form integrals hard-coded, is becoming nigh unreadable. Clearly there is a simple coherent structure underlying all of these steps. We could easily generalise to piecewise cubics by having $10 = 1 + 2 + 3 + 4$ nodes per triangle — just find a nodal basis for cubics on a reference triangle, and plug them into the same Python script 3.15 to derive coefficients of closed form integrals (which, again, will be in terms of dot products between K_1, K_2, and K_3). We then would need to write out the whole mesh traversal and index manipulations required, in order to implement what conceptually is a trivial increase of order from 2 to 3. This process (the derivation and implementation of a finite element algorithm) is clearly a good candidate for automation.

3.5.5 The resulting matrix assembly algorithm

3.5.6 The resulting finite element algorithm

3.5.7 Validation and results

```python
import sympy as sym

# Helper function to convert a barycentric index to an index into
# a flat array.
def barycentric_index_to_linear(i,j,k):
    assert(i+j+k == 2)
    ...

# Define algebraic symbols for the variables.
x,y,z = sym.symbols("x y z")

# The reference triangle has vertices v, v', v''.
# We define K_1, K_2, K_3 by
#     K_1 = (v - v'')^⊥,
#     K_2 = (v' - v)^⊥,
#     K_3 = (v'' - v')^⊥.
# K_1 and K_2 are the columns of the matrix K
# (which is the inverse transpose Jacobian without the constant term).
# while K_3 = -K_1 - K_2.
K1,K2,K3 = sym.symbols("K1 K2 K3")

# Define the quadratic basis functions Φ_ijk
# given in barycentric coordinates (x,y,z).
# The order with respect to barycentric indices is 200, 020, 002, 110, 011, 101.
nodal_basis_functions_barycentric = [
    x*(1 - 2*y - 2*z),
    y*(1 - 2*z - 2*x),
    z*(1 - 2*x - 2*y),

    4*x*y,
    4*y*z,
    4*z*x
]
# Map the nodal basis functions to the reference triangle
# by substituting z with 1 - x - y.
# These remapped basis functions will be denoted by Φ^R_ijk.
nodal_basis_functions = [p.subs(z,1-x-y).expand() for p in
↪  nodal_basis_functions_barycentric]
# Compute the gradients ∇Φ^R_ijk.
nodal_basis_gradients = [(sym.diff(p, x), sym.diff(p, y)) for p in
↪  nodal_basis_functions]

# Provide a function to compute the integral
#     ∫_R (K∇Φ^R_abc) · (K∇Φ^R_ijk) dx̂
#     = ∫_{x=0}^{x=1} ∫_{y=0}^{y=1-x} (K∇Φ^R_abc) · (K∇Φ^R_ijk) dy dx.
def integrate_gradients(a,b,c, i,j,k):
    grad1 = nodal_basis_gradients[barycentric_index_to_linear(a,b,c)]
    grad2 = nodal_basis_gradients[barycentric_index_to_linear(i,j,k)]
    f = (K1*grad1[0] + K2*grad1[1])*(K1*grad2[0] + K2*grad2[1])
    f_dy = sym.integrate(f, (y, 0,1-x))
    f_dy_dx = sym.integrate(f_dy, (x, 0,1))
    # Print the result.
    print("{}{}{},{}{}{}:".format(a,b,c,i,j,k),
    ↪  f_dy_dx.simplify().subs(-K1-K2,K3))
```

Figure 3.15: Python code to compute the closed form expressions required when forming the linear system. This code uses the SymPy library [42].

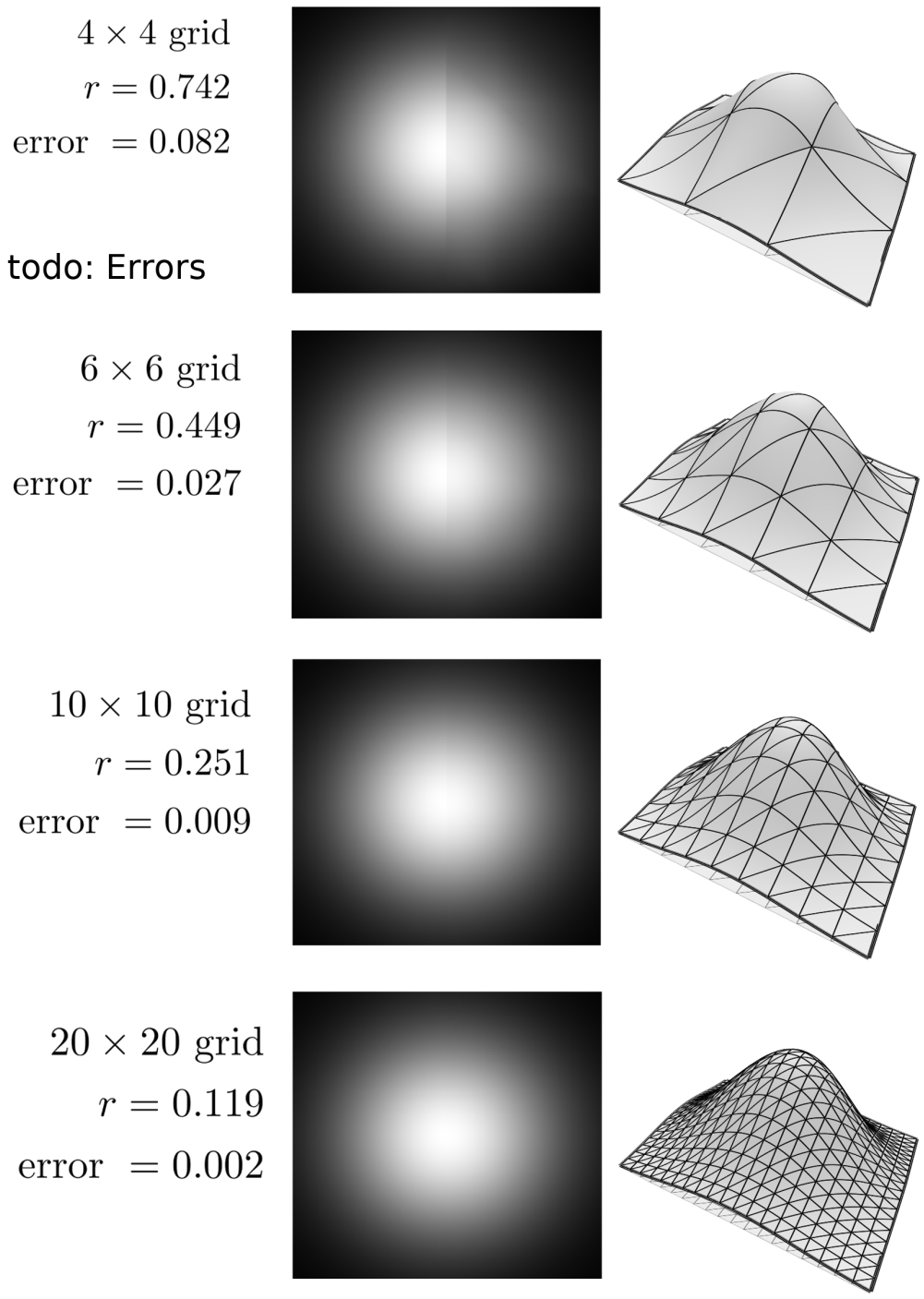


Figure 3.16: Approximate solutions to the test problem (3.27) with varying r. The middle squares contain plots of the exact solution (on the left rectangle), and the approximate solution (on the right rectangle). Black is $h = 0$, and white is $h = 1$. On the right, a 3D view of the solution and mesh is shown.

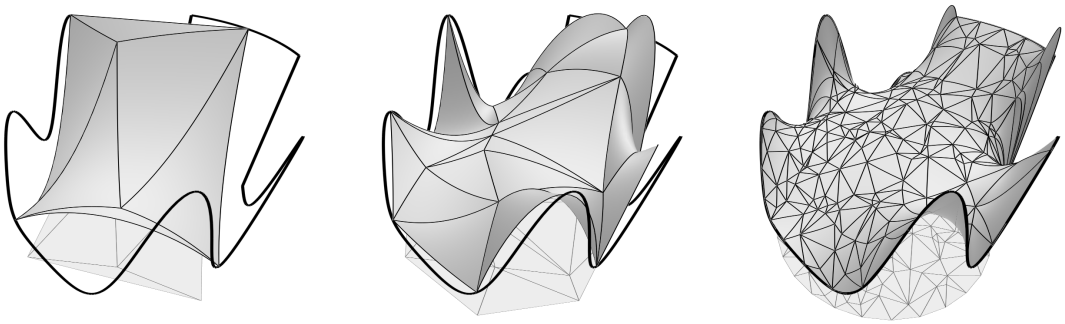
Routine *Poisson_Quadratic_FEM_Matrix_Assembly*:

Data: Domain Ω with boundary Γ.
Heat source function $g : \Omega \to \mathbb{R}$.
Dirichlet boundary function $h_\Gamma : \Gamma \to \mathbb{R}$.
Triangulation of Ω consisting of:
Interior vertices $p_1, \cdots, p_n$.
Boundary vertices $p_1^\Gamma, \cdots, p_{n_\Gamma}^\Gamma$.
Interior edges $e_1, \cdots, e_m$.
Boundary edges $e_1^\Gamma, \cdots, e_{m_\Gamma}^\Gamma$.
Triangles $T_1, \cdots, T_{n_T}$.

Result: The coefficients $(A, \hat{f})$ of the system $A\hat{h} = \hat{f}$ (3.13), which can be solved by the caller for $\hat{h}$ in order to reconstruct an approximate solution $\tilde{h} : \Omega \to \mathbb{R}$ of the boundary value problem (3.2), $-\Delta h = g,\ h|_\Gamma = h_\Gamma$.

$N \leftarrow n + m$;
$A \leftarrow N \times N$ zero matrix;
$\hat{f} \leftarrow$ length N zero vector;
for $j = 1, \cdots, n$ **do**
  Let v denote the interior vertex p_j.
  Let $v_1, \cdots, v_k$ denote the vertices adjacent to v.
  Let $T_1^v, \cdots, T_k^v$ denote the triangles adjacent to v.
  (Triangle T_t^v joins vertices v, v_t, v_{t+1}.)
  Let $e_1^v, \cdots, e_k^v$ denote the edges from v to each $v_1, \cdots, v_k$.
  for $t = 1, \cdots, k$ **do**
    $C \leftarrow \frac{1}{4\mathrm{Area}(T_t^v)}$;
    $K_1 \leftarrow (v - v_{t+1})^\perp, \quad K_2 \leftarrow (v_t - v)^\perp, \quad K_3 \leftarrow (v_{t+1} - v_t)^\perp$;
    $A[j,j] \leftarrow A[j,j] + \frac{1}{2}C\|K_3\|^2$;
    if v_t *is on the boundary* **then**
      $\hat{f}[I(v_t)] \leftarrow \hat{f}[I(v_t)] + \frac{1}{6}h_\Gamma(v_t)CK_1 \cdot K_3$;
    else
      $A[j, I(v_t)] \leftarrow A[j, I(v_t)] - \frac{1}{6}CK_1 \cdot K_3$;
    end
    if v_{t+1} *is on the boundary* **then**
      $\hat{f}[I(v_{t+1})] \leftarrow \hat{f}[I(v_{t+1})] + \frac{1}{6}h_\Gamma(v_{t+1})CK_2 \cdot K_3$;
    else
      $A[j, I(v_{t+1})] \leftarrow A[j, I(v_{t+1})] - \frac{1}{6}CK_2 \cdot K_3$;
    end
    $A[j, I(e_t^v)] \leftarrow A[j, I(e_t^v)] + \frac{2}{3}CK_1 \cdot K_3$;
    $A[j, I(e_{t+1}^v)] \leftarrow A[j, I(e_{t+1}^v)] + \frac{2}{3}CK_2 \cdot K_3$;
  end
end
for $j = 1, \cdots, m$ **do**
  Let e denote the interior edge e_j.
  Denote by T_0^e and T_1^e the two triangles incident to e.
  for $t = 0, 1$ **do**
    $C \leftarrow \frac{1}{4\mathrm{Area}(T_t^e)}$;
    Let v, v', v'' be the anticlockwise vertices of T_t^e, where v is the vertex opposite to e.
    Let e' and e'' be the edges from v to v' and from v to v'', respectively.
    $K_1 \leftarrow (v - v'')^\perp, \quad K_2 \leftarrow (v' - v)^\perp, \quad K_3 \leftarrow (v'' - v')^\perp$;
    $A[n+j, n+j] \leftarrow A[n+j, n+j] + \frac{4}{3}C(\|K_3\| - K_1 \cdot K_2)$;
    if e'' *is on the boundary* **then**
      $b[n+j] \leftarrow b[n+j] - \frac{4}{3}h_\Gamma(e''\mathrm{midpoint})CK_1 \cdot K_3$;
    else
      $A[n+j, I(e'')] \leftarrow A[n+j, I(e'')] + \frac{4}{3}CK_1 \cdot K_3$;
    end
    if e' *is on the boundary* **then**
      $b[n+j] \leftarrow b[n+j] - \frac{4}{3}h_\Gamma(e'\mathrm{midpoint})CK_2 \cdot K_3$;
    else
      $A[n+j, I(e')] \leftarrow A[n+j, I(e')] + \frac{4}{3}CK_2 \cdot K_3$;
    end
    if v' *is on the boundary* **then**
      $b[n+j] \leftarrow b[n+j] - \frac{2}{3}h_\Gamma(v')CK_1 \cdot K_2$;
    else
      $A[n+j, I(v')] \leftarrow A[n+j, I(v')] + \frac{2}{3}CK_1 \cdot K_2$;
    end
    if v'' *is on the boundary* **then**
      $b[n+j] \leftarrow b[n+j] - \frac{2}{3}h_\Gamma(v'')CK_1 \cdot K_2$;
    else
      $A[n+j, I(v'')] \leftarrow A[n+j, I(v'')] + \frac{2}{3}CK_1 \cdot K_2$;
    end
  end
end
return $(A, \hat{f})$;

Algorithm 2: Text



Chapter 4

Solving the Stokes equations with the finite element method

4.1 Introduction

The Stokes equations (2.13), which are solved for an incompressible Navier-Stokes flow, assume the Reynolds number is $Re \ll 1$ and thus convective processes are neglible in comparison to viscous processes. We will begin with the steady-state form (2.14). Due to this simplification, the Steady stokes equations (2.14), repeated here:

$$-\mu\Delta u + \nabla p = \rho g, \quad \nabla \cdot u = 0,$$

form a constrained linear equation. As we saw in section 2.4, the pressure term p is a Lagrange multiplier introduced with the constraint $\nabla \cdot u = 0$.

4.1.1 The lid-driven cavity flow problem

A standard test case in computational fluid dynamics is the *lid-driven cavity flow* problem. We let the domain be is $\Omega = [-1, 1]^2$ and define a Dirichlet boundary condition:

$$u|_\Gamma = u_\Gamma(x, y) = \begin{cases} (1,0)^T & \text{if } y = 1, \\ (0,0)^T & \text{otherwise.} \end{cases} \tag{4.1}$$

We will use a finite element method to solve the steady Stokes flow (2.14) with this domain and boundary condition. A highly viscous incompressible fluid under these conditions will form a single stable vortex.

4.1.2 Attempting solution by an iterative pressure update

Since the steady Stokes equations are a linearly constrained vector Poisson equation, it would be convenient to use a solution method which trivially extends our already-implemented Poisson solver. One idea is to define a coupled system of equations replacing the constrained equation (2.14), resulting in an initial boundary value problem in auxilliary parameter γ:

$$\begin{aligned} -\nu\Delta u &= -\nabla p_*, \\ \frac{\partial p_*}{\partial \gamma} &= -C\nabla \cdot u, \\ \text{with} \quad u|_\Gamma = u_\Gamma, u(\hat{x}, 0) &= u_0(\hat{x}), \quad \text{and} \quad p_*(\hat{x}, 0) = 0, \end{aligned} \tag{4.2}$$

where $C > 0$ controls the speed of the divergence elimination. With explicit Euler in the parameter γ, the resulting iterative fixed point method is

$$\begin{aligned} u^{(0)} &\leftarrow u_0, \\ p_*^{(0)} &\leftarrow 0, \\ -\nu\Delta u^{(n)} &\leftarrow -\nabla p_*^{(n)}, \\ p_*^{(n+1)} &\leftarrow p_*^{(n)} - \Delta\gamma C\nabla\cdot u^{(n)}, \end{aligned} \tag{4.3}$$

This has some problems, notably, that if $\nabla\cdot u$ is constant the pressure gradient will remain unchanged, meaning that $u^{(n)}$ will reach a fixed point while the pressure blows up. It is feasible that this iteration could be modified such that the only effective fixed points occur when $\nabla\cdot u = 0$, but luckily, the method converges for the lid-driven cavity problem.

Implementing the pressure iteration

To implement this iteration, the divergence $\nabla\cdot u^{(n)}$ must be projected onto the pressure space P1. For each pressure basis function $\psi_1^p, \cdots, \psi_{n_p}^p$, an integral is computed and stored in a vector of values

$$\hat{v}_{\text{proj}} \leftarrow \left[\int_\Omega \psi_j^p \nabla\cdot u^{(n)}\right]_{j=1,\cdots,n_p}.$$

This is the L^2 projection of $\nabla\cdot u^{(n)}$ onto the P1 finite element space, so to retrieve the hat function coefficients, form the P1 (Gramian) projection matrix

$$A \leftarrow \left[\int_\Omega \phi_i^p \psi_j^p\right]_{i,j=1,\cdots,n_p}$$

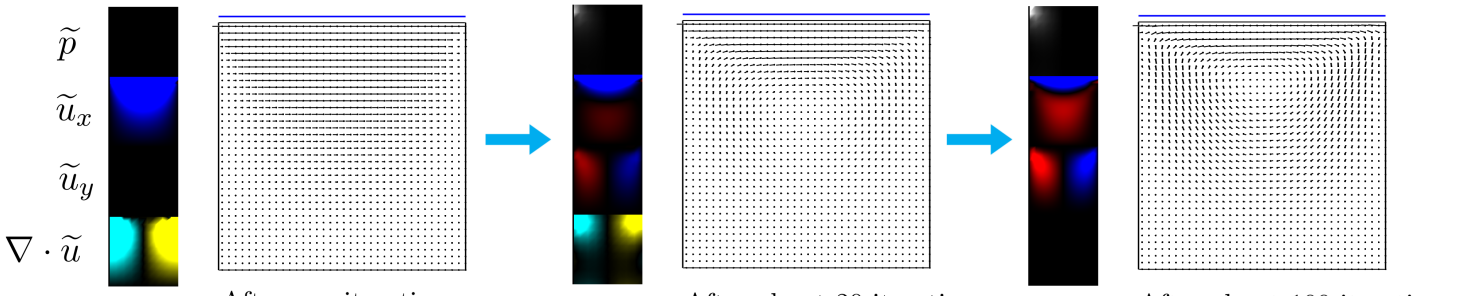
and solve the linear system

$$A\hat{v} = \hat{v}_{\text{proj}}.$$

Due to the extreme sparsity and near-orthogonality of this system, it can be solved very efficiently. Finally, the pressure update is

$$\hat{p}_*^{(n+1)} \leftarrow \hat{p}_*^{(n)} - \Delta t C\hat{v},$$

where $\hat{p}_*^{(n)}$ denotes the vector of coefficients of hat functions which combine to $p_*^{(n)}$. The results for the lid-driven cavity problem are displayed in figure **??**.



After one iteration. After about 20 iterations. After about 100 iterations.

This has taken quite a large number of iterations, each iteration solving two large sparse linear systems (the Gramian system being the cheaper to solve). However, the result is as expected, and this method is easy to implement given an already-working quadratic-element Poisson solver.

4.1.3 A mixed finite element method

As described in section 2.4, the pressure p is a Lagrange multiplier that appears when solving the optimization problem (2.16):

$$\begin{aligned} \underset{u}{\text{minimize}} \quad & E(u) = \frac{\mu}{2} \langle \nabla u, \nabla u \rangle - \langle u, \rho g \rangle \\ \text{subject to} \quad & \nabla \cdot u = 0. \end{aligned}$$

4.1.4 The weak form of the steady Stokes equations

As described in section 2.4, the pressure p is a Lagrange multiplier that appears when solving the optimization problem (2.16):

$$\begin{aligned} \underset{u}{\text{minimize}} \quad & E(u) = \frac{\mu}{2} \langle \nabla u, \nabla u \rangle - \langle u, \rho g \rangle \\ \text{subject to} \quad & \nabla \cdot u = 0. \end{aligned}$$

As a first idea, we can introduce p as a variable to solve for. Solving for the pressure (the "dual variable") simultaneously with the velocity (the "primal variable") is called a primal-dual method for the optimization (2.16), and the resulting finite element method is called *mixed*. Pressure then needs to be discretized, so we introduce another test space Φ_{pressure}. To get a weak form of the steady Stokes equations (2.14), which are two equations including the constraint $\nabla \cdot u = 0$, we introduce another trial space $\Psi_{\text{constraint}}$, whose functions will be integrated against $\nabla \cdot u$. The weak form is then

$$\begin{aligned} &\int_\Omega (-\mu \Delta u + \nabla p)\, \psi \, dx = \int_\Omega \rho g \cdot \psi \, dx, \\ &\int_\Omega (\nabla \cdot u)\, q \, dx = 0, \quad \text{where } \psi \in \Psi, q \in \Psi_{\text{constraint}}, \end{aligned}$$

which by integration by parts can be written as

$$\begin{aligned} &\int_\Omega \mu \nabla u : \nabla \psi - p \nabla \cdot \psi \, dx = \int_\Omega \rho g \cdot \psi \, dx, \\ &\int_\Omega - (\nabla \cdot u)\, q \, dx = 0, \quad \text{where } \psi \in \Psi, q \in \Psi_{\text{constraint}}. \end{aligned} \tag{4.4}$$

To simplify notation, we can introduce the "mixed spaces"

$$\Phi^{\mathcal{M}} = \Phi \times \Phi_{\text{constraint}} \quad \text{and} \quad \Psi^{\mathcal{M}} = \Psi \times \Psi_{\text{constraint}}$$

and, using the notation of [44], (4.4) can be rewritten as

$$a((u,p),(\psi,q)) = L((v,q)), \qquad (u,p) \in \Phi^{\mathcal{M}}, \quad (\psi,q) \in \Psi^{\mathcal{M}} \tag{4.5}$$

where a is a bilinear form defined by

$$a((u,p),(\psi,q)) = \int_\Omega \nabla u : \nabla \psi - p \nabla \cdot \psi - (\nabla \cdot u)\, q \, dx$$

and f is a linear functional defined by

$$L((v,q)) = \int_\Omega \rho g \cdot \psi \, dx.$$

—todo: Use H^1, L^2, discretize spaces later.

4.1.5 Taylor-Hood mixed finite elements for the Stokes problem

In the derivations of chapter (), many of the manipulations were trivial applications of linearity and splitting of boundary and interior terms, and is standard in the finite element literature to define forms such as the a and L above. If we discretize to n (interior) basis functions there are $2n$ basis functions for the mixed test and trial spaces, which are the natural choice:

$$\begin{aligned}\Phi^{\mathcal{M}} &= \text{span}\left((\phi_1, 0), \cdots, (\phi_n, 0), (0, \phi_1^p), \cdots, (0, \phi_n^p)\right), \\ \Psi^{\mathcal{M}} &= \text{span}\left((\psi_1, 0), \cdots, (\psi_n, 0), (0, \psi_1^p), \cdots, (0, \psi_n^p).\right)\end{aligned} \tag{4.6}$$

All we must do is plug approximations, from finite dimensional function spaces, into the weak form (4.5), while performing trial integrals only over the basis functions in the a finite dimensional mixed trial space, resulting in the linear system of equations

$$\begin{aligned}a((\tilde{u}, \tilde{p}), (\psi_j, 0)) &= L((\psi_j, 0)), \quad j = 1, \cdots, n \\ a((\tilde{u}, \tilde{p}), (0, \psi_j^p)) &= L((0, \psi_j^p)), \quad j = 1, \cdots, n,\end{aligned} \tag{4.7}$$

We must choose the finite-dimensional spaces $\Phi, \Psi, \Phi_{\text{constraint}}$, and $\Psi_{\text{constraint}}$. The algorithm for the Stokes problem, with Dirichlet boundary conditions, will then consist of forming the matrix and right-hand side of (4.8), then passing it to a linear solver.

We will use the Bubnov–Galerkin method (letting $\Psi = \Phi$), and make a standard choice of mixed finite element space,

$$\Phi^{\mathcal{M}} = \Phi \times \Phi_{\text{constraint}} = P_2^2 \times P^1,$$

where P_2 and P_1 denote, respectively, the piecewise quadratic and piecewise linear elements derived in chapter (). This mixed space is formed from the "Taylor-Hood elements", and is known to result in a stable method for the Stokes problem[1].

So we don't have to repeat the equations for each kind of mixed trial function, as we do in (4.7), we can use the mixed basis order (4.6) and define $\phi_l^{\mathcal{M}}$ to be the l'th mixed basis function in this order (e.g. $\psi_1^{\mathcal{M}} = (\psi_1, 0)$ and $\psi_{n+1}^{\mathcal{M}} = (0, \psi_1^p)$).

A note on tensor notation for the vector field space

The space P_2^2 here is a space of piecewise quadratic vector fields formed as $P_2^2 = P^2 \times P^2$. This represents the velocity field by two functions in the usual scalar quadratic space P^2. This is the simplest way to construct a finite element space of vector fields, although others (not based on a product construction) are possible. We define ∂x and ∂y to be the unit vector fields on the domain Ω. The basis functions for P_2^2 are the natural choice of vector fields $\phi_1^{\text{scalar}}\partial x$, $\phi_1^{\text{scalar}}\partial y$, $\cdots$, $\phi_n^{\text{scalar}}\partial x$, $\phi_n^{\text{scalar}}\partial y$. However, we would like to think of the velocity coefficients as vector samples, rather than scalar coefficients of this array of $2n$ basis vector fields. We can do this by letting $\phi_1, \cdots, \phi_n$ be aggregates of axis-aligned vector fields:

$$\phi_i = \begin{bmatrix}\phi_{ix} \coloneqq \phi_i^{\text{scalar}}\partial x \\ \phi_{iy} \coloneqq \phi_i^{\text{scalar}}\partial y\end{bmatrix}.$$

We can then represent the interior variation as

$$\tilde{u}_{\text{interior}} = \sum_{i=1}^{n} u_i \cdot \phi_i.$$

[1] While we could choose just about any spaces for the non-mixed method for the Poisson problem, which is elliptic, the Stokes flow problem inherently involves a Lagrange multiplier, and is known as a "saddle point problem". The resulting finite element matrix is hyperbolic. The Taylor-Hood elements satisfy the Ladyzhenskaya–Babuška–Brezzi (LBB) condition, which is a theoretical result which guarantees that the resulting linear system for the Stokes problem will be well-conditioned. If we took $P^1 \times P^1$ instead, for example, this condition would not hold.

It is important to keep this in mind, as although this notation is natural, it may cause confusion: While a is a bilinear form, the value $a((\phi_1, 0), \psi_4^{\mathcal{M}})$, for example, is actually "vectorized" into

$$\begin{bmatrix} a((\phi_{1x}, 0), \psi_4^{\mathcal{M}}) \\ a((\phi_{1y}, 0), \psi_4^{\mathcal{M}}) \end{bmatrix}.$$

This is contracted with the vector coefficient u^1 to get a scalar:

$$u^1 \cdot a((\phi_1, 0), \psi_4^{\mathcal{M}}) = u_x^1 a((\phi_{1x}, 0), \psi_4^{\mathcal{M}}) + u_y^1 a((\phi_{1y}, 0), \psi_4^{\mathcal{M}}).$$

As in chapter (), we let u be approximated by

$$\tilde{u} = \tilde{u}_\Gamma + \tilde{u}_{\text{interior}} = \tilde{u}_\Gamma + \sum_{i=1}^{n} u_i \cdot \phi_i,$$

where $\tilde{u}_\Gamma \in \Phi^*$ approximates the Dirichlet boundary function $u|_\Gamma = u_\Gamma$, and $\tilde{u}_{\text{interior}} \in \Phi$ is the "interior variation". A boundary condition for pressure is not given (—todo: Should the pressure have a Dirichlet boundary set to 0?)

$$\tilde{p} = \sum_{i=1}^{n} p_i \phi_i^p.$$

Now (4.7) expands to, rearranging constants to the right-hand side,

$$\sum_{i=1}^{n} u_i \cdot a((\phi_i, 0), \psi_l^{\mathcal{M}}) + \sum_{i=1}^{n} p_i a((0, \phi_i^p), \psi_l^{\mathcal{M}}) = L(\psi_l^{\mathcal{M}}) - a((\tilde{u}_\Gamma, 0), \psi_l^{\mathcal{M}}), \quad l = 1, \cdots, 2n. \tag{4.8}$$

As explained in the note on tensor notation, the left-most summation is over contractions of u^i with the "vectorized" bilinear form a, as ϕ_i is an aggregate of vector fields, $\phi_i = (\phi_{ix}, \phi_{iy})^T$.

The $\psi_l^{\mathcal{M}}$ mixed trial function ranges over all basis trial functions, 4.6. Suppose $\psi_l^{\mathcal{M}} = (\psi_1, 0)$. We have $\psi_1 = \phi_1$ (due to using Bubnov–Galerkin), which is the aggregate of vector fields defined in the note on tensor notation. Therefore this one apparent equation, with $\psi_l^{\mathcal{M}} = (\psi_1, 0)$ actually defines two equations,

$$\sum_{i=1}^{n} u_i \cdot a((\phi_i, 0), (\psi_{1x}, 0)) + \sum_{i=1}^{n} p_i a((0, \phi_i^p), (\psi_{1x})) = L((\psi_{1x}, 0)) - a((\tilde{u}_\Gamma, 0), (\psi_{1x}, 0)),$$
$$\sum_{i=1}^{n} u_i \cdot a((\phi_i, 0), (\psi_{1y}, 0)) + \sum_{i=1}^{n} p_i a((0, \phi_i^p), (\psi_{1y})) = L((\psi_{1y}, 0)) - a((\tilde{u}_\Gamma, 0), (\psi_{1y}, 0)).$$

Therefore equation (4.8) forms a $(2n+n_p) \times (2n+n_p)$ linear system (with $2n$ coefficients for the velocity, n_p for the pressure).

Automation

It has been very beneficial to go from the weak form of our specific Stokes problem (4.4) to the abstract form (4.5) in terms of the forms a and L. When the approximating solutions $\tilde{u}$ and $\tilde{p}$ are plugged in, all that takes place is a linear expansion and rearrangement, accounting for boundary terms.

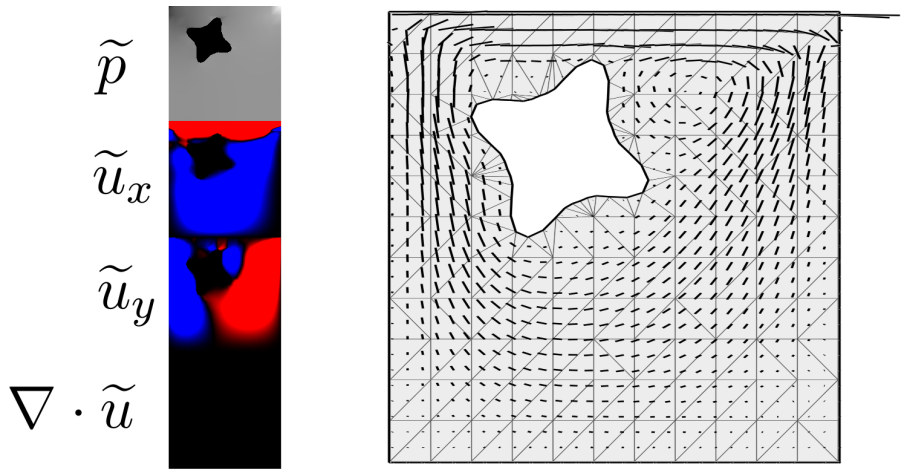
It is feasible that this process could be automated, as in, we could pass a symbolic representation of the weak form (4.4), a mesh discretizing the domain, a specification of the finite element function spaces (in some symbolic notation such as "`P1 x P2_2`"), and boundary

conditions. This is all the data specific to our problem — with this data, the finite element algorithm is fully specified, and it is a routine task to do all the expansions and rearrangements to specify the linear system, then to write a matrix assembly algorithm to traverse the mesh and construct this system.

This in fact has been done [20] [19] [41] [43], and is an active area of research.

We simply plug these approximation definitions into the weak form (4.4), resulting in the linear system of equations:

$$\begin{aligned}
&\sum_{i=1}^{n}\left[u_i\cdot\int_\Omega -\mu\nabla\phi_i:\nabla\psi_j\,d\hat{x}+p_i\int_\Omega -\psi_j\cdot\nabla\phi_{pi}\,d\hat{x}\right]=\int_\Omega \rho g\cdot\psi_j+\mu\nabla\phi_\Gamma:\nabla\psi_j\,d\hat{x},\\
&\sum_{i=1}^{n}u_i\cdot\int_\Omega -\phi_i\cdot\nabla q_j\,d\hat{x}=\int_\Omega \phi_\Gamma\cdot\nabla q_j\,d\hat{x},\qquad j=1,\cdots,n.
\end{aligned}\tag{4.9}$$



Chapter 5

A semi-Lagrangian mixed finite element method for the incompressible Navier-Stokes equations

5.1 Extending the steady Stokes flow method to unsteady Navier-Stokes flow

To extend the the previously described mixed finite element method to the general incompressible Navier-Stokes equations, we must introduce time dependence and advection. We will focus on the Navier-Stokes equations in a closed cavity, with $\rho = 1$, and with zero initial condition and a no-slip boundary condition everywhere. The equation is then

$$\begin{aligned} \frac{\partial u}{\partial t} + u \cdot \nabla u &= \nu \Delta u - \nabla p + \rho g, \\ \nabla \cdot u &= 0, \\ u(\hat{x}, 0) = 0, \quad u|_\Gamma &= 0. \end{aligned} \tag{5.1}$$

Time dependence is simple. Due to the diffusive nature of the Stokes equation, a viable option for time discretization (to maintain stability) is implicit Euler. The discretized time-dependent Stokes equations are then

$$\begin{aligned} \int_\Omega \frac{u - u_{\text{prev}}}{\Delta t} \cdot \psi^u \, d\hat{x} &= \int_\Omega \nu \nabla u : \nabla \psi^u - p \nabla \cdot \psi^u + \rho g \cdot \psi^u \, d\hat{x}, \\ \int_\Omega -\psi^p \nabla \cdot u &= 0 \qquad \text{for all } \psi^u \in \Psi^u \text{ and } \psi^p \in \Psi^p. \end{aligned} \tag{5.2}$$

The last step is to introduce advection: the term $u \cdot \nabla u$. There are many choices.

Fully implicit Euler with Newton iterations

If full implicit Euler is used, and the nonlinear advection term is expanded, the result is a nonlinear system of equations. The simplest option is then a Newton iteration, solving a sequence of linear systems after computing the Gateaux derivative of the system at each step. This fully implicit method may be highly accurate, but requires a not-so-straightforward modification of the Stokes flow code.

A mixed implicit-explicit (IMEX) method

An alternative method is IMEX, where $u \cdot \nabla u$ is instead $u_{\text{prev}} \cdot \nabla u_{\text{prev}}$ in the implicit Euler step. This gives a stable implicit momentum diffusion, with a fast, explicit advection step. There are, however, still difficulties in the accurate computation of $u_{\text{prev}} \cdot \nabla u_{\text{prev}}$ projected onto the P2 finite element space.

5.1.1 A splitting method with semi-Lagrangian advection

The difficulties of the above two methods can be avoided by performing a splitting method. The essence of the difficulty of solving a fully hyperbolic advection equation lies in the motion of solution data along *characteristic curves*. The diffusive component of the Navier-Stokes equations, however, makes the system parabolic: there are no well-defined characteristic curves. If we instead alternate between the inertial advective motion of the fluid, and then apply those non-advective forces (viscosity, body forces, and pressure), then this first step can use the method of characteristics. The simplest such method is *semi-Lagrangian advection*, which is common in meteorology and computer graphics [45]. The resulting split equations are

$$\begin{aligned} \frac{u^* - u_{\text{prev}}}{\Delta t} &= -u_{\text{prev}} \cdot \nabla u_{\text{prev}}. \\ \frac{u - u^*}{\Delta t} &= \nu \nabla u - \nabla p + \rho g. \end{aligned} \tag{5.3}$$

The second step can be solved with the time-step Stokes flow equation described previously. For the first, advective, step we can use the method of characteristics. Given that u is sampled at vertices and midpoints in the triangulation, the characteristic curve at a node $\hat{x}$ is tangent to $u(\hat{x})$. An explicit Euler step will simply "shift" the node to $\hat{x} + \Delta t u(\hat{x})$, with the same velocity (due to inertia). This is the Lagrangian perspective, and will require either modification of the mesh, or reprojection of the "shifted" finite element spaces onto the non-shifted finite element space. The alternative, *semi*-Lagrangian method, is to sample the velocity *backward* along the characteristic curve,

$$u(\hat{x}) \leftarrow u(\hat{x} - \Delta t u(\hat{x})).$$

Given the no-slip boundary condition, for small enough Δt it seems reasonable to say that $u(\hat{x} - \Delta t u_{\text{prev}}(\hat{x}))$ is 0 when $\hat{x} - \Delta t u(\hat{x})$ lies outside of the domain. The resampling is done with the exact u_{prev} reconstructed by quadratic basis functions. This can be achieved by either mesh traversal, to locate the triangle containing $\hat{x} - \Delta t u_{\text{prev}}(\hat{x})$, followed by combination of the quadratic triangle basis functions, or by densely sampling u_{prev} into a fine grid, which is used as a lookup table through linear filtering. Given that the domain is 2D, this second option can be done extremely quickly by triangle rasterization (function sampling) on a modern GPU. This avoids the necessity of additional data structures for mesh search.

5.2 Results

I have implemented the semi-Lagrangian method, as described above, by two small extensions of the Stokes flow method of chapter 5 (timestepping and split semi-Lagrangian advection).

The sample problem

As a specific test case, the kinematic viscosity is set to $\nu = 0.001$, and the domain Ω is taken as the square with circular obstruction

$$\Omega = [-1, 1]^2 - D,$$

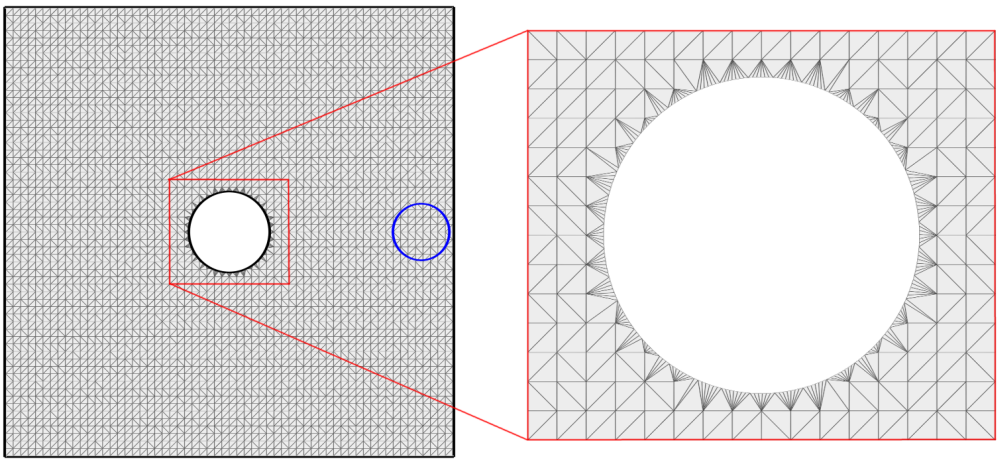
where D is the disk of radius 0.18 centred at $\hat{x} = (0, 0)$. The initial condition and boundary conditions are the simplest possible: 0 initial velocity, and no-slip everywhere. The source function g is what introduces motion into the system. It is defined as

$$g(\hat{x}) = \begin{cases} (-3000, 0) & \text{if } \hat{x} \in D_g, \\ (0, 0) & \text{otherwise,} \end{cases}$$

where D_g is a disk of radius 0.125 centred at roughly $\hat{x} = (0.85, 0)$. This emulates a sort of inflow condition at the right-hand-side, without the necessity of handling non-trivial boundary conditions.

The discretization

The base square mesh is 60×60, with velocity sampled at additional midpoints. This mesh, along with the source disk, is displayed in figure **??**. The circular obstruction boundary is approximated naively with many fans of thin triangles, which, thankfully, the finite element method can handle with ease.



The time step is constant, chosen as $\Delta t = 1/300$. A total of 900 time steps have been computed, from $t = 0$ to $t = 3$. This computation took 14 hours on a 2012 laptop, and the code is very unoptimized — however, the major bottleneck is by far the sparse linear solve, so the speed of this algorithm depends almost entirely on a fast sparse saddle point linear solver and an effective preconditioner. The advection step is effectively free in comparison (interactive-speed advection versus a minute for the solve on my machine). The resulting sparse matrix is 31895×31895, with a fill of approximately 0.0003845/1.

Visualisation of the results

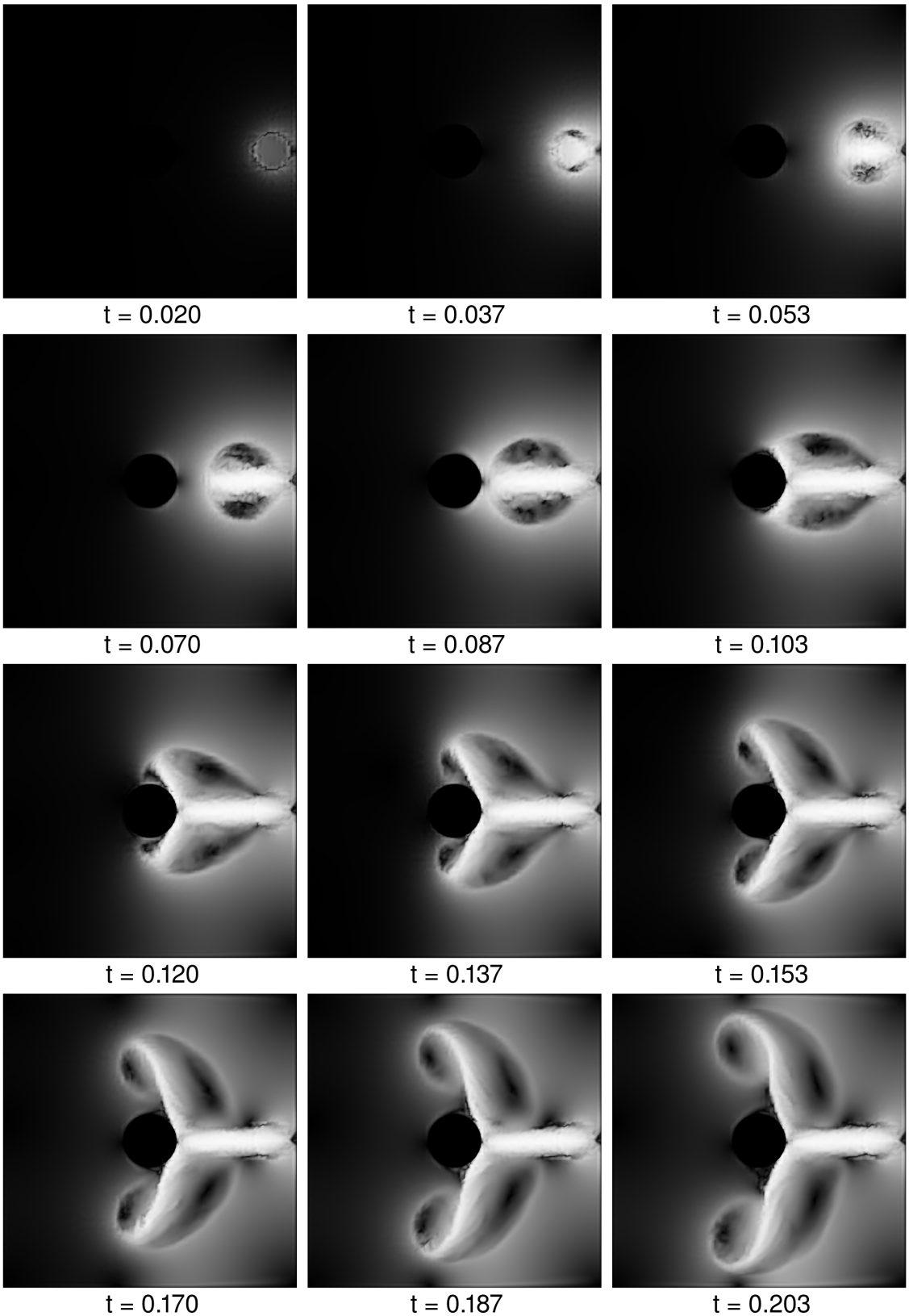
Figures **??** and **??** display the resulting solution at varying times. A video of the solution (at roughly 1/10 the speed) is available online (`https://youtu.be/m5He2x3gNz8`). The

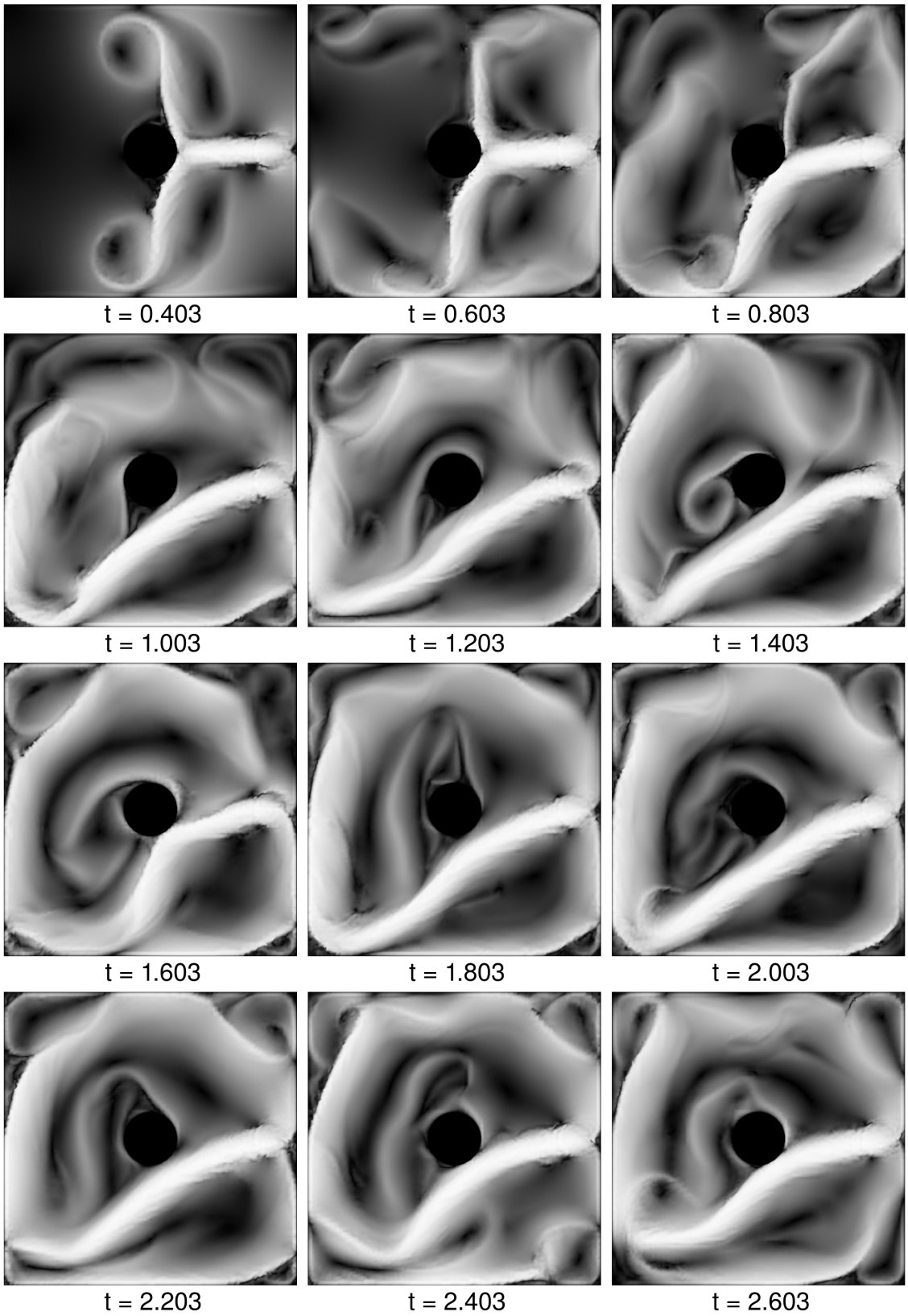
average fluid speed is roughly $\|u\| = 6$. The image colour ranges from 0 (black) to 1 (white), determined by the function

$$\text{Colour}(\hat{x}) = 1 - \exp(-0.1 \cdot \|u\|).$$

For example, the source flow is bright white as it has the highest speed, while the centres of vortices are almost pitch black.

The initial flow shown in **??**, as expected, splits symmetrically at the obstruction into two sets of coupled vortices, which will then split as they hit the walls in **??**. Subsequently, in figure **??** (which displays a larger range of time steps), symmetry breaks down, and the source flow deflects downward. The solution past this point cannot be considered as anywhere close to the real solution, due to the onset of turbulence in this medium-Reynolds-number flow. However, important qualitative effects (for example incompressibility and oppositely-oriented vortices being pushed against the corners of the cavity) are apparent. The incompressibility condition is in fact held everywhere with a maximum absolute error of at most 10^{-3}, usually on the order of 10^{-4}.


t = 0.020
t = 0.037
t = 0.053
t = 0.070
t = 0.087
t = 0.103
t = 0.120
t = 0.137
t = 0.153
t = 0.170
t = 0.187
t = 0.203


t = 0.403
t = 0.603
t = 0.803
t = 1.003
t = 1.203
t = 1.403
t = 1.603
t = 1.803
t = 2.003
t = 2.203
t = 2.403
t = 2.603

Bibliography

[1] Isaac Newton, *Philosophiae Naturalis Principia Mathematica (Third edition)*, 1726.

[2] Johann Bernoulli, *"Problema novum ad cujus solutionem Mathematici invitantur." (A new problem to whose solution mathematicians are invited.)*, 1696. (retrieved from wikipedia/brachistochrone_curve)

[3] A. F. Monna, *Dirichlet's principle: A mathematical comedy of errors and its influence on the development of analysis*, 1975.

[4] Stig Larsson, *Partial differential equations with numerical methods*, 2003.

[5] Peter Lax, *Hyperbolic Systems of Conservation Laws and the Mathematical Theory of Shock Waves*, 1973.

[6] Cornelius Lanczos, *The Variational Principles of Mechanics*, 1952.

[7] G. K. Batchelor, *Introduction to Fluid Dynamics*, 1967.

[8] L. Gary Leal, *Advanced Transport Phenomena: Fluid Mechanics and Convective Transport Processes*, 2007.

[9] Vivette Girault, Pierre-Arnaud Raviart, *Finite Element Methods for Navier-Stokes equations*, 1986.

[10] Richard Feynman, *Surely You're Joking, Mr. Feynman!*, 1985.

[11] V.I. Arnol'd, *Mathematical Methods of Classical Mechanics*, 1978.

[12] Alan Turing, *The Chemical Basis of Morphogenesis*, 1952.

[13] *The Princeton Companion to Applied Mathematics*, 2015.

[14] Mario Botsch, Leif Kobbelt, Mark Pauly, Pierre Alliez, Bruno Lévy, *Polygon Mesh Processing*, 2010.

[15] M. Meyer, M. Desbrun, P. Schroder and A.H. Barr, *Discrete Differential-Geometry Operators for Triangulated 2-Manifolds*, 2003.

[16] P. G. Ciarlet, *The Finite Element Method for Elliptic Problems*, 1978.

[17] Daniel Cremers, *Variational Methods in Computer Vision*, (https://vision.in.tum.de/teaching/online/cvvm)

[18] Lawrence Evans, *Partial Differential Equations*, 2010.

[19] Documentation for DOLFIN-1.5.0 (Python), (https://fenicsproject.org/olddocs/dolfin/1.5.0/python/index.html)

[20] Anders Logg, Kent-Andre Mardal, Garth N. Wells (editors), *The FEniCS book*, 2012.

[21] Hans Petter Langtangen, Anders Logg, *Solving PDEs in Python: The FEniCS tutorial I*, 2016.

[22] E. Ward Cheney, *Introduction to Approximation Theory*, 1966.

[23] Jos Stam, *Flows on surfaces of arbitrary topology*, 2003.

[24] David Ham, Finite Element Course (Imperial College London, 2013-2014), http://wp.doc.ic.ac.uk/spo/finite-element/

[25] Howard Elman, David Silvester, Andy Wathen, *Finite Elements and Fast Iterative Solvers, with Applications in Incompressible Fluid Dynamics, 2nd edition*, 2014.

[26] Gilbert Strang, *A Framework for Equilibrium Equations*, 1988.

[27] Susanne C. Brenner, L. Ridgway Scott, *The Mathematical Theory of Finite Element Methods*, 2008.

[28] Gene H. Golub, Charles F. Van Loan, *Matrix Computations, third edition*, 1996.

[29] Jonathan Richard Shewchuk, *Triangle: Engineering a 2D Quality Mesh Generator and Delaunay Triangulator*, 1996.

[30] Hang Si, *TetGen: A quality tetrahedral mesh generator and a 3D Delaunay triangulator*, 2015.

[31] Joseph O'Rourke, *Computational Geometry in C*, 1998.

[32] Balay et al., PETSc web page, https://petsc.org/, 2021.

[33] Conrad Sanderson, Ryan Curtin, *Armadillo: a template-based C++ library for linear algebra*, 2016.

[34] Gaël Guennebaud, *Eigen: A C++ linear algebra library*, 2013.

[35] Nicholas Sharp, Keenan Crane, and others, Geometry Central (surface mesh library), www.geometry-central.net, 2019.

[36] Daniel Sieger, Mario Botsch, The Polygon Mesh Processing Library, http://www.pmp-library.org, 2020.

[37] Leif Kobbelt, OpenMesh (surface mesh library), https://www.graphics.rwth-aachen.de/software/openmesh/, 2021.

[38] Sergey Repin, *One hundred years of the Galerkin method*, 2017.

[39] Boris Galerkin, *Beams and plates. Series in some questions of elastic equilibrium of beams and plates (In Russian)*, 1915.

[40] Sobolev Institute of Mathematics, Sergei Sobolev, biographical web page, http://www.math.nsc.ru/conference/sobolev/english/About_Sobolev_SL.htm

[41] Florian Rathgeber, David A. Ham, and others, *Firedrake: automating the finite element method by composing abstractions*, 2016.

[42] A. Meurer, C.P. Smith., and many others, *SymPy: symbolic computing in Python*, 2017.

[43] Anders Logg, *Automating the Finite Element Method*, 2006.

[44] FEniCS Dolfin documentation: Stokes equations with Taylor-Hood elements, https://fenicsproject.org/olddocs/dolfin/1.5.0/python/demo/documented/stokes-taylor-hood/python/documentation.html

[45] Jos Stam, *Stable fluids*, —