Numerical Methods for (Partial) Differential Equations

Shizheng Wen

shiwen@student.ethz.ch

Table of contents

Week 0

1 Introduction

1.1 General Introduction

1.1.1 Goals

Flavor of this course

This course combines *emphasis on algorithms* with their sound mathematical motivation and derivation, rigorous, but not too formal.
It aims to impart an "*intuitive understanding*" of numerical methods, their properties, potential, and limitations.

The main *skills* to be acquired in this course are the following:

- Ability to *implement* advanced numerical methods for the solution of partial differential equations in C++ efficiently (based on numerical libraries, of course)
- Ability to *modify* and *adapt* numerical algorithms guided by awareness of their mathematical foundations
- Ability to *select* and *assess* numerical methods in light of the predictions of theory
- Ability to *identify features* of a PDE (= partial differential equation) based model that are relevant for the selection and performance of a numerical algorithm
- Ability to *understand research publications* on theoretical and practical aspects of numerical methods for partial differential equations.

1.1.2 Topic

The better title to describe this course is "numerical methods for the simulation of continuum models with local interactions."

- Continuum ≡ Unknown are functions/fields ∈ ∞-dimensional function space. $\mathbf{u}(t) \rightarrow$ ODE; $u(x,y,t) \rightarrow$PDE
  - ∞ refers to the spatial dimension. In temporal dimension, we can view it as continuous.
- Model ≡ mathematical description of some aspects of reality. Here mathematical description refers to PDE-based. (we can also see some statistical-based descriptions of reality)

⇒ PDE embedded in physcial context: encodes **strutural properties** → should be respected/explained by a numerical method ("structure-preserving"). See chapter 4.8 This means that different PDEs have its own properties, our numerical algorithms is specificed for certain type of PDEs.

Example: Fluid mechanics

(Stationary, incompressible) Navier-Stokes equations:

$$\begin{aligned} -\nu\Delta\mathbf{u} + D\mathbf{u}\cdot\mathbf{u} + \mathbf{grad}\, p &= \mathbf{f} \quad \text{in } \Omega, \\ \mathbf{div}\,\mathbf{u} &= 0 \quad \text{in } \Omega, \\ \mathbf{u} &= 0 \quad \text{on } \partial\Omega. \end{aligned} \tag{0.4.3.1}$$

ν ≙ dynamic viscosity
$\mathbf{f}: \Omega \mapsto \mathbb{R}^3$ ≙ given external force field
$\mathbf{u}: \Omega \mapsto \mathbb{R}^3$ ≙ velocity field (unknown)
$p: \Omega \mapsto \mathbb{R}$ ≙ pressure (unknown)

If the convective term $D\mathbf{u}\cdot\mathbf{u}$ is omitted we obtain the Stokes-equations, see Chapter 12

- Here, $\operatorname{div} \mathbf{u} = 0$ refers to incompressibility; $-\nu\Delta\mathbf{u}$ refers to dissipation.

1.1.3 Contents and Outline

Model classes (determined by structural properties) covered in this course:

1. Equilibrium models

   Elastic membranes, electrostatics, stationary heat conduction

   (a) Chapter I: Relevant theory

   (b) Chapter II: The finite element method (FEM)

   (c) Chapter III: Theory of the FEM

   (d) Chapter IV: Nonlinear equilibrium models

2. Evolution models

   Heat equation, wave equation, Schrödinger equation

   (a) Chapter IX: Second-order linear evolution problems

3. Transport models

   (a) Chapter X: Convection-Diffusion Problems

   (b) Chapter XI: Numerical methods for conservation laws

1.2 Coding projects

C++ libraries and building system relied on this course:

- Eigen
- LehrFEM++
- cmake

1.3 Prerequisites for this Course

1.3.1 Tools from the Linear Algebra

These basic concepts include vector spaces (a set with operation like vector addition and scalar multiplication satisfying vector axioms), in particular $\mathbb{R}^n$ and $\mathbb{C}^n$, $n \in \mathbb{N}$, subspaces, affine spaces, linear independence, bases, matrices and vectors, and linear systems of equations. Recalling the definitions not in the limelight of linear algebra.

Affine Spaces

Definition 0.3.1.1. Affine (sub)space

Given a vector space V, a subspace $V_0 \subset V$, and some $\widehat{v} \in V$, we call the set

$$\widehat{V} := \widehat{v} + V_0 := \{v = \widehat{v} + v_0 : \ v_0 \in V_0\} \subset V$$

an **affine (sub)space** (of V).

- Also referred as linear manifold. 是数学中的几何机构，这种结构式是欧氏空间仿射特性的推广。在仿射空间中，点与点之间做差可以得到向量，点与向量做加法将得到另一个点，但是点与点之间不可以做加法。

- 仿射空间中没有特定的原点，因此不能将空间中的每一点和特定的向量对应起来。仿射空间中只有从一个点到另一个点的位移向量，或称平移向量。

Let us also recall, what you should know about bases:

Definition 0.3.1.2. Basis of a finite dimensional vector space

Let V be a real vector space. A finite subset $\{b^1,\dots,b^N\} \subset V, N \in \mathbb{N}$, is a basis of V, if for *every* $v \in V$ there are *unique* coefficients $\mu_\ell \in \mathbb{R}$, $\ell \in \{1,\dots,N\}$, such that $v = \sum_{\ell=1}^{N} \mu_\ell b^\ell$. Then N agrees with the dimension of V.

1.3.1.1 Basic Notations

By default, all vectors in $\mathbb{R}^n$ or $\mathbb{C}^n$ are regarded as column vectors. We write $\mathbf{a}, \mathbf{b}, \dots$ for "small vector" $\in \mathbb{R}^n$, which often contain the coordinates of points. Bold capital letters like $\mathbf{A}, \mathbf{B}, \dots$ are reserved for matrices $\in \mathbb{R}^{n,m}$ or $\mathbb{C}^{n,m}$.

NOTE: "Large" vectors of basis expansion coefficients $\in \mathbb{R}^N, N \in \mathbb{N}$, are designated by $\vec{\mu}, \vec{\eta}, \dots$. Vector components are usually singled out as follows, which indices invariably starting from 1:

$$\vec{\mu} = [\mu_1\, \mu_2\, \mu_3\, \cdots\, \mu_N]^\top \in \mathbb{R}^N \quad \blacktriangleright \quad (\vec{\mu})_j := \mu_j\,, \quad j = 1,\dots,N\,.$$

Special vectors are the zero vector $\mathbf{0} \in \mathbb{R}^N$, the "large" unit vectors $\vec{\epsilon}^k \in \mathbf{R}^n, k \in \{1,\dots,n\}$, defined by

$$\left(\vec{\epsilon}^k\right)_j = \delta_{jk} := \begin{cases} 1 & \text{, if } j = k\,, \\ 0 & \text{, if } j \neq k \end{cases} \qquad \text{(Kronecker symbol)},$$

and the column vector $\mathbf{1}$ with entries all $= 1$.

"Large" vectors refer to coefficients of the basis expansion in this vector space which means that any vector is this vector can be defined as long as we define our basis. "Large" unit vectors specify a class of vectors which is the basis of the vector space and expressed by expansion coefficients.

dot product notation: $\vec{\mu}\cdot\vec{\eta} := \vec{\mu}^\top\vec{\eta} \in \mathbb{R}\,, \quad \vec{\mu},\vec{\eta} \in \mathbb{R}^N$

1.3.1.2 Linear and Bilinear Forms

Definition 0.3.1.3. Linear forms

Given a vector space V over $\mathbb{R}$, a linear form/linear functional (LF) ℓ is a mapping $\ell : V \mapsto \mathbb{R}$ that satisfies

$$\ell(\alpha u + \beta v) = \alpha\ell(u) + \beta\ell(v) \quad \forall u, v \in V\,, \forall \alpha, \beta \in \mathbb{R}\,.$$

- 由于其称之为linear functional,因此我们可以看作函数的函数。也就是从一个函数映射到一个scalar。这里的vector space满足linear axiom，其本质上可以是一个函数空间。

Linear algebra also examines multi-linear forms, mappings from $V \times V \times \cdots \to \mathbb{R}$, which are linear in each argument.

Definition 0.3.1.4. (Bi-)linear forms

Given an $\mathbb{R}$-vector space V, a A bilinear form (BLF) a on V is a mapping $\mathsf{a} : V \times V \mapsto \mathbb{R}$, for which

$$\mathsf{a}(\alpha_1 v_1 + \beta_1 u_1, \alpha_2 v_2 + \beta_2 u_2) = \alpha_1\alpha_2\,\mathsf{a}(v_1, v_2) + \alpha_1\beta_2\,\mathsf{a}(v_1, u_2) + \beta_1\alpha_2\,\mathsf{a}(u_1, v_2) + \beta_1\beta_2\,\mathsf{a}(u_1, u_2)$$

for all $u_i, v_i \in V, \alpha_i, \beta_i \in \mathbb{R}, i = 1, 2$.

✏️: notation: For bilinear forms we write a(· , ·), b(· , ·), etc; note the special font. Linear form is just a mapping from vector space to real space $\mathbb{R}$. Linear form is from 1 vector and bilinear form is from 2 vectors to the real space.

- 给人的感觉有点像是二元函数。

Examples for linear forms and bilinear forms:

EXAMPLE 0.3.1.5 (Linear forms on $\mathbb{R}^n$) For a fixed column vector $\vec{\alpha} \in \mathbb{R}^n$, $n \in \mathbb{N}$, the mapping

$$\mathbb{R}^n \to \mathbb{R} \quad , \quad \vec{\xi} \mapsto \vec{\alpha}^\top \vec{\xi} \, , \tag{0.3.1.6}$$

is a linear form on $\mathbb{R}^n$. The converse is also true: every linear form ℓ on $\mathbb{R}^n$ can be written in the form (0.3.1.6) with some vector $\vec{\alpha} \in \mathbb{R}^n$. It components are given by letting ℓ act on the unit vectors: $(\vec{\alpha})_j = \ell(\vec{e}^j), j = 1, \ldots, n$ ⌟

EXAMPLE 0.3.1.7 (Bilinear forms on $\mathbb{R}^n$) For any square matrix $\mathbf{A} \in \mathbb{R}^{n,n}$, $n \in \mathbb{N}$, the mapping

$$\mathbb{R}^n \times \mathbb{R}^n \to \mathbb{R} \quad , \quad (\vec{\xi}, \vec{\eta}) \mapsto \vec{\xi}^\top \mathbf{A} \vec{\eta} \, , \tag{0.3.1.8}$$

represents a bilinear form on $\mathbb{R}^n$. In fact, for every bilinear form a on $\mathbb{R}^n$ we can find a square matrix $\mathbf{A} \in \mathbb{R}^{n,n}$ such that $\mathsf{a}(\vec{\xi}, \vec{\eta}) = \vec{\xi}^\top \mathbf{A} \vec{\eta}$. The entries of $\mathbf{A}$ can be computed by plugging unit vectors into a

$$\mathsf{a}(\vec{\xi}, \vec{\eta}) = \vec{\xi}^\top \mathbf{A} \vec{\eta} \quad \Leftrightarrow \quad (\mathbf{A})_{i,j} = \mathsf{a}(\vec{e}^i, \vec{e}^j) \, , \quad i, j \in \{1, \ldots, n\} \, .$$

✏️: Thus, from the previous two examples we learn that, with respect to a chosen basis, every linear form on V can be represented by a column vector and every bilinar form on V allows the description by a square matrix. This means that every linear mapping can be represented by a column vector and a matrix (Note: this matrix should be a square matrix). Very important concept!

what if our vector space has infinite dimensions?

Remark-Linear forms from bilinear forms:

Let a be a binlinear form on $\mathbf{R}^n$. Let us fix a vector $\vec{\xi} \in \mathbb{R}^n$. Then the mapping$\mathbb{R}^n \to \mathbb{R} \quad , \quad \vec{\eta} \mapsto \mathsf{a}(\vec{\xi}, \vec{\eta})$ is a linear form on $\mathbb{R}^n$.

1.3.1.3 Norms and Inner Products

In this chapter, we introduce the basic concept of norm at first. Then, by defining inner product or scalar product (s.p.d bilinear form on a real vector space V , we defined the norm on the real vector space V again. Furthermore, in this chpater, we introduced definite of matrix by the one-to-one relationship between matrix and bilinear form.

The concept of norm generalizes the length of a vector in the vector space. It is just a mapping from $V \to \mathbb{R}_0^+$. If this mapping satisfies (N2) and (N3), but fails on (N1), then it is called a semi-norm.

Definition 0.3.1.10. Norm (on a vector space) → [Hip19, ??]

A **norm** $\|\cdot\|_V$ on an $\mathbb{R}$-vector space V is a mapping $\|\cdot\|_V : V \mapsto \mathbb{R}_0^+$, such that

(definiteness)	$\|v\|_V = 0 \iff v = 0 \quad \forall v \in V$	(N1)
(homogeneity)	$\|\lambda v\|_V = \|\lambda\| \|v\|_V \quad \forall \lambda \in \mathbb{R}, \forall v \in V \, ,$	(N2)
(triangle inequality)	$\|w + v\|_V \le \|w\|_V + \|v\|_V \quad \forall w, v \in V \, .$	(N3)

Norms on $\mathbb{R}^n$ The most important norms on the vector space $\mathbb{R}^n$ are the

$$\text{maximum norm:} \quad \left\|\vec{\xi}\right\|_\infty := \max\left\{\left|\left(\vec{\xi}\right)_j\right|,\ j=1,\ldots,n\right\}, \tag{0.3.1.12}$$

$$\text{Euclidean norm:} \quad \left\|\vec{\xi}\right\|_2 := \left(\sum_{j=1}^n \left|\left(\vec{\xi}\right)_j\right|^2\right)^{\frac{1}{2}}, \tag{0.3.1.13}$$

$$\text{1-norm:} \quad \left\|\vec{\xi}\right\|_1 := \sum_{j=1}^n \left|\left(\vec{\xi}\right)_j\right|. \tag{0.3.1.14}$$

The Euclidean norm is sometimes called 2-norm. Frequently, for small vectors, we drop the subscript 2 from the notation for the Euclidean norm.

Norms spawned by inner products

We will mainly be concerned with norms spawned by inner products, which are defined next.

Definition 0.3.1.15. Symmetric bilinear form

A bilinear form (→ Def. 0.3.1.4) a on a vector space V is symmetric, if and only if

$$\mathsf{a}(u,v) = \mathsf{a}(v,u) \quad \forall u,v \in V.$$

- 也就是定义的这种二元运算和vectors的位置无关，等效在矩阵上就是对称矩阵。

Important classes of bilinear forms "have signs":

Definition 0.3.1.16. Positive (semi-)definite bilinear form

A bilinear form $a : V \times V \to \mathbb{R}$ on a real vector space V is called

$$\text{positive semi-definite} \quad :\Leftrightarrow \quad \mathsf{a}(v,v) \geq 0 \quad \forall v \in V,$$
$$\text{positive definite} \quad :\Leftrightarrow \quad \mathsf{a}(v,v) > 0 \quad \forall v \in V \setminus \{0\}.$$

The one-to-one relationship of square matrices and bilinear forms on $\mathbb{R}^n$ provides the notion of being positive definite for matrices. Introducing the definite of matrix by bilinear form.

Definition 0.3.1.17. Positive definite matrix

A square matrix $\mathbf{A} \in \mathbb{R}^{n,n}$ is positive definite, if

$$\vec{\eta}^\top \mathbf{A}\vec{\eta} > 0 \quad \text{for all} \quad \vec{\eta} \in \mathbb{R}^n \setminus \{0\}.$$

- We abbreviate the property of a bilinear form or square matrix to be symmetric and positive definite at the same time as **s.p.d.**.

Definition 0.3.1.18. Inner product

A symmetric positive definite (s.p.d.) bilinear form on a real vector space V is also called an inner product or scalar product on V.

- 在数学上，内积空间是增添了一个额外的结构的矢量空间。这个额外的结构叫做内积或标量积。这个增添的结构将一对矢量与一个纯量连接起来，允许我们严格地谈论矢量的“夹角”和“长度”，并进一步谈论矢量的正交性。内积空间由欧几里得空间抽象而来（内积是点积的抽象），这是泛函分析讨论的课题。dot product算是一种inner product。这里讨论的inner product是一种更广义的存在。
- 内积算是一种特殊的bilinear form，一种在自身vector space上的二元运算。

⇓ Inner products allow the following **famous elementary estimate**

Theorem 0.3.1.19. Cauchy-Schwarz inequality

If a is a symmetric positive semi-definite bilinear form on the real vector space V, then

$$|\mathsf{a}(u,v)| \leq \mathsf{a}(u,u)^{\frac{1}{2}}\,\mathsf{a}(v,v)^{\frac{1}{2}}. \tag{0.3.1.20}$$

⇓ Introduce the **Norms from inner products**

Theorem 0.3.1.21. Norms from inner products

If a *is an inner product (= symmetric positive definite bilinear form, Def. 0.3.1.18) on the real vector space* V*, then*

$$\|\cdot\|_a : V \to \mathbb{R} \quad , \quad \|v\| := \mathsf{a}(v,v)^{\frac{1}{2}} , \tag{0.3.1.22}$$

defines a norm (→ Def. 0.3.1.10) on V*.*

- Usually, the norm is a linear form. Here, we used bilinear form to define such linear form. And this bilinear form should be the inner product (s.p.d.).
- 就是基于内积定义范数的概念。内积空间允许我们严格讨论矢量的“夹角”和“长度”。

On $\mathbb{R}^n$ square matrices and bilinear forms are in a one-to-one relationship. Thus properties of bilinear forms induce corresponding properties of matrices:

Definition 0.3.1.24. Symmetric positive definite matrices

A matrix $\mathbf{A} \in \mathbb{R}^{n,n}$ is symmetric positive definite (s.p.d.), if

$$\mathbf{A}^\top = \mathbf{A} \quad \text{and} \quad \vec{\xi}^\top \mathbf{A}\vec{\xi} > 0 \quad \forall \vec{\xi} \in \mathbb{R}^n \setminus \{\mathbf{0}\} .$$

- 向量空间上面定义的任何运算都可以用矩阵来进行表述。

1.3.1.4 Diagonalization of Matrices

Definition: $\mathbf{AS} = \mathbf{SD}$, $\mathbf{D} \in \mathbb{R}^{n,n}$ diagonal , $\mathbf{S} \in \mathbb{R}^{n,n}$ regular/invertible.

- Meaning: Diagonalization is a central concern in linear algebra. It amounts to finding a basis of $\mathbb{R}^n$ such that $\mathbf{A} \in \mathbb{R}^{n,n}$ becomes diagonal with respect to this basis.

In general real matrices cannot be diagonalized according to above definition, but often they can be diagonalized in $\mathbb{C}$. There is one important class of matrices for which diagonalization in $\mathbb{R}$ is always possible:

Theorem 0.3.1.26. Real diagonalization of symmetric matrices

For every symmetric matrix $\mathbf{A} \in \mathbb{R}^{n,n}$*, that is,* $\mathbf{A}^\top = \mathbf{A}$*, we can find a diagonal matrix* $D \in \mathbb{R}^{n,n}$ *and an orthogonal matrix* $\mathbf{Q} \in \mathbb{R}^{n,n}$ *such that*

$$\mathbf{Q}^\top \mathbf{A}\mathbf{Q} = \mathbf{D} . \tag{0.3.1.27}$$

- This result can be extended to more general inner products: If $\mathbf{A} = \mathbf{A}^\mathbf{T} \in \mathbb{R}^{n,n}$ and $\mathbf{B} \in \mathbf{R}^{n,n}$ is s.p.d., then we can find a regular matrix $\mathbf{S} \in \mathbb{R}^{n,n}$ such that $\mathbf{AS} = \mathbf{BSD}$, $\mathbf{D}$ diagonal , $\mathbf{S}^\top\mathbf{BS} = \mathbf{I}$

1.3.2 Tools from Real Analysis

Recall the definition of Landau symbols ("O"-notation)

1.3.2.1 Calculus in 1D

1D Taylor formula with integral remainder term

Theorem 0.3.2.2. Taylor's theorem

If $f :]a,b[\to \mathbb{R}$, $a < b$, is $k+1$-times continuously differentiable and $x \in]a,b[$, then

$$f(x+h) = f(x) + f'(x)h + \tfrac{1}{2}f''(x)h^2 + \cdots + \frac{1}{k!}f^{(k)}(x)h^k + R(x;h)\,, \tag{0.3.2.3a}$$

with integral remainder term

$$R(x;h) = \int_x^{x+h} \frac{f^{(k+1)}(\xi)}{k!}(x-\xi)^k\,d\xi\,, \tag{0.3.2.3b}$$

for all h: $x+h \in]a,b[$.

- In NumCSE, we learnt that the domain of convergence for taylor series is limited.
- The reminder term can be written in Lagrange form:
  $R(x;h) = \dfrac{f^{(k+1)}(\zeta)}{(k+1)!}h^{k+1}$ for some $\zeta = \zeta(x;h) \in [\min\{x, x+h\}, \max\{x, x+h\}]$.

When we are solely interested in the decay of the remainder term as $h \to 0$, we often write:

$$f(x+h) = f(x) + f'(x)h + \tfrac{1}{2}f''(x)h^2 + \cdots + \frac{1}{k!}f^{(k)}(x)h^k + O(h^{k+1}) \quad \text{for} \quad h \to 0\,.$$

1.3.2.2 Differentiation in Multiple Dimensions

Partial derivatives

A fundamental result about higher order partial derivatives is that their order does not matter in general.

Theorem 0.3.2.14. Partial derivatives commute

If all second partial derivatives of $f : \Omega \subset \mathbb{R}^d \to \mathbb{R}^n$ are continuous functions on the open domain Ω, then

$$\frac{\partial^2 f}{\partial x_i \partial x_j}(x) = \frac{\partial^2 f}{\partial x_j \partial x_i}(x)\,, \quad 1 \le i,j \le d\,, \quad x \in \Omega\,.$$

The assertion of the theorem generalizes to all high-order partial derivatives:

$$\frac{\partial^j u_i}{\partial x_{k_1} \ldots \partial x_{k_j}}(x)\,, \quad k_\ell \in \{1,\ldots,d\}\,, k_\ell \in \{1,\ldots,j\}$$

is invariant with respect to permutations of $k_1,\ldots,k_j$, if u is at least j-times continuously differentiable.

Jacobian

For a differentiable function $u : D \subset \mathbb{R}^n \to \mathbb{R}^d$its derivative $Du(\mathbf{x})$ has matrix representation, the so-called Jacobian:

$$\mathsf{D}u(\boldsymbol{x}) \;\leftrightarrow\; Du(\boldsymbol{x}) = \left[\frac{\partial u_i}{\partial x_j}(\boldsymbol{x})\right]_{i,j=1}^{n} = \begin{bmatrix} \frac{\partial u_1}{\partial x_1}(\boldsymbol{x}) & \frac{\partial u_1}{\partial x_2}(\boldsymbol{x}) & \cdots & \cdots & \frac{\partial u_1}{\partial x_n}(\boldsymbol{x}) \\ \frac{\partial u_2}{\partial x_1}(\boldsymbol{x}) & & & & \frac{\partial u_2}{\partial x_n}(\boldsymbol{x}) \\ \vdots & & & & \vdots \\ \frac{\partial u_n}{\partial x_1}(\boldsymbol{x}) & \frac{\partial u_n}{\partial x_2}(\boldsymbol{x}) & \cdots & \cdots & \frac{\partial u_n}{\partial x_n}(\boldsymbol{x}) \end{bmatrix} \in \mathbb{R}^{n,n}$$

- Because PDE must have multivariable for every solution. Therefore, the symbol Du will be seen very often.

Hessian

A matrix reprsentation can also be given for the second derivative of a real-valued function $u : D \subset \mathbb{R}^d \to \mathbb{R}$.It is known as Hessian $\mathsf{H}u(\boldsymbol{x}) = \mathsf{D}^2u(\boldsymbol{x}) = \left[\frac{\partial^2 u}{\partial x_i \partial x_j}(\boldsymbol{x})\right]_{i,j=1}^{d} \in \mathbb{R}^{d,d}$

- 一个重要的概念就是无论是ODE还是PDE，solutions通常存在两个方向上的维度。一个就是solution自变量的个数，这个决定方程是ODE还是PDE。另一个是solution的个数，通常也和方程的个数有关，相当于LSE那种形式，方程个数等价于解向量个数。不过这里我们通常是用矢量来进行表示。而这里，Jacobian可以看作多方程。Hessian可以看作单方程。两者的本质都是求偏导,看下面这个例子就可以分清晰地明白了。
  - The following is a PDE for a vector field $\mathbf{u} : \Omega \to \mathbb{R}^2$:

$$\mathbf{grad}\,\mathrm{div}\,\mathbf{u} = [f_1, f_2, f_3]^\top\,, \quad f_i : \Omega \to \mathbb{R}$$

Differential operators

Differential operators are special linear combinations of partial derivatives. As such they spawn linear operators on spaces of differentiable functions defined on a domain $\Omega \subset \mathbb{R}^d$.

🔴 Important first-order differential operators are

- the gradient , defined for differentiable scalar functions $u : \Omega \to \mathbb{R}$, is the column vector
  -
$$\mathbf{grad}\,u(\boldsymbol{x}) := \begin{bmatrix} \frac{\partial u}{\partial x_1} \\ \vdots \\ \frac{\partial u}{\partial x_d} \end{bmatrix} \in \mathbb{R}^d\,, \quad \boldsymbol{x} \in \Omega$$
  - corresponding to the one-solution (one equaiton) why have mentioned before. However, we will see the gradient of vector in the following.
- the divergence, defined for differentiable vector fields $\mathbf{u} : \Omega \to \mathbb{R}^d$, is the scalar function
  -
$$\mathrm{div}\,\mathbf{u}(\boldsymbol{x}) := \frac{\partial u_1}{\partial x_1}(\boldsymbol{x}) + \cdots + \frac{\partial u_d}{\partial x_d}(\boldsymbol{x}) \in \mathbb{R}\,, \quad \boldsymbol{x} \in \Omega\,,$$
  - corrseponding to the multi-solution (several equations)
- the rotation, defined for $d = 3$ and differentiable vector fields $\mathbf{u} = [u1,\ u2,\ u3]^T : \Omega \to \mathbb{R}^3$, is the column vector
  -
$$\mathbf{curl}\,\mathbf{u}(\boldsymbol{x}) := \begin{bmatrix} \frac{\partial u_3}{\partial x_2} - \frac{\partial u_2}{\partial x_3} \\ \frac{\partial u_1}{\partial x_3} - \frac{\partial u_3}{\partial x_1} \\ \frac{\partial u_2}{\partial x_1} - \frac{\partial u_1}{\partial x_2} \end{bmatrix} \in \mathbb{R}^3\,, \quad \boldsymbol{x} \in \Omega$$

These operators have distinct properties that account for their prominent occurrence in many mathematical models. This can be used to discover the structural properties of the PDE-based model. For instance, we conclude by straightforward computations $\boxed{\mathbf{curl} \circ \mathbf{grad} = 0}$, $\boxed{\mathrm{div} \circ \mathbf{curl} = 0}$.

综合来看，differential opeartors相当于将一个函数（或者是vector space中的一个vector）映射到另一个值，这个值可以是scalar也可以是vector。

🔴 A key second-order differential operator is the Laplacian, defined for a twice differentiable scalar function $u : \Omega \to \mathbb{R}$ as $\Delta u(\boldsymbol{x}) := \mathrm{div}\,\mathbf{grad}\,u(\boldsymbol{x}) = \frac{\partial^2 u}{\partial {x_1}^2}(\boldsymbol{x}) + \cdots + \frac{\partial^2 u}{\partial {x_d}^2}(\boldsymbol{x})\,, \quad \boldsymbol{x} \in \Omega$

A key point in this chapter is that you need to clearly be aware of the resulting dimension after such linear differential operator.

1.3.2.3 Spaces of Continuously Differentiable Functions

Throughout we will need vector spaces of functions with particular properties, mainly concerning their smoothness. For the sake of concise presentation we have to introduce certain compact notations.

Given a set (domain) $\Omega \subset \mathbb{R}^d$, $d \in \mathbb{N}$, we introduce the space of real-valued continuous functions on Ω

$$C^0(\Omega) := \{v : \Omega \to \mathbb{R}\colon v \text{ is continuous}\} .$$

If Ω is a closed set we say that the functions in $C^0(\Omega)$ are "continuous up to the boundary". For an open set Ω the space $C^0(\Omega)$ can even contain unbounded functions!

If $I \subset \mathbb{R}$ is an (open or closed) interval we write

$$C^k(I) := \left\{ v : I \to \mathbb{R}\colon v^{(j)} := \frac{d^j v}{dx^j} \text{ exists and is continuous for all } j \in \{1, \dots, k\} \right\} ,$$

for the space of k-times, $k \in \mathbb{N}_0$, continuously differentiable functions on I. This definition can be extended to d-variate functions on $\Omega \subset \mathbb{R}^d$:

$$C^k(\Omega) := \left\{ v : \Omega \to \mathbb{R}\colon \frac{\partial^{|\boldsymbol{\alpha}|} v}{\partial^{\alpha_1} x_1 \cdots \partial^{\alpha_d} x_d} \text{ exists and is continuous for all } \boldsymbol{\alpha} \in \mathbb{N}_0^d,\ |\boldsymbol{\alpha}| \le k \right\} .$$

We can even set $k \leftarrow \infty$ and obtain the space $C^\infty(\Omega)$ of infinitely many times differentiable functions, also dubbed smooth functions.

Piecewise continuously differentiable functions:

We will also need spaces of functions that are smooth locally:

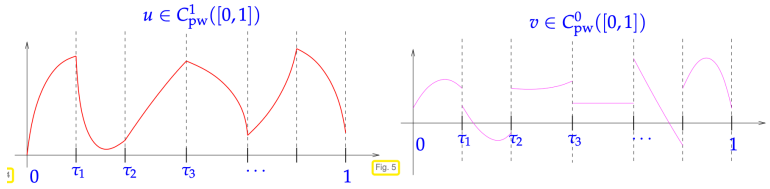
Definition 0.3.2.22. Piecewise continuously differentiable functions

For a closed domain $\Omega \subset \mathbb{R}^d$, $d \in \mathbb{N}$, we define

$$C^k_{\mathrm{pw}}(\Omega) := \{v \in C^{k-1}(\Omega)\colon v|_{\overline{\Omega}_j} \in C^k(\overline{\Omega}_j),\ j = 1, \dots, m\} ,$$

where $\{\Omega_j\}_{j=1}^m$, $m \in \mathbb{N}$, is a partition of Ω in the sense that $\Omega_k \cap \Omega_i = \emptyset$, if $i \neq k$, and $\Omega = \overline{\Omega}_1 \cup \cdots \cup \overline{\Omega}_m$.

See following examples for better understanding:



Corollary 0.3.2.23. Derivative of piecewise smooth functions

For $k \in \mathbb{N}$ differentiation of piecewise continuously differentiable functions on the interval $[a,b]$, $a < b$, yields other piecewise continuously differentiable functions:

$$u \in C^k_{\mathrm{pw}}([a,b]) \quad \Rightarrow \quad \frac{du}{dx} \in C^{k-1}_{\mathrm{pw}}([a,b]) .$$

It also goes without saying that functions in $C^0_{\mathrm{pw}}(\Omega)$ can be integrated over Ω and that

$$f \in C^0_{\mathrm{pw}}(\Omega)\colon \qquad \int_\Omega f(\boldsymbol{x})\,\mathrm{d}\boldsymbol{x} = \sum_{j=1}^m \int_{\Omega_j} f(\boldsymbol{x})\,\mathrm{d}\boldsymbol{x}$$

Functions with zero restriction to the boundary:

For closed $\Omega \subset \mathbb{R}^d$, functions vanishing on the boundary of Ω represent special subspaces of $C^k_{\mathrm{pw}}(\Omega)$ for any k:

$$C^k_{\mathrm{pw},0}(\Omega) := \left\{ v \in C^k_{\mathrm{pw}}(\Omega)\colon v|_{\partial\Omega} = 0 \right\}$$

Here, by $v|_{\partial\Omega}$ we mean the restriction of the function v to the boundary $\partial\Omega$.

1.3.2.4 Norms on Function Spaces

Definition 0.3.2.25. Supremum norm

The supremum norm of an (essentially) bounded function $\mathbf{u} : \Omega \mapsto \mathbb{R}^n$ is defined as

$$\|\mathbf{u}\|_\infty \left(= \|\mathbf{u}\|_{L^\infty(\Omega)} \right) := \sup_{x\in\Omega} \|\mathbf{u}(x)\| \,, \quad \mathbf{u} \in (L^\infty(\Omega))^n \,. \tag{0.3.2.26}$$

- The norm $\|\mathbf{u} - \mathbf{v}\|_{L^\infty(\Omega)}$ measures the maximal distance of point values of the functions $\mathbf{u}$ and $\mathbf{v}$.

Definition 0.3.2.27. Mean square norm/L^2-norm, see Def. 1.3.2.3

For a function $\mathbf{u} \in (C^0_{\mathrm{pw}}(\Omega))^n$ the mean square norm/L^2-norm is given by

$$\|\mathbf{u}\|_0 \left(= \|\mathbf{u}\|_{L^2(\Omega)} \right) := \left(\int_\Omega \|\mathbf{u}(x)\|^2 \,\mathrm{d}x \right)^{1/2} \,, \quad \mathbf{u} \in (L^2(\Omega))^n \,.$$

- Obviously, the L^2-norm is weaker than the supremum norm: $\|v\|_{L^2([a,b])} \le \sqrt{|b-a|}\|v\|_{L^\infty([a,b])} \quad \forall v \in C^0_{\mathrm{pw}}([a,b])$

1.3.2.5 Multi-Dimensional Integration

The integral of a continuous function $u : \Omega \to \mathbb{R}$ on a "reasonably smooth" domain $\Omega \subset \mathbb{R}^d$ can be understood in the sense of Riemann summation.

Riemann summation

For $d = 2$ fill Ω with tiny axes-parallel squares Q_ℓ $\ell = 1, \ldots, n$, see figure, and write $\boldsymbol{x}_\ell$ for their centers and define

$$I_n := \sum_{\ell=1}^{n} f(\boldsymbol{x}_\ell)|Q_\ell| \,, \tag{0.3.2.28}$$

where $|Q_\ell|$ is the volume of the small square Q_ℓ. Then define the integral as the limit

$$\int_\Omega f(\boldsymbol{x}) \,\mathrm{d}\boldsymbol{x} := \lim_{n\to\infty} I_n \,.$$

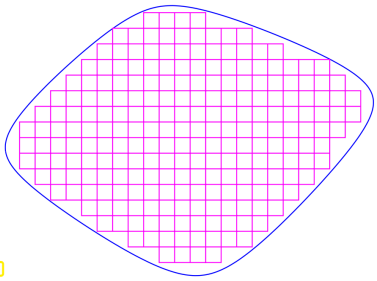
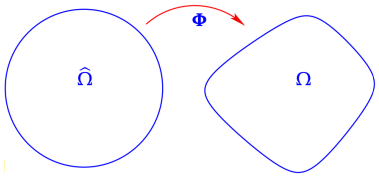


Fig. 6

Local affine-linear approximation



Let the two domains $\widehat{\Omega}$ and Ω be connected by a continuously differentiable mapping $\boldsymbol{\Phi} : D \subset \mathbb{R}^d \to \mathbb{R}^d$:

$$\boldsymbol{\Phi} \in (C^1(\overline{\widehat{\Omega}}))^d \quad , \quad \Omega = \boldsymbol{\Phi}(\widehat{\Omega}) \,.$$

⇓ We got following theorem

Theorem 0.3.2.31. Transformation formula for integrals

Given two domains $\widehat{\Omega}, \Omega \subset \mathbb{R}^d$ and a continuously differentiable mapping $\boldsymbol{\Phi} : \overline{\widehat{\Omega}} \to \overline{\Omega}$, then

$$\int_\Omega f(\boldsymbol{x}) \,\mathrm{d}\boldsymbol{x} = \int_{\widehat{\Omega}} f(\boldsymbol{\Phi}(\widehat{\boldsymbol{x}})) \,|\det \mathrm{D}\boldsymbol{\Phi}(\widehat{\boldsymbol{x}})| \,\mathrm{d}\widehat{\boldsymbol{x}} \tag{0.3.2.32}$$

for any integrable function $f : \Omega \to \mathbb{R}$.

1.3.3 Required Elementary Numerical Methods

This course builds upon the ETH lecture 401-0663-00L Numerical Methods for CSE. In particular, the following topics from computational mathematics provide fundamental tools for this course:

- Techniques for handling sparse matrices and sparse linear systems.
- Numerical quadrature, concepts and methods.
- Numerical method for solving initial value problems for ordinary differential equations (numerical integration), in particular stiff initial value problems.

Lecture notes for the course "Numerical Methods for CSE are available" for download here.

1.4 Mathematical Modeling with Partial Differential Equations

Continuum models Partial differential equations (PDEs) are at the core of most mathematical models arising from a continuum approach, where the configuration or state of a system is described by means of a function on a (multi-dimensional) domain $\Omega \subset \mathbb{R}^d$, $d \geqslant 1$.

1.4.1 PDEs: Basic Notions

A partial differential equation for an unknown function $\mathbf{u} = [u_1, \ldots, u_n]^T : \Omega \subset \mathbb{R}^d \to \mathbb{R}^n, d, n \in \mathbb{N}$, depending on the independent variables $x_1, \ldots, x_d$ ($\mathbf{u} = \mathbf{u}(x_1, \ldots, x_d)$) has the form

$$F(\mathbf{u}, \mathsf{D}\mathbf{u}, \mathsf{D}^2\mathbf{u}, \ldots, \mathsf{D}^m\mathbf{u}) = \mathbf{0}$$

Where F is a general function and $D^j\mathbf{u}$ denotes a tensor of dimension $n \times d \times \cdots \times d$ with nd^j entries defined as

$$\left(\mathsf{D}^j\mathbf{u}(\boldsymbol{x})\right)_{i,k_1,\ldots,k_j} := \frac{\partial^j u_i}{\partial x_{k_1} \ldots \partial x_{k_j}}(\boldsymbol{x}), \quad k_\ell \in \{1, \ldots, d\}, \ell \in \{1, \ldots, j\}$$

✏️ We write $x_1, \ldots, x_d$ for the independent "spatial" variables, $\mathbf{x} = [x_1, \ldots, x_d]^T \in \mathbb{R}^d$.

- $j = 1$ the derivative $D\mathbf{u}$ Bills down to the classical **Jacobian** of $\mathbf{u}$.
- $j = 2$ and $n = 1$ the matrix $D^2\mathbf{u}$ agrees with the **Hessian** of the scalar valued function $u = u_1$

For ODE, the unknown functions can be multiple, but the dimension of the unknown function should be one. For PDE, the dimension of the unknow function should be multiple no matter the number of the unknown functions.

1.5 Useful Links

- Moodle page: https://moodle-app2.let.ethz.ch/course/view.php?id=19025
- Lecture Document: https://www.sam.math.ethz.ch/~grsam/NUMPDEFL/NUMPDE.pdf
- Homework problem collection: https://people.math.ethz.ch/~grsam/NUMPDEFL/HOMEWORK/NPDEFL_Problems.pdf
- Course code repository: https://github.com/erickschulz/NPDECODES
- Course repository: https://people.math.ethz.ch/~grsam/NUMPDEFL/
- Discuna: https://app.discuna.com/fTwNebgISoHVgbl8F5Ca

Week 1

2 Second-Order Scalar Elliptic Boundary Value Problems

模型都是通过真实物理系统引出来的，这里我们先考虑一类物理系统的建模，即equalibrium models。其问题最后的统一描述即系统能量最小化，基于此即刻得到我们系统所处的微分方程方程。(对于transport model和evolution model，是否就不存在所谓的能量最小化了呢？）注意我这里没有提及边界条件，因此给出的是系统general的状态描述。这种所谓的状态描述我们可以通过能量泛函的最小来表达，也可以通过对能量泛函一阶变分=0来进行描述，这两种描述是否等价有待考究。

1. 对于所研究的平衡性问题，我们首先考虑的是其模型的configuration space，然后我们发现configuration space是一个infinite-dimension的函数空间。当边界条件固定时，不同的外力会导致不同的resulting solution。这时函数空间变化为一个连续且受限（满足BC）的函数空间，此函数空间时affine vector space. 2. 然后，我们讨论了平衡性条件，该平衡性条件是建模的关键，即我们知道了解的存在依赖于什么约束。从而我们发现我们之前讨configuration space有点太强了，piece-wise continuous即可（考虑了point loading）。关于如何写出系统的能量函数，这一章节直接给出，具体的如何推导出来我们会在后面讨论。讨论完elastic energy模型后，我们又讨论了electrostaci energy模型，其中u是电势，是scalar，而u的梯度即是电场，讨论的是一个三维问题。⚠️这里还要注意的是dielectric tensor的fundamental property，这对我们后面讨论configuration space非常有用。3. 我们总共讨论了三个问题，两个是elastic model，有一个是elctrostatic model，这三个问题的通性是他们全是quadratic functional，因此我们这里的问题转换成了quadratic functional的minimization problems.

2.1 Preface

Boundary value problem

The term “boundary value problems” occurring in the title of this chapter alludes to the classical (“strong”) form of writing down most PDE-based mathematical models for stationary, that is, time-independent phenomena.

Boundary value problem (BVP)

Given a partial differential operator $\mathcal{L}$, a domain $\Omega \subset \mathbb{R}^d$, a boundary differential operator $\mathcal{B}$, boundary data g, and a source term f, seek a function $u : \Omega \mapsto \mathbb{R}^n$ such that

$$\begin{aligned} \mathcal{L}(u) &= f \quad \text{in} \quad \Omega\,, \\ \mathcal{B}(u) &= g \quad \text{on } \textit{part of (or all)} \text{ boundary } \partial\Omega\,. \end{aligned} \tag{1.1.0.2}$$

Terminology: A boundary value problem is called scalar: $\Leftrightarrow n = 1$ (也就不想Burger's Equ.那种存在x和y方向量)

Supplement:

- Elliptic BVPs (➢ “equilibrium problems”, solutions are minimizers of an “energy”.)
- Parabolic initial boundary value problems (IBVPs) (➢ evolution towards equilibrium, see Section 9.2)
- Hyperbolic IBVPs, among them wave propagation problems and conservation laws (➢ transport/propagation, see Chapter 11)

✏️ There is a rigorous mathematical definition of "elliptic".

In the following chapter, our logics is that:

In this course, we will focus on infinite-dimensional vector space, which we must borrow the tools from analysis. ⚠对于ODE system，仍旧是在函数空间进行讨论，即我们的unknown solution是function而不是一个向量，是一个无限维度的变量。因此不同于代数方程方程，微分方程即在无限维度中讨论问题，而因为是无限维度，简单的线性变化，矩阵相乘这些代数域中用的工具就不够用了，我们要进一步利用分析手段。微积分，实变和复变函数。举个例子，在无限维空间中，线性变化就称之为算子operator。

1. Define the configuration space of such problem ($\equiv$ Infinite-dimensional function space) I discussed such situation in the linear/bilinear form
2. Build the model (energy functional) Deriving from physics.
3. Derive the variational equation satisfying the equilibrium condition (minimization problem) $\Rightarrow$ Optimization problem.

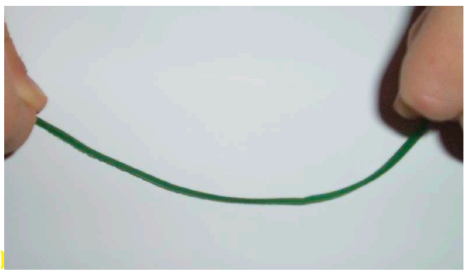
Sobolev spaces: provide the framework for rigorous mathematical investigation of variational equations. However, we will approach Sobolev spaces as "spaces of physically meaningful solutions" or "spaces of solutions with finite energy". From this perspective dealing with Sobolev spaces will be reduced to dealing with their norms.

2.2 Equilibrium Models: Examples

First, introduce the concept of configuration space. Here, we should know that the configuration space should be a infinite-dimensional function space, and it should satisfy some principles such that enable the model make sense. Then, we discuss how to determine the solution in the function space -> equilibrium condtions (minimial potential energy.). Subsequently, introduce the potential energy for elastic model and find that it contains the derivates in the potential energy. Should we let the configure space continuous differentiation? Give an example of point-load setup, and tells that piece-wise continuous differentiation is enough. Therefore. we refine our configure spaces compatible with potential energy. By the way, why we refer this problem as elliptic is closely related to the selection criterion (energy minimizaiton).

The energy minization works as a selection criterion to help us search for the solution from the infinite function space. Can we add such restraints in our Neural Network as regularization or bias to enable it find the solution much quicker and more robust.

2.2.1 Elastic Membrane



Goal: Compute the shape of string (1D) / membrane (2D) under vertical loading

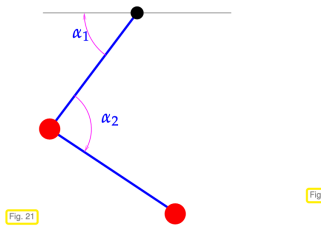
First step: Develop of a mathematical model, based on situable configuration space.

2.2.1.1 Configuration Spaces

Notion 1.2.1.1. Configuration space

The **configuration space** of a mathematical model is a set, each of whose elements completely describe all relevant aspects of a state of the modeled physical system.

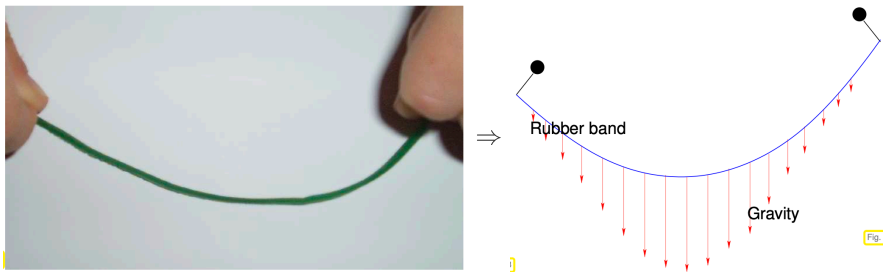
- 就是物理系统所有可能存在的解集。由于这里我们讨论的是independent PDE，解集即state space。



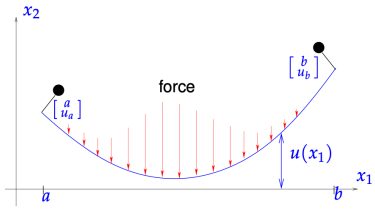
Double pendulum: state described by two angles, configuration space $[0, 2\pi[\times [0, 2\pi[$.

- ODE-based model

Example: 1D



Before we build our configuration space, we need a coordinate system to describe our state such that we can build our configuration space.



Then the shape of the elastic string can be described by the graph of a real-valued function

$$u : [a, b] \to \mathbb{R} ,$$

over the domain $[a, b]$, $a < b$, which is a finite interval in 1D:

string $= \left\{ \begin{bmatrix} x_1 \\ x_2 \end{bmatrix} \in \mathbb{R}^2 : x_2 = u(x_1),\ a \le x_1 \le b \right\}$.

Further evident properties of u:

- Of course, the function u must be continuous, because the string must not have gaps (ripped apart).
- The positions of the pins fix $u(a)$ and $u(b)$ and give boundary conditions (BDC)

$$u(a) = u_a \quad , \quad u(b)) = u_b \quad \text{for given} \quad u_a, u_b \in \mathbb{R}$$

Basic configuration space for elastic string model

The configuration space for the elastic string model under vertical loading is the infinite-dimensional affine function space

$$\widehat{V}_S := \{u : [a, b] \to \mathbb{R} \cdot 1\mathrm{m},\ u \text{ continuous},\ u(a) = u_a,\ u(b) = u_b\} . \qquad (1.2.1.7)$$

- Recall the definition of affine function

–

Definition 0.3.1.1. Affine (sub)space

Given a vector space V, a subspace $V_0 \subset V$, and some $\hat{v} \in V$, we call the set

$$\hat{V} := \hat{v} + V_0 := \{v = \hat{v} + v_0 : \ v_0 \in V_0\} \subset V$$

an affine (sub)space (of V).

– In the concrete case of $\hat{V}_s$ the "hold-all" space V is $C^0([a,b])$, the subspace V_0 is $C^0_{\blacksquare}([a,b])$ (subspace of the continuous function) and $\hat{v}$ is any function $\in V$ satisfying the pining conditions (BDC).

- Which string shapes are not covered by $\hat{V}_s$?

– Answer:

Assumption 1.2.1.4. Warp-free/loop-free shape of string

Each force line intersects the elastic string at most once.

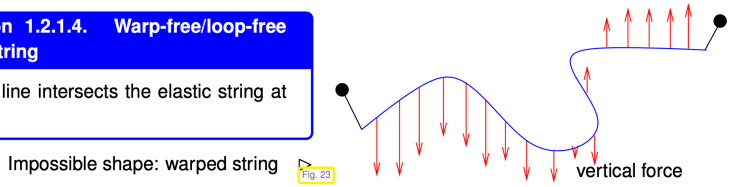


Fig. 23

– Therefor, we can guess that the V_0 is inherently determined by the PDE/constraint.

ChatGPT tells me: if you have the homogeneous BC ($u = 0$ at BC), then, the problem are much easier to solve.

Example: 2D

Shape of membrane
⇕
Graph of $u : \Omega \mapsto \mathbb{R}$

"membrane" on spatial domain $\Omega =]0,1[^2$
(– – – ≙ rigid frame with known geometry)

Physical units: $[u] = 1\text{m}$

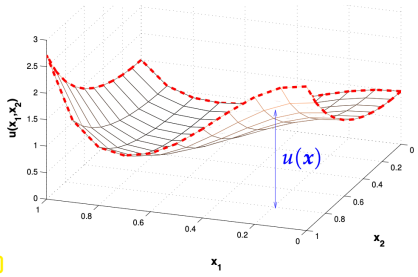


Fig. 25

⇓ The corresponding configuration space

Configuration space for membrane model

Under vertical loading the configuration space for membrane shapes is the infinite-dimensional affine function space

$$\hat{V}_M = \{u : \overline{\Omega} \to \mathbb{R} \cdot 1\text{m} \text{ continuous}, \ u|_{\partial\Omega} = g\} \qquad (1.2.1.12)$$

Question: What is a suitable cfg.-spc. for a balloon under pressure

dispalcement over sphere S: $u : S \to \mathbb{R} \Rightarrow \hat{V} = C^0(S)$

2.2.1.2 Equalibrium conditions

Of course, for prescribed boundary conditions and given loading elastic strings and membranes will attain a unique particular shape corresponding to one element of the configuration space. A key aspect of the mathematical model must be to specify a selection criterion for that element. → Modeling

Vertical loading ⇔ force density $f : \Omega \to \mathbb{R}$ *Physics*: Selection criterion for shape

- Note that the force density f need not be continuous but only integrable. Hence, piecewise continuity is sufficient.

Nature is economical and prefers "optimal" states, which gives us the desired selection criterion:

Equilibrium condition for stationary mechanical systems

The elastic string/membrane attain that shape that achieves a minimal potential energy.

Potential energies of elastic string/membrane

1D: For an elastic string under vertical loading by $f \in C^0_{\mathrm{pw}}(]a,b[)$ whose shape is given by $u \in \widehat{V}_S$ (→ (1.2.1.7)) the total potential energy is

$$J_S(u) := \int_a^b \tfrac{1}{2}\sigma(x)\left|\frac{du}{dx}(x)\right|^2 - f(x)u(x)\,\mathrm{d}x \,. \tag{1.2.1.18}$$

elastic energy — potential energy in force field

Here, the stiffness $\sigma = \sigma(x)$ potentially dependent on the position x, $[\sigma(x)] = 1\mathrm{N}$. This ensures that J has units of energy: $[J(u)] = 1\mathrm{J}$.

2D: For a clamped membrane loaded with a force density f and with displacement $u \in \widehat{V}_M$ (→ (1.2.1.12)) we find for the total potential energy

$$J_M(u) := \int_\Omega \tfrac{1}{2}\sigma(\boldsymbol{x})\|\mathbf{grad}\, u(\boldsymbol{x})\|^2 - f(\boldsymbol{x})u(\boldsymbol{x})\,\mathrm{d}\boldsymbol{x} \,, \tag{1.2.1.19}$$

elastic energy — potential energy in force field

Here, the stiffness coefficient $\sigma = \sigma(\boldsymbol{x})$ has units $[\sigma(\boldsymbol{x})] = \frac{\mathrm{N}}{\mathrm{m}}$, which leads to $[J_M(u)] = 1\mathrm{J}$.

Where $\sigma(\boldsymbol{x})\|\mathbf{grad}\, u(\boldsymbol{x})\|^2 = \sigma(x_1,x_2)\left|\frac{\partial u}{\partial x_1}(x_1,x_2)\right|^2 + \sigma(x_1,x_2)\left|\frac{\partial u}{\partial x_2}(x_1,x_2)\right|^2$, $\boldsymbol{x} = \begin{bmatrix} x_1 \\ x_2 \end{bmatrix}$

- 1D的就是导数的绝对值，2D的就是梯度的绝对值，由于梯度是个向量，故绝对值也变成了用norm来表示。

Now we can quantitatively characterize the equilibrium shape based on the qualitative equilibrium condition:

Equilibrium shape

The equilibrium shape for an elastic string (S) or elastic membrane under vertical loading is

$$u = \operatorname*{argmin}_{v \in \widehat{V}_*} J_*(v) \,, \quad * = S, M \,, \tag{1.2.1.24}$$

with configuration spaces $\widehat{V}_S / \widehat{V}_M$ from (1.2.1.7)/(1.2.1.12) and total potential energy functionals J_S / J_M defined in ((**Q**1.2.1.30.F))/(1.2.1.19).

- 定义functional的概念，即函数的函数。不同于以往的函数，如果是functional，则自变量是infinite-dimensional
- Scaling ≡ express all quantities relative to reference quantities. ⇒ non-dimensional

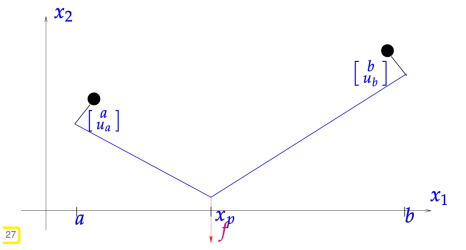
2.2.1.3 Smoothness of Displacements

$$J_S(u) := \int_a^b \tfrac{1}{2}\sigma(x)\left|\frac{du}{dx}(x)\right|^2 - f(x)u(x)\,\mathrm{d}x \,. \tag{1.2.1.18}$$

elastic energy — potential energy in force field

- $\|\frac{du}{dx}(x)\| \to$ derivative $\Rightarrow u \in C^1([a,b])$: continuous differentiable necessary? The answer is no!

1D



point force $f(x) = \delta(x - x_p)$, $a < x_p < b$

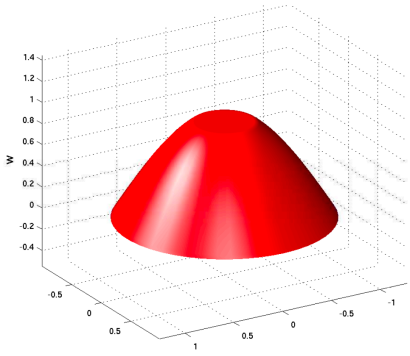
- $J_s(u)$ sitll meaningful. $\frac{du}{dx}$ p.w. continuous $\Rightarrow$ integrable

2D

A function $u \in C^1_{\text{pw}}(\overline{\Omega})$, where Ω is the unit disk:

$$\Omega := \{x \in \mathbb{R}^2 : \|x\| < 1\} .$$

This function is an eligible displacement for a membrane mounted on a flat circular frame, because $J_M(u)$ can be computed by piecewise integration of the squared norm of the discontinuous gradient **grad** u.



! These considerations yield the following upgraded definition of the configuration spaces:

Energy-compatible configuration spaces

For vertical loading the following configuration spaces for displacements u take into account prescribed boundary conditions and allow the computation of potential energies

1D (elastic string): $$\widehat{V}_S := \left\{ u \in C^1_{\text{pw}}([a,b]),\ u(a) = u_a,\ u(b) = u_b \right\} , \tag{1.2.1.28}$$

2D (membrane): $$\widehat{V}_M := \left\{ u \in C^1_{\text{pw}}(\overline{\Omega}),\ u|_{\partial\Omega} = g \right\} . \tag{1.2.1.29}$$

- Extend the configuraiton space.

Due to integration trait of energy function, the corresponding compatible configuration space is extended. This is totally different from the strong form of the PDE.

2.2.2 Electrostatic Fields

In this chapter. we introduce another second-order scalar elliptic BVP, electrostatics. Here, important point to be stated is that the dielectric tensor is a 3 by 3 matrix. In order to satisfy the positive of the electromagnetic field energy, the dielectric tensor should satisfy some restraints. Furthermore, after we still use potential to ensure our configuration space. (piece-wise differential continuity).

Next, we consider a question arising from electrostatics

A typical arrangement for an electrostatic setup is sketched beside ▷

- chunk of metal in conducting box
- prescribed voltage drop metal—box

We seek the electric field $\mathbf{E} : \Omega \mapsto \mathbb{R}^d$ in $\Omega \subset \mathbb{R}^d$, $d = 2, 3$.
($\Omega \hat{=}$ blue region ▷)

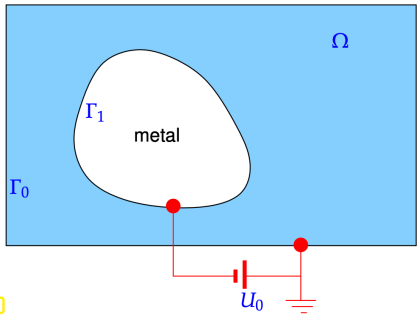
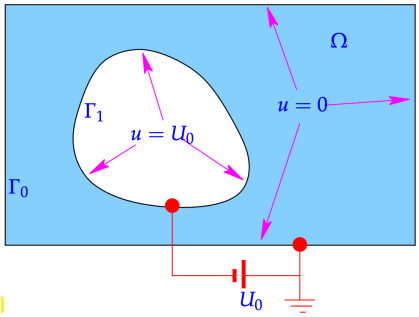


Fig. 32

From Maxwell'sequations in a stationary setting, where all fields are constant in time: $\mathbf{E} = -\mathbf{grad} u$ for a scalar function $u : \Omega \to \mathbb{R}$ called the electric (scalar) potential.

Boundary conditions for electrostatic potential

The resulting configuration space is as follows:



Thus in the situation of Fig. 32 the electric potential must statisfy the following **boundary conditions**

$$\begin{aligned} u &= 0 \quad \text{on } \Gamma_0 \,, \\ u &= U_0 \quad \text{on } \Gamma_1 \,. \end{aligned} \tag{1.2.2.4}$$

which implies the *preliminary* configuration space

$$\widehat{V}_E = \left\{ u \in C^0(\overline{\Omega}) \,,\, u \text{ satisfies (1.2.2.4)} \right\} .$$

This choice may be problematic: What is the meaning of $\mathbf{grad}\, u$. We are going to address this shortly.

Electrostatic field energy

Electromagnetic field energy: (electrostatic setting with scalar potential u)

$$J_E(u) = \tfrac{1}{2} \int_\Omega (\boldsymbol{\epsilon}(\boldsymbol{x})\, \mathbf{grad}\, u(\boldsymbol{x})) \cdot \mathbf{grad}\, u(\boldsymbol{x})\, \mathrm{d}\boldsymbol{x} \,, \tag{1.2.2.6}$$

where $\boldsymbol{\epsilon} : \Omega \mapsto \mathbb{R}^{3,3}$ is the dielectric tensor, $\boldsymbol{\epsilon}(\boldsymbol{x})$ is symmetric, with units $[\boldsymbol{\epsilon}] = \frac{\mathrm{A\,s}}{\mathrm{V\,m}}$.

Note that in terms of partial derivatives and matrix components $\boldsymbol{\epsilon}(\boldsymbol{x}) = \left[\epsilon_{ij}\right]_{i,j=1}^{3}$ we have

$$(\boldsymbol{\epsilon}(\boldsymbol{x})\, \mathbf{grad}\, u(\boldsymbol{x})) \cdot \mathbf{grad}\, u(\boldsymbol{x}) = \sum_{i=1}^{3} \sum_{j=1}^{3} \epsilon_{ij}(\boldsymbol{x}) \frac{\partial u}{\partial x_i}(\boldsymbol{x}) \frac{\partial u}{\partial x_j}(\boldsymbol{x}) \,.$$

- This definition is from physics.
- Essential: $J_E(u) \geq 0 \Rightarrow \epsilon$ is uniformly positive definite
- A fundamental property of the dielectric tensor (for "normal" materials) is uniform positivity:
  - $\exists 0 < \epsilon^- \leq \epsilon^+ < \infty$: $\epsilon^- \|\mathbf{z}\|^2 \leq (\boldsymbol{\epsilon}(\boldsymbol{x})\mathbf{z}) \cdot \mathbf{z} \leq \epsilon^+ \|\mathbf{z}\|^2 \quad \forall \mathbf{z} \in \mathbb{R}^3, \forall \boldsymbol{x} \in \Omega$. [apply with $\mathbf{z} \equiv \mathbf{grad} u$]
    - Note: $\mathbf{z} \cdot \mathbf{z} = \|\mathbf{z}\|^2$

The corresponding configuration space:

Configuration space for electrostatic field problems

As configuration space for scalar potentials we use the affine space

$$\widehat{V}_E := \left\{ u \in C^1_{\mathrm{pw}}(\overline{\Omega}) \,,\, u \text{ satisfies (1.2.2.4)} \right\} . \tag{1.2.2.13}$$

- 同理，我们只需要C^1piece-wise连续即可，不需要C^1可微或者可导。

- Note: $\hat{V}_E \subset C^0(\overline{\Omega})$

Equilibrium condition for electrostatics

Electrostatic field problem solved by potential $u_* \in \widehat{V}_E$, for which the field energy $J_E(u)$ becomes minimal:

$$u_* = \operatorname*{argmin}_{u \in \widehat{V}_E} J_E(u) \,. \tag{1.2.2.16}$$

2.2.3 Quadratic Minimization Problems

In this chapter, we first define the concept of quadratic functional. And it should be noted that the quadratic functional should defined on a vector space V_0, which contained the zero vector. Subsequently, we further defined the concept of quadratic minimization problems. However, in the problems we studied before, the minimization problem is defined on the affine space. Therefore, we can transfer this problem into quadratic minimization problem: solution = u_0 (a function satisfying BDC) + $\underbrace{argmin}_{u \in V_0} J(u)$.

Section 1.2.1.2 [1D, string]

$$u_* = \operatorname*{argmin}_{\substack{u \in C^1_{\mathrm{pw}}([a,b]) \\ u(a)=u_a\, u(b)=u_b}} \underbrace{\int_a^b \tfrac{1}{2}\sigma(x)\left|\frac{du}{dx}(x)\right|^2 - f(x)u(x)\,\mathrm{d}x}_{=J_S(u),\ \text{see ((Q1.2.1.30.F))}} \tag{1.2.3.1a}$$

Section 1.2.1.2 [2D membrane]

$$u_* = \operatorname*{argmin}_{\substack{u \in C^1_{\mathrm{pw}}(\overline{\Omega}) \\ u=g \text{ on } \partial\Omega}} \underbrace{\int_\Omega \tfrac{1}{2}\sigma(\boldsymbol{x})\|\mathbf{grad}\, u(\boldsymbol{x})\|^2 - f(\boldsymbol{x})u(\boldsymbol{x})\,\mathrm{d}\boldsymbol{x}}_{=:J_M(u),\ \text{see (1.2.1.19)}} \,, \tag{1.2.3.1b}$$

Section 1.2.2 [2D, 3D electrostatics]

$$u_* = \operatorname*{argmin}_{\substack{u \in C^1_{\mathrm{pw}}(\overline{\Omega}) \\ u=U \text{ on } \partial\Omega}} \underbrace{\int_\Omega \tfrac{1}{2}(\boldsymbol{\epsilon}(\boldsymbol{x})\,\mathbf{grad}\, u(\boldsymbol{x}))\cdot\mathbf{grad}\, u(\boldsymbol{x})\,\mathrm{d}\boldsymbol{x}}_{=:J_E(u),\ \text{see (1.2.2.6)}} \,. \tag{1.2.3.1c}$$

The next topic we discussed after is Existence & Uniqueness of minimizers. First. we focued on bilinear form, and discussed that the necessary condition of Existence and uniqueness of bilinear form should be positive definite. Then, we talked about the sufficient condition for uniqueness of QMP is that the bilinear form should be positive definite. (NOTE: uniqueness is different from existence). And for V_0 belong to finite-dimensional vector space, if $a(\cdot,\cdot)$ s.p.d $\rightarrow$ E&U a是symmetric是用来保证quadratic。而a是p.d.则是用来保证解存在的necessary condition，并且我们也可以注意到，uniqueness的条件并existence更松。

Next, we recall from LA: if bilinear form is s.p.d. $\Rightarrow$ define inner product. And every inner product induce a norm. Therefore, we use it to define the energy norm. $\|u\|_a = (a(u,u))^{1/2}, u \in V_0$. And we found that the problem we studied are all s.p.d.

In the end. we discussed the necessary condition of linear form for the existence of a minimizer: $l(\cdot)$ is continuous and bounded w.r.t $\|\cdot\|_a$. This lemma told us we can only expect the existence of the minimizer if and only if l is bounded by energy norm in the vector space. Then, the lecturer use a example of needle load of membrane to illustrate that when the lemma is not satisfied. The functional is going to infinite.

这个定理更严格来说是说这个有着s.p.d. bilinear form 下有界（bounded from below）当且仅当linear form is bounded by energy norm. And we should know that J is Bounded from below is the existence for the existence of the minimizer (bounded from below下有界)

Recall the minimization problems arising from the equilibrium conditions:

Section 1.2.1.2 [1D, string]

$$u_* = \operatorname*{argmin}_{\substack{u\in C^1_{\mathrm{PW}}([a,b]) \\ u(a)=u_a\, u(b)=u_b}} \underbrace{\int_a^b \tfrac{1}{2}\sigma(x)\left|\frac{du}{dx}(x)\right|^2 - f(x)u(x)\,\mathrm{d}x}_{=J_S(u),\ \text{see ((Q1.2.1.30.F))}} \tag{1.2.3.1a}$$

Section 1.2.1.2 [2D membrane]

$$u_* = \operatorname*{argmin}_{\substack{u\in C^1_{\mathrm{PW}}(\bar{\Omega}) \\ u=g \text{ on } \partial\Omega}} \underbrace{\int_\Omega \tfrac{1}{2}\sigma(\boldsymbol{x})\|\mathbf{grad}\, u(\boldsymbol{x})\|^2 - f(\boldsymbol{x})u(\boldsymbol{x})\,\mathrm{d}\boldsymbol{x}}_{=:J_M(u),\ \text{see (1.2.1.19)}}\ , \tag{1.2.3.1b}$$

Section 1.2.2 [2D, 3D electrostatics]

$$u_* = \operatorname*{argmin}_{\substack{u\in C^1_{\mathrm{PW}}(\bar{\Omega}) \\ u=U \text{ on } \partial\Omega}} \underbrace{\int_\Omega \tfrac{1}{2}(\boldsymbol{\epsilon}(\boldsymbol{x})\,\mathbf{grad}\, u(\boldsymbol{x}))\cdot\mathbf{grad}\, u(\boldsymbol{x})\,\mathrm{d}\boldsymbol{x}}_{=:J_E(u),\ \text{see (1.2.2.6)}}\ . \tag{1.2.3.1c}$$

Obviously, these minimization problems are rather similar. $J_* \equiv$ quadratic functional

In this section we will discuss their structure (Give a general form for these problems) and preliminary results about existence and uniqueness of solutions.

2.2.3.1 Definition

Definition Generalize these problems

In the first chapter, we have discussed about the linear form/bilinear form. However, in former examples, the dimension of the vector space is finite. Here, we discuss about the infinite-dimensinal vector space. a and l are functional defined in this vector space. 注意，这里functional的定义，有点像norm的定义（但又不是norm，我们可以基于bilinear form来定义vector space上的norm），而不是线性空间上operation的定义，因为operation的定义是需要封闭在该空间中，而这是建立了一个从funtion到real space的映射。

Bilinear forms and linear forms are key building blocks for the energy functionals J_S, J_M, and J_E from (1.2.3.1). They all match the following type of functional:

Definition 1.2.3.2. Quadratic functional

A quadratic functional on a *real vector space* V_0 is a mapping $J : V_0 \mapsto \mathbb{R}$ of the form

$$J(u) := \tfrac{1}{2}\mathsf{a}(u,u) - \ell(u) + c\,,\quad u \in V_0\,, \tag{1.2.3.3}$$

where $\mathsf{a} : V_0 \times V_0 \mapsto \mathbb{R}$ is a symmetric bilinear form (→ Def. 0.3.1.4), $\ell : V_0 \mapsto \mathbb{R}$ a linear form, and $c \in \mathbb{R}$.

- $a(u,u)$表示的是自己和自己作用的映射，有点像二次函数里面的x^2。此外，quadratic functional形式上和linear form一致，都是function u 到$\mathbb{R}$的映射。不同的是一个是linear，另一个是quadratic。
- ⚠Note that the bilinear form should be symmetric.

example Quadratic functionals on $\mathbb{R}^N$

If $V_0 = \mathbb{R}^N$ (finite-dimensional case), then a quadratic functional has the general representation:

$J(\vec{\boldsymbol{\eta}}) = \frac{1}{2}\vec{\boldsymbol{\eta}}^\top \mathbf{A}\vec{\boldsymbol{\eta}} - \vec{\boldsymbol{\beta}}^\top\vec{\boldsymbol{\eta}} + c$, $\mathbf{A} = \mathbf{A}^\top \in \mathbb{R}^{N,N}$, $\vec{\boldsymbol{\beta}} \in \mathbb{R}^N$, $c \in \mathbb{R}$

On $\mathbb{R}^N$: binlinear form $\Leftrightarrow$ matrix $\in \mathbb{R}^{N,N}$ linear form $\Leftrightarrow$ vector $\in \mathbb{R}^N$

因此，这里的quadratic functional同样可以用form来进行表示，唯一不同的不过是form更general，而functional更多指的是infinite-dimension function sapce。但是我需要指出的是，这些functional转换成vector space的思路非常理所当然地可以应用在现代ML框架中，因为计算机中只能处理finite-dimensional case，而你要做的就是在finite-dimensional space上处理问题，但是其却和infinite-dimensional等价 -> discrete-continuous equivalence.

Question: what is $a(\cdot,\cdot)$ and $l(\cdot,\cdot)$ for J_M

Note: $\|\mathbf{z}\|^2 = \mathbf{z} \cdot \mathbf{z}, \mathbf{z} \in \mathbb{R}^2$

$$J_M(v) = \int_\Omega \tfrac{1}{2}\sigma(x)\|\mathbf{grad}\, v(x)\|^2 - f(x)v(x)\,\mathrm{d}x \text{ from (1.2.3.1b):}$$

$$\mathsf{a}(w,v) = \int_\Omega \sigma(x)\mathbf{grad}\, w(x) \cdot \mathbf{grad}\, v(x)\mathrm{d}x\,, \quad \ell(v) = \int_\Omega f(x)v(x)\,\mathrm{d}x\,.$$

- Energies are quadratic functional. $u, v \in V_0 \equiv C^1_{\mathrm{pw},0}(\overline{\Omega})$
- 事实上，you have to prove that it is linear,但是我们发现这个形式非常像在有限维函数空间中linear form和bilinear form的矩阵和向量形式。所以我们直接默认了。

Obviously, (1.2.3.1) is all about minimizing quadratic functionals. We introduce a name for this kind of minimization problems.

Definition 1.2.3.11. Quadratic minimization problem

A minimization problem

$$w_* = \operatorname*{argmin}_{w \in V_0} J(w)$$

is called a quadratic minimization problem, if J is a *quadratic functional* on a real vector space V_0.

Offset functions:

We can find that neither (1.2.3.1a), nor (1.2.3.1b) nor (1.2.3.1c) are genuine quadratic minimization problems in the sense of Def. 1.2.3.11, because they are posed over affine spaces instead of vector spaces. Just notice that the configuration spaces may not contain the zero function. We can use trick called offset fucntion to solve this problem.

Consider: $\underbrace{\operatorname{argmin} J(\hat{v})}_{\hat{v} \in \widehat{V}}$ w.r.t $\hat{v} = u_0 + u$

according to the bilinearity of a and the linearity of l:

$$\begin{aligned} J(u+u_0) &= \tfrac{1}{2}\mathsf{a}(u+u_0, u+u_0) - \ell(u+u_0) + c \\ &= \tfrac{1}{2}\mathsf{a}(u,u) + \tfrac{1}{2}\mathsf{a}(u,u_0) + \tfrac{1}{2}\mathsf{a}(u_0,u) + \tfrac{1}{2}\mathsf{a}(u_0,u_0) - \ell(u) - \ell(u_0) + c \\ &= \tfrac{1}{2}\mathsf{a}(u,u) + \underbrace{\mathsf{a}(u,u_0) - \ell(u)}_{=:\widetilde{\ell}(u)} + \underbrace{\tfrac{1}{2}\mathsf{a}(u_0,u_0) - \ell(u_0) + c}_{=:\widetilde{c}}\,, \end{aligned} \qquad (1.2.3.14)$$

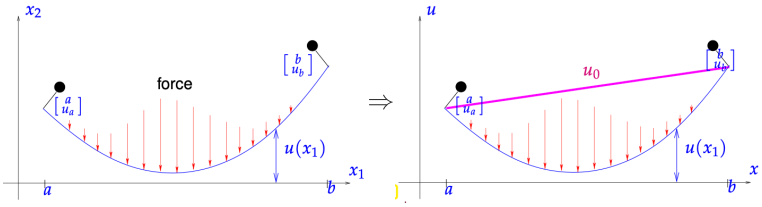
Note: for fixed u_0: $u \to a(u, u_0)$ is a linear form! Hence, the minimizer of J over the affine subspace $u_0 + V_0, V_0 \subset V$ a subspace of V, can be computed as follows:

$$\operatorname*{argmin}_{u \in u_0 + V_0} J(u) = u_0 + \operatorname*{argmin}_{v \in V_0} J(v + u_0) = u_0 + \operatorname*{argmin}_{v \in V_0} \widetilde{J}(v)\,,$$
$$\text{with} \quad \widetilde{J}(v) := \tfrac{1}{2}\mathsf{a}(v,v) + \widetilde{\ell}(v) + \widetilde{c}\,,$$

- u_0 is a function satisfying BDC
- $\widetilde{J}(u)$ satisfies the definition 1.2.3.11.

因此，我们的minimization problems全部可以看作是Quadratic minimization problems。我们的研究重心也全部转移为对quadratic minimization problems这一典型事例的研究上🧐。

⇓ Example: what is the u_0 for the elastic string model?



For elastic string model, with u_0 from $u_0(x) = u_a \frac{b-x}{b-a} + u_b \frac{x-a}{b-a}$

$$V_0 = C^1_{\mathrm{pw},0}([a,b])\,, \qquad \tilde{\ell}(v) := \int_a^b f(x)v(x)\,\mathrm{d}x - \mathsf{a}(u_0, v)$$

2.2.3.2 Existence & Uniqueness of Minimizers

Our models would be highly dubious, if they gave rise to quadratic minimization problems that failed to have unique solutions. We first tackle this issue in an abstract context. Next, we will process this question.

必要条件是指必须具备的重要条件，而充分条件是指一定能够保证结果出现的条件；必要条件可以由结果推出条件，而充分条件是由条件一定能够推出结果，但由结果推出的不仅仅是这个条件，还有别的存在”

Following, we give the necessity for the **existence of a minimizer**. 我们先讨论存在性问题

Corollary 1.2.3.26. Necessary condition for existence of minimizer

The quadratic functional $J(v) := \frac{1}{2}\mathsf{a}(v,v) - \ell(v)$ (→ Def. 1.2.3.2) on a vector space V_0 is bounded from below, only if *the bilinear form $\mathsf{a} : V_0 \times V_0 \to \mathbb{R}$ is positive semi-definite.*

- 这个结论证明其必要性很容易，假设不满足，则发现J会随着增加趋向于负无穷大，因此has no minimum.

Next, we discussed about the **uniqueness of the minimizer** for the quadratic functional problem. There is a **necessary and sufficient condition** in terms of a simple property of a. 充分必要条件for唯一性

Theorem 1.2.3.31. Uniqueness of solutions of quadratic minimization problems

If the bilinear form $\mathsf{a} : V_0 \times V_0 \mapsto \mathbb{R}$ is positive definite (→ Def. 1.2.3.27), then any solution of

$$u_* = \operatorname*{argmin}_{u \in V_0} J(u) \quad , \quad J(u) := \tfrac{1}{2}\mathsf{a}(u,u) - \ell(u) + c\,,$$

is unique for any linear form $\ell : V_0 \mapsto \mathbb{R}$.

- 其中，positive definite定义如下：

Definition 1.2.3.27. Positive definite bilinear form → Def. 0.3.1.16

A (symmetric) bilinear form $\mathsf{a} : V_0 \times V_0 \mapsto \mathbb{R}$ on a real vector space V_0 is positive definite, if

$$u \in V_0 \setminus \{0\} \iff \mathsf{a}(u,u) > 0\,.$$

以上结论放到二次函数中来看就是二次项系数系数是正的（这里是binlinear form是positive definite）从而来保证存在最小值。相当于通过binlinear form来定义更加的general.

$$\begin{array}{ccccc} J(v) & = & \frac{1}{2}a(v,v) & - & \ell(v) \\ \updownarrow & & \updownarrow & & \updownarrow \\ f(x) & = & \frac{1}{2}ax^2 & - & bx \end{array}$$

with $a > 0$!

Graphs of quadratic functionals with s.p.d. bilinear forms are “higher-dimensional” upward-open parabolas.

graph of quadratic functional $\mathbb{R}^2 \mapsto \mathbb{R}$ ▷

Geometric intuition suggests unique global and local minimum.

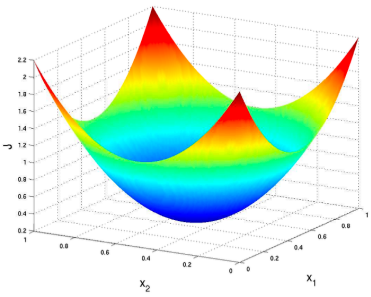


Fig. 37

Convexity of quadratic functionals

给出了real-valued function凸的定义。

Definition 1.2.3.33. Convexity of a real-valued function → [Str09, Def. 5.5.2]

A function $F : V_0 \to \mathbb{R}$ on a vector space V_0 is called convex, if

$$F(\lambda \mathbf{x} + (1-\lambda)\mathbf{y}) \le \lambda F(\mathbf{x}) + (1-\lambda)F(\mathbf{y}) \quad \forall \mathbf{x}, \mathbf{y} \in V_0 . \tag{1.2.3.34}$$

A function is concave, if its negative is convex.

It is well known that a twice continuously differentiable function $F : \mathbb{R} \to \mathbb{R}$ is convex if and only if its second derivative F'' is non-negative everywhere. Thus, the convexity of a quadratic functional based on a positive definite quadratic bilinear form is easily seen by considering the second derivative of the function

$$\varphi(t) := J(u + tv) \quad \Rightarrow \quad \ddot{\varphi}(t) = \mathsf{a}(v, v) > 0 \quad \text{, if } v \neq 0 .$$

2.2.3.3 Energy Norm

Recall from LA: $A = A^T$ s.p.d $\Rightarrow (\vec{\mu}, \vec{\eta}) \to \vec{\mu} A \vec{\eta} \equiv$ inner product话说正常的inner product没有A。这里是想告诉我们，bilinear form是一个二元运算符，这个二元运算符作用在vector space中的一个vector上面时，可以定义一个norm。这里，我们在无穷维的vector space中，特指对于一个quadratic functional，其通过bilinear functional定义的norm称之为energy norm。有这样的norm之后我们就不要定义其他的norm了。这样看来bilinear form的本质其实就是norm。但是这里要注意，定义能量norm的前提是a必须是spd，也就是满足了Uniqueness，但不一定满足existence。

Definition 1.2.3.35. Energy norm, *cf.* [Hip19, ??]

Let a be the symmetric positive definite bilinear form $\mathsf{a} : V_0 \times V_0 \mapsto \mathbb{R}$ (→ Def. 0.3.1.16) underlying a quadratic functional J. Then the related energy norm is

$$\|u\|_{\mathsf{a}} := (\mathsf{a}(u,u))^{1/2} , \quad u \in V_0 .$$

- Symmetric positive definite bilinear form
  - symmetric: definition of quadrtic functional.
  - positive definite: uniquness and existence
- Inner product的定义可以将vector space扩展到内积空间 - 具备长度和角度
- Norm的定义可以将vector space进一步扩展到赋范空间

Example: Electrostatic energy functional J_E

$$\mathsf{a}(u,v) := \int_\Omega (\boldsymbol{\epsilon}(\boldsymbol{x}) \operatorname{\mathbf{grad}} u(\boldsymbol{x})) \cdot \operatorname{\mathbf{grad}} v(\boldsymbol{x}) \, \mathrm{d}\boldsymbol{x} \qquad V_0 := C^1_{\mathrm{pw},0}(\overline{\Omega}).$$

Know: $\exists 0 < \epsilon^- \le \epsilon^+ < \infty$: $\epsilon^- \|\mathbf{z}\|^2 \le (\boldsymbol{\epsilon}(\boldsymbol{x})\mathbf{z}) \cdot \mathbf{z} \le \epsilon^+ \|\mathbf{z}\|^2 \quad \forall \mathbf{z} \in \mathbb{R}^3, \forall \boldsymbol{x} \in \Omega$

Question: Is $a(\cdot,\cdot)$ s.p.d.?

we infer the bounds

$$0 \le \epsilon^- \int_\Omega \|\operatorname{\mathbf{grad}} u(\boldsymbol{x})\|^2 \, \mathrm{d}\boldsymbol{x} \le \mathsf{a}(u,u) \le \epsilon^+ \int_\Omega \|\operatorname{\mathbf{grad}} u(\boldsymbol{x})\|^2 \, \mathrm{d}\boldsymbol{x} \quad \forall u .$$

Hence, it is sufficient to examine the simpler bilinear form

$$\mathsf{d}(u,v) := \int_\Omega \operatorname{\mathbf{grad}} u(\boldsymbol{x}) \cdot \operatorname{\mathbf{grad}} v(\boldsymbol{x}) \, \mathrm{d}\boldsymbol{x} , \quad u, v \in C^1_{\mathrm{pw},0}(\overline{\Omega}) .$$

❷: Obviously $\mathsf{d}(u,u) = 0 \Rightarrow \operatorname{\mathbf{grad}} u = 0 \overset{(1.2.1.22)}{\Longrightarrow} u \equiv \text{const in } \Omega$

Observe: $u = 0$ on $\partial\Omega \Rightarrow u = 0$

- such that $a(\cdot,\cdot)$ is s.p.d.

2.2.3.4 A continuity Condition for $l(\cdot)$

In Cor. 1.2.3.26 and Thm. 1.2.3.31 we found necessary conditions on the bilinear form a for the existence of a minimizer of an abstract quadratic minimization problem. Now we answer the questions whether the linear functional l also has to satisfy some conditions.

The following insight demonstrates that the linear form l has to match the bilinear form a($\cdot$, $\cdot$) to make possible to existence of a minimizer of J:

Lemma 1.2.3.39. Boundedness condition on linear form

The quadratic functional J (→ Def. 1.2.3.2)

$$J(u) := \tfrac{1}{2}a(u,u) - \ell(u) + c\,, \quad u \in V_0\,, \tag{1.2.3.3}$$

based on a symmetric positive definite bilinear form a (→ Def. 1.2.3.27) is bounded from below on V_0, if and only if

$$\exists C > 0: \quad |\ell(u)| \le C\|u\|_a \quad \forall u \in V_0\,, \tag{1.2.3.40}$$

where $\|\cdot\|_a$ is the energy norm induced by a, see Def. 1.2.3.35.

- The formula in the red bracket has the name: $l(\cdot)$ is continuous bounded w.r.t $\|\cdot\|_a$
- This lemma means that only when the boundedness condition is successfully applied, can the quadratic function J is bounded from below on V_0
- This condition often satisfies in finite dimension but violate under infinite dimension.
- bounded from below is necessary for the existence of minimizer.

The lecture note use an example "Point load" to illustrate the lemma.

2.3 Sobolev Spaces

对于任何一个functional，其energy space是通过其energy norm来定义的，目的就是寻找使得energy norm make sense的最大function space。在这里讨论了一个很有意思的点，是关于L^2 norm的boundary condtion问题。在我们研究一个quadratic functional的时候，如果我们在L^2里面进行讨论，则我们构造的新的energy space（larger space）忽略了boundary condition。但是在H_0^1 space中，我们新构建的energy space则又可以考虑boundary condition。

这一章节主要讨论的是很多时候，就是我们满足s.p.d.在function space中解也不存在。因此，这种情况下，我们需要扩大我们的function space。而我们寻找largest space of functions主要是基于a still makes sense! 也就是基于要解决的quadratic minimization problem来做设计。是的bilinear form仍旧make sense的最大空间。在这个过程中，我们也发现了一些比较奇怪的情况，就是对于L^2 space, 我们在新定义的energy space中直接舍弃掉了boundary condition。通过举了一个though experiment，我们发现，在这个energy space，似乎有边界的解和无边界的解在L2度量下是一样的，从而我们可以理解为这两个解是相同的。所以我们可以理解为BDC meaningful in L2 energy space中。但对于H1 space，BDC则可以被imposed。并且，当你dropping BDC of H_0^1的时候，你的norm不能和H_0^1的时候相同。因为H_0^1其实我们变向地约束了解是const的情况。而在H^1的时候，你的解可以为常数，但是我们知道norm必须大于0，所以为了避免这种情况，我们又加上了L_2norm。

在这些问题中，我们知道我们之前研究的minimization problem一直是把affiline space转换到vector space下的优化问题。在这种情况下，转换后的vector space我们要求边界为0。但是现在我们似乎不需要转换，而用H^1space给出maximal function space（这三种问题的讨论都合理）这时可以施加不为0的边界。

接着我们讨论了一个问题，也就是我们讨论了2D membrane的解的existence。也就是f受到什么样的限制，才能使得我们的解可能存在。这里推导的过程我们用到了：1. linear form bounded by energy norm.(这是最后论证的目标) 2. Cauchy-Scharz inequality 从而把linear form展开，目的是为了尝试推到常数C 3. First Poincare-Friedrichs inequality（把L2 norm和energy norm联系起来）

后面还有一些问题，之后再研究

在L^2 space中，虽然我们没考虑边界条件得到一个解，但是

Corollary 1.2.3.26. Necessary condition for existence of minimizer

The quadratic functional $J(v) := \frac{1}{2}a(v,v) - \ell(v)$ (→ Def. 1.2.3.2) on a vector space V_0 is bounded from below, only if *the bilinear form $a : V_0 \times V_0 \to \mathbb{R}$ is positive semi-definite.*

Theorem 1.2.3.31. Uniqueness of solutions of quadratic minimization problems

If the bilinear form $a : V_0 \times V_0 \mapsto \mathbb{R}$ is positive definite (→ Def. 1.2.3.27), then any solution of

$$u_* = \operatorname{argmin}_{u \in V_0} J(u) \quad , \quad J(u) := \frac{1}{2}a(u,u) - \ell(u) + c \, ,$$

is unique for any linear form $\ell : V_0 \mapsto \mathbb{R}$.

Lemma 1.2.3.39. Boundedness condition on linear form

The quadratic functional J (→ Def. 1.2.3.2)

$$J(u) := \frac{1}{2}a(u,u) - \ell(u) + c \, , \quad u \in V_0 \, , \tag{1.2.3.3}$$

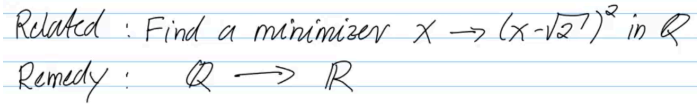
based on a symmetric positive definite bilinear form a (→ Def. 1.2.3.27) is bounded from below on V_0, if and only if

$$\exists C > 0: \quad |\ell(u)| \le C\|u\|_a \quad \forall u \in V_0 \, , \tag{1.2.3.40}$$

where $\|\cdot\|_a$ is the energy norm induced by a, see Def. 1.2.3.35.

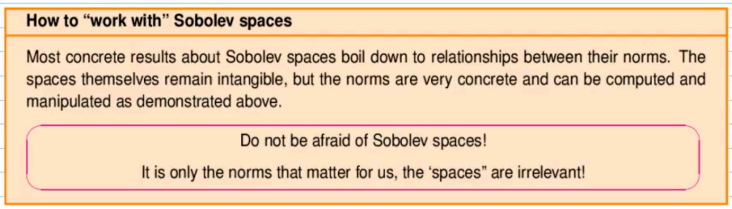
Very strange, for finite dimension space, QMP for J exists. However, for infinite dimensional space, we cannot find a minimizer for J

举了有理数的例子。实数包含了有理数和无理数，而有理数包含了整数和分数。



If you cannot find a solution in the current function space, then we may extend to a larger spaces.

一直在讨论的问题就是minimizer在何种情况下是存在的。在这个过程中我们需要extend our function space，问题变化到sobolev space上去讨论。同时我们也需要对load function做一定的限制。在讨论sobolev spaces的时候我们也要明白，space如何其实没有太大意义，我们重点需要关心的是sobolev spaces导出的norm。但这又引起了我的困惑，energy norm不是用$a(\cdot,\cdot)$决定的吗？和sobolev space的关系是什么？



- L2 space指代的是什么?
- Sobolev space指代的是什么?

Parlance: In mathematical terms (1.2.3.40) means that ℓ is continuous w.r.t. the energy norm $\|\cdot\|_{\mathsf{a}}$.

Remark 1.2.3.43 (Existence of minimizers in finite dimensions) Beside uniqueness, existence of minimizers of quadratic minimization problems (→ Def. 1.2.3.11) with positive definite bilinear form a is a key issue. In a finite dimensional setting this is not a moot point, see Fig. 37 for a “visual proof”. We do not even need any assumption on ℓ!

Theorem 1.2.3.44. Existence of unique minimizer in finite dimensions

Let $J(u) := \frac{1}{2}\mathsf{a}(u,u) - \ell(u) + c$ with symmetric positive definite (→ Def. 1.2.3.27) bilinear form $\mathsf{a} : V_0 \times V_0 \to \mathbb{R}$ (→ Def. 0.3.1.4), linear form $\ell : V_0 \to \mathbb{R}$, $c \in \mathbb{R}$, be a quadratic functional on the vector space V_0.
If V_0 has finite dimension, then the quadratic minimization problem (→ Def. 1.2.3.11)

$$u_* = \operatorname*{argmin}_{u \in V_0} J(u)$$

always possesses a unique *solution.*

However, as we will see below infinite dimensional spaces hold a lot of surprises and existence of solutions of quadratic minimization problems becomes a subtle issue, even if the bilinear form is positive definite.

注意，在Lemma 1.2.3.39 Boundedness condition,我们也仅仅只能给出必要条件，也就是说即使bilinear form满足s.p.d。linear form满足boundedness condition，我们也不能一定确定该问题的解是存在的。

2.4 Linear Variational Problems

What does "grad$J = 0$" mean in ∞ dimensions? $\Rightarrow$ variational calculus

main idea of variational calculus is to reduce everything into one-dimension

这一最核心的问题是其论证了equivalence of quadratic minimization (unique minimizer) problem and linear variation problem (unique solution). （这一点有点疑惑的是没有对linear form做bounded，按道理来说这是解存在的必要条件）。其从quadratic minimization推到variational problem的手段是将无穷维的calculus reduce into one-dimension，然后再对这个one-dimension求导。这也是为什么我们的最后的variational equation中会出现test space。而这里我们可以看到test space其实就是subspace，而原先的trail space还是我们的affline space。接着给了一个例子，的出了second-order linear elliptic directlet problem的一般variational form（因为我们第一章全讨论的是second-order linear elliptic dirichlet problem.

Theorem 1.4.1.8. Equivalence of quadratic minimization problem and linear variational problem

For a (generalized) quadratic functional $J(v) = \frac{1}{2}a(v,v) - \ell(v) + c$ on a vector space V and with a symmetric positive definite bilinear form $a : V \times V \to \mathbb{R}$ are equivalent:

(i) The quadratic minimization problem for J has unique minimizer $u_* \in \widehat{V}$ over the affine subspace $\widehat{V} = g + V_0, g \in V$.

(ii) The linear variational problem

$$u \in \widehat{V}: \quad a(u,v) = \ell(v) \quad \forall v \in V_0 ,$$

has a unique solution $u_* \in \widehat{V}$.

A generic second-order linear elliptic Dirichlet problem $(*)$ in variational form seeks

$$u \in H^1(\Omega), \; u = g \text{ on } \partial\Omega: \quad \int_\Omega (\boldsymbol{\alpha}(\boldsymbol{x}) \, \mathbf{grad}\, u(\boldsymbol{x})) \cdot \mathbf{grad}\, v(\boldsymbol{x}) \, \mathrm{d}\boldsymbol{x} = \int_\Omega f(\boldsymbol{x}) v(\boldsymbol{x}) \, \mathrm{d}\boldsymbol{x} \quad \forall v \in H^1_0(\Omega) . \tag{1.4.2.4}$$

Symmetric uniformly positive definite material tensor $\boldsymbol{\alpha} : \Omega \mapsto \mathbb{R}^{d,d}$

Loading/source function

接下来的一大主题是我们讨论了对于second-order linear elliptic dirichlet problem。其问题的稳定性如何。essential是我们要定义norm，最后我们得出的结论也是在这个norm下的pertubation，会对结果在另一个norm下的pertubation如何。==目前仍不知道为什么要定义这么多norm，为什么不在统一的norm下讨论？难不成换成其他的norm结论还能不一样？这些推导挺有意思的，可以再仔细研究研究

然后，我们就要讨论如何求解minimization problem。我们知道对于quadratic functional，最小值就是导数为零的点。我们先拿一个finite dimension的例子做演示，经过一番推导，我们发现最后问题的求解转换成了linear system of equation (LSE).其中系数矩阵就是bilinear form所对应的矩阵。

那么，对于一个infinite dimensions的minimization problem，问题就变了，这时需要用到variational calculas。我们将我们研究的泛函问题确定在quadratic。再经过一些简单的推导，quadratic minimizaiton problem被等效成了linear variational problem，下面是对linear variational problem的描述。

Definition 1.4.1.6. (Generalized) Linear variational problem (LVP)

With V a vector space, $\widehat{V} \subset V$ an affine space, and $V_0 \subset V$ the associated subspace the equation

$$u \in \widehat{V}: \quad a(u,v) = \ell(v) \quad \forall v \in V_0 , \tag{1.4.1.7}$$

is called a (generalized) **linear variational problem**, if

- $a : V \times V_0 \mapsto \mathbb{R}$ is a bilinear form, that is, linear in both arguments (→ Def. 0.1.1.7),
- and $\ell : V_0 \to \mathbb{R}$ is a linear form (→ Def. 0.1.1.5).

trial space　　test space

同时，我们也给出了两个问题等效性描述

Theorem 1.4.1.8. Equivalence of quadratic minimization problem and linear variational problem

For a (generalized) quadratic functional $J(v) = \frac{1}{2}a(v,v) - \ell(v) + c$ on a vector space V and with a symmetric positive definite bilinear form $a : V \times V \to \mathbb{R}$ are equivalent:

(i) The quadratic minimization problem for J has unique minimizer $u_* \in \widehat{V}$ over the affine subspace $\widehat{V} = g + V_0, g \in V$.

(ii) The linear variational problem

$$u \in \widehat{V}: \quad a(u,v) = \ell(v) \quad \forall v \in V_0 ,$$

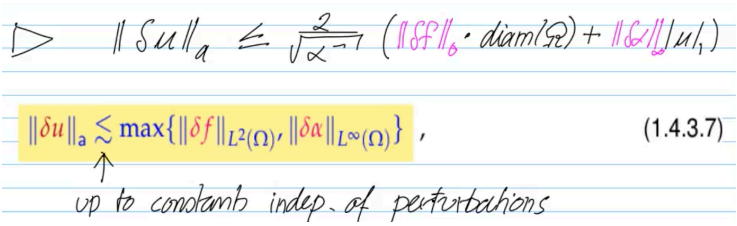
has a unique solution $u_* \in \widehat{V}$.

下面，我们看一些更加具体的问题。

接着，我们还研究了该问题的稳定性stability。这里的稳定性研究主要是外部载he

之所以要做这件事是因为对于任何一个问题如果你想要应用numerical methods去研究，你就需要研究stability，而研究stability的前提就是得定义norm，得在data space中的一些微小扰动会对result space产生什么样的影响。这是问题的关键。

我们最后得到的结论是up to constants independent of pertubations.



Q&A

- Linear form = linear functional
- continuous = bounded

?unbounded linear functional

Affline space的理解:

我们都学过欧几里德空间，同时在线形代数课上老师让你们证明了任何通过原点的直线或者平面是这个欧几里德线形空间的子线形空间。

那么，那些没有通过原点的直线或者平面可以理解为仿射空间的一个例子。它们在一定程度上具有线性。

大家通俗的把仿射空间看成一个没有原点的线形空间。当我们讲线性空间的时候，脑子记的第一反应是这个空间必须有零元素，如果具体化举欧几里得空间的例子就是说原点必须在这个空间里。所以仿射空间不是线性空间。另外，向量空间，也是线性空间，可以没有定义范数(norm),这里的向量就是一个抽象的概念。定义了范数(norm)的向量空间就定义了范数的向量空间了。我猜你讲的向量可能是具体到欧几里德空间里的向量了。向量通过原点这个说法我觉得有问题，向量是空间中的元素，同时你所指的原点也是空间中的一个元素。你说的过原点是指把欧几里德空间和笛卡尔坐标系联系起来了，任何一个向量对应到笛卡尔坐标系上就是从原点出发指向坐标的箭头了。建议你翻数学辞典翻原始定义。再通过具体例子来理解抽象概念，但不要搞混

线性空间:

回顾一下线性空间的定义：对两种运算封闭，并且满足八条运算法则（即中分配律、结合律、交换律、零元素、负元素）

定义7（线性流形/仿射空间，linear manifold/affine space）：设 X 是一个线性空间，M 是 X 的一个子空间，M 在 X 中作某个平移（不妨记为 $x_0 \in X$）后所构成的向量集合 $V := x_0 + M = \{x_0 + m | m \in M\}$，称为 X 中的一个线性流形/仿射空间。

例8：非齐次线性方程组 $Ax = b$ 的解空间 $V = \{x | Ax = b\}$ 构成一个线性流形。

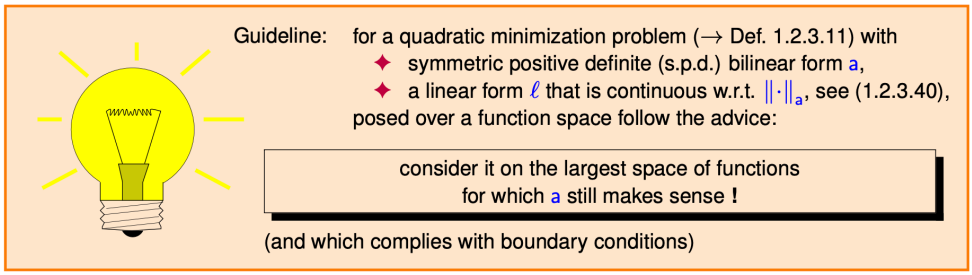
> 记 $M = \{x | Ax = \theta\}$ 是齐次线性方程组 $Ax = \theta$ 的解空间，记 x_0 是非齐次线性方程组的一个特解，即 $Ax_0 = b$。根据线性方程组理论，$Ax = b$ 的所有解可表示为通解与特解的叠加，即 $V = \{x + x_0 | x \in M\} = x_0 + M$，是一个线性流形。

模型都是通过真实物理系统引出来的，这里我们先考虑一类物理系统的建模，即equalibrium models。其问题最后的统一描述即系统能量最小化，基于此即刻得到我们系统所处的微分方程方程。注意我这里没有提及边界条件，因此给出的是系统general的状态描述。这种所谓的状态描述我们可以通过能量泛函的最小来表达，也可以通过对能量泛函一阶变分=0来进行描述，这两种描述是否等价有待考究。

1. 对于所研究的平衡性问题，我们首先考虑的是其模型的configuration space，然后我们发现configuration space是一个infinite-dimension的函数空间。当边界条件固定时，不同的外力会导致不同的resulting solution。这时函数空间变化为一个连续且受限（满足BC）的函数空间，此函数空间时affine vector space. 2. 然后，我们讨论了平衡性条件，该平衡性条件是建模的关键，即我们知道了解的存在依赖于什么约束。从而我们发现我们之前讨configuration space有点太强了，piece-wise continuous即可（考虑了point loading）。关于如何写出系统的能量函数，这一章节直接给出，具体的如何推导出来我们会在后面讨论。讨论完elastic energy模型后，我们又讨论了electrostaci energy模型，其中u是电势，是scalar，而u的梯度即是电场，讨论的是一个三维问题。⚠️这里还要注意的是dielectric tensor的fundamental property，这对我们后面讨论configuration space非常有用。3. 我们总共讨论了三个问题，两个是elastic model，有一个是elctrostatic model，这三个问题的通性是他们全是quadratic functional，因此我们这里的问题转换成了quadratic functional的minimization problems.但是需要强调的是，他们又不是完美的quadratic problem，这是因为configuration space不是real vector space，而是affine space。但是通过简单的变化，我们发现，对于在affine space上的minimizaiton problem又可以转化为一个满足边界条件的u_0加上经过变化后在vector space上的minimization problem。因此，我们的研究目标将为the minimization problem of quadratic functional.接着，我们讨论了对于quadratic functional，其解的存在性与唯一性，给出了很多的必要条件：首先，如果解只是存在，那么我们只需要bilinear form满足positive semi-definite，这个是必要条件。如果想要解是existence且uniqueness，那么需要bilinear form满足positive definite，这个是充分必要条件？紧接着我们就根据bilinear form是s.p.d从而定义energy norm，有这个energy norm不太需要重新定义norm了。最后我们如果对linear form做一定的限制，那么解将会是bounded。Actually, what is the different between uniquess and bounded?

4. 接着我们发现，即使当bilinear form满足s.p.d.，linear form满足continuous,对于function space，我们的解同样不存在（这说明上述的讨论其实只是必要条件，而不是充分必要条件）。因此这便是我们的动机"Mathematical theory is much concerned about proving existence of suitably defined solutions for minimization problems"。但是这我们encounter profound problems，这说明我们的function space取得太小了，因此，为了讨论最小值的存在，我们要进一步扩大我们的function space，因此a class of abstract function spaces has been devised to deal with the question of existence of solutions of QMP.

   因此，接下来问题的关键回到了如何choose the right function space for QMP. 在这个过程中，energy form将会起到关键性作用，我们首先给出了下面的考虑



因此，对于上诉的问题，我们给出了下面的建议：也就是energy form的平方不是无限大。（这里需要注意的是我们的BC还是需要满足的，其他的关于函数具有什么样的形式，连续还是不连续又或者是否部分连续我们都无所谓了）

In the concrete case of quadratic minimization problems like (1.2.3.1b) (minimization of potential energy of a membrane) and (1.2.3.1c) (minimization of the energy of an electrostatic field) we arrive at the following recommendation:

Choose "$V_0 := \{\text{functions } v \text{ on } \Omega\text{: } a(v,v) < \infty\}$"

Then V_0 is called the **energy (function) space** generated by the energy norm $\|\cdot\|_a$.

首先我们考虑了$L^2(\Omega)$function space,其研究的泛函主要是一下形式：$J(u) := \int_\Omega \frac{1}{2}|u(x)|^2 - u(x)\,dx$. 。其energy norm的平方小于∞所确定的function space就是$L^2(\Omega)$，这里我们称其norm就是：

Definition 1.3.2.3. Space $L^2(\Omega)$ → Def. 0.3.2.27

The function space defined in (1.3.2.2) is the space of square-integrable functions on Ω and denoted by $L^2(\Omega)$.

It is a normed space with norm $\left(\|v\|_0 := \right) \|v\|_{L^2(\Omega)} := \left(\int_\Omega |v(x)|^2\,dx\right)^{1/2}$.

✎ Notation: $L^2(\Omega)$ ← superscript "2", because square in the definition of norm $\|\cdot\|_0$

- 这里也可以看到其实不同的extended function space其本质就是不同的energy norm所决定的。

但是，当我们采用上述的extended function space，我们发现了一个问题，就是我们的boundary condition似乎被忽略了，通过一大堆的论证和解释（见example 1.3.2.5），我们得出了如下的结论：

Boundary conditions cannot be imposed in $L^2(\Omega)$!

This example has taught us an important lesson:

Energy-based function spaces and boundary conditions

Given a function space V_0 "defined" through its energy norm, only boundary conditions compatible with the energy norm can be imposed on functions $\in V_0$.

- 所以边界条件到底怎么考虑啊？？似乎是在开区间上讨论解的存在性？而不是在闭区间上讨论？？后面扩展的内容中，当$L^2(\Omega)$是Hilber space时，根据定理，我们知道在开区间上，解存在且唯一。

⇓ 接着，我们考虑另一种quadratic functional形式的energy space。$J(u) := \int_\Omega \frac{1}{2}\|\mathbf{grad}\, u\|^2 - f(x)u(x)\,dx$ $\left(u \in C^1_{pw,0}(\Omega)\ ?\right)$

Definition 1.3.4.3. Sobolev space $H^1_0(\Omega)$

The space of integrable functions on Ω with square integrable gradient that vanish on the boundary $\partial\Omega$,

$$V_0 := \{v : \Omega \mapsto \mathbb{R} \text{ integrable: } v = 0 \text{ on } \partial\Omega,\ \int_\Omega |\,\mathbf{grad}\, v(x)|^2\,dx < \infty\}\,, \qquad (1.3.4.2)$$

is the Sobolev space $H^1_0(\Omega)$ with norm

$$|v|_{H^1(\Omega)} := \left(\int_\Omega \|\mathbf{grad}\, v\|^2\,dx\right)^{1/2}.$$

✎ Notation: $H^1_0(\Omega)$ ← superscript "1", because first derivatives occur in norm

← subscript "0", because zero on $\partial\Omega$

✎ Notation: Alternative notations for H^1-norm: $|v|_{H^1(\Omega)} = |v|_{1,\Omega} = |v|_1$, where the last notation is used, when the domain is clear from the context.

- 这里我们称之为Sobolev space $H_0^1(\Omega)$。⚠️不同的是，这个sobolev space不仅仅用energy norm的平方小于无限大确定，同时也施加了边界条件，这可能是因为边界条件compatible with the energy norm。关于此点的讨论我们
- 很重要的一点是，我们知道H_0^1其实是个很大的function space，其同时也包括unbounded functions like point evaluation. unbounded的另一种表述是not a continuous linear form on $H^1(\Omega)$

后面，我们又drop BDC of $H_0^1(\Omega)$，从而得到了最大的function space $H^1(\Omega)$。定义如下：

Definition 1.3.4.8. Sobolev space $H^1(\Omega)$

The Sobolev space

$$H^1(\Omega) := \{v : \Omega \mapsto \mathbb{R} \text{ integrable: } \int_\Omega |\mathbf{grad}\, v(\boldsymbol{x})|^2 \,\mathrm{d}\boldsymbol{x} < \infty\}$$

is a normed function space with norm

$$\|v\|_{H^1(\Omega)}^2 := \|v\|_0^2 + |v|_{H^1(\Omega)}^2 .$$

- 在这个sobolev space中，norm表述不单单是energy norm，是L^2 和Sobolev space H_0^1的叠加（理解不了啊）

接着，我们抽象的讨论完这些sobolev space，我们想再次回到我们的QMP问题，因此，我们对问题进行了新的描述，把之前的piecewise continuos function space全部换成了sobolev space。然后我们拿了2D membrane来进行研究。我们发现，当a为s.p.d.时，f(x)(唯一一个外界输入量，非系统本身)在$L^2(\Omega)$时，系统具有unique minimizer on $H_0^1(\Omega)$。在这个推导过程中，我们用到了cauchy-schwarz inequality和first poincare-friedrichs inequality.

Corollary 1.3.4.19. Admissible loading/source functions linear 2nd-order elliptic problems

If $f \in L^2(\Omega)$, then $\ell(v) := \int_\Omega f(\boldsymbol{x})v(\boldsymbol{x})\,\mathrm{d}\boldsymbol{x}$ is a continuous linear functional on $H_0^1(\Omega)$.

As in Section 1.2.3.4 in this lemma “continuity” has to be read as

$$\exists C > 0: \quad |\ell(u)| \le C|u|_{H^1(\Omega)} \quad \forall u \in H_0^1(\Omega) \,. \tag{1.2.3.43}$$

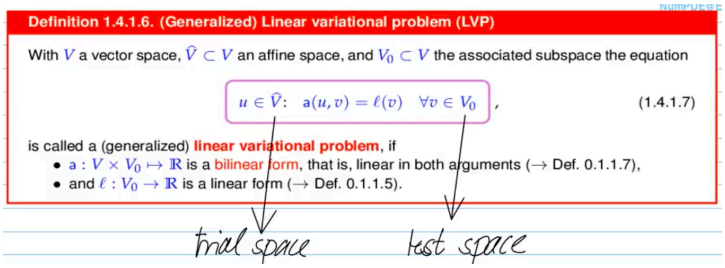
- 还有个值得注意的点是linear form的continuity可以被翻译为是满足对energy norm的有界性。

接着，我们又讨论了一下关于sobolev space的其他信息，核心就是说当piecewise smmoth functions通常约等于$H^1(\Omega)$，因此可以在精神上讲piecewise smooth fnction替换成$H^1(\Omega)$

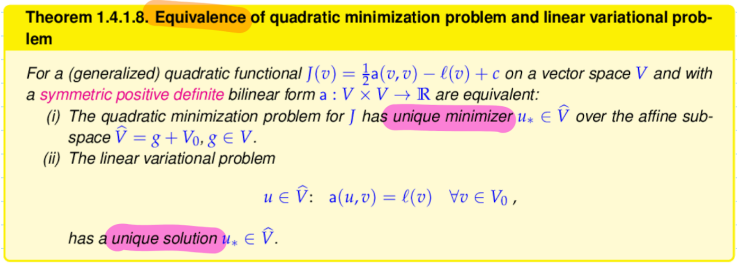
5. 接着我们研究线性变分问题。

对于不是solution space不是无穷维的情况，对于其minimizer，通常矩阵代数直接取导数为0即可。最后得到的是一个linear system of equaitons 具体的推导流程再熟悉一下

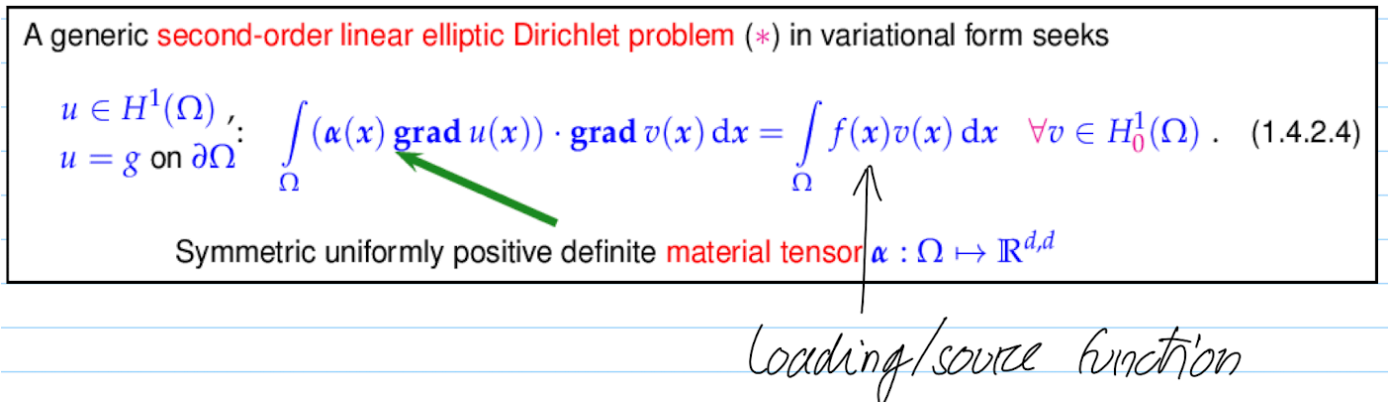
而对于函数空间无限维的情况，我们怎么来考虑，这种情况我们称之为quadratic variational calculus，经过一番推导，我们最后得到linear variation equaiton。



- 同时我们也知道，if LVP has a solution, then it will be unique if a is s.p.d. 这就是QMP和LVP的等价性



然后，我们举了一个具体的例子，linear second-order variational problems，这个地方是对静电场举例。这里我其实不太明白为什么涉及到对sobolev space和affine space 的讨论。之后再好好考虑一下



最后，我们讨论了系统的稳定性，关注的问题是当不属于系统固有的量（外界输入量，$\alpha(x)$和$f(x)$）变化时，解的稳定性。

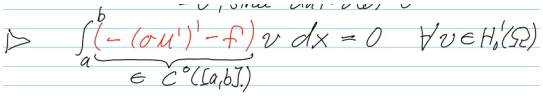
Week 2

Several key points to understand the functional and linear variational problem:

- Solution space (L^2 and Soblev space)
- minimization problem of quadratic functional = linear variational problems
  - for functional, the functions ofter exist in the form of piecewise.
  -

2.5 Equlibrium Models BVP

integration by part的一个用处就是转移导数，同时产生boundary term。在PDE中，我们对解的光滑性做了进一步假设之后才能导出藏在PDE中的solution. 为什么我们那样假设光滑性，理由就在这里：



Assumption 1.5.1.2. Smoothness required for two-point boundary value problems

The following smoothness requirements have to be satisfied

$$u \in C^2([a,b]) \quad , \quad \sigma \in C^1([a,b]) \quad , \quad f \in C^0([a,b]) \, . \tag{1.5.1.3}$$

为什么我们称dirichlet BVP，是因为求解域上的点是已知的。

这一章节的核心目标是推导出LVP中潜藏的PDE。在这个推导的过程中，我们需要用到分部积分。而分部积分会转移导数，所以我们对解的光滑性需要做进一步的要求。因此最后得到的PDE的解我们称之为strong solution。而LVP的解我们称之为weak solution。在推导的过程中，我们通常结合product rule和分部积分公式进行求解。高维的分部积分公式成为Gauss theorem,高维的product rule称之为general product rule。而两者的结合则成为Green's first formula

当我们解决的是dirichlet BC(也就是BC上的值是prescribed了），问题很简单。而如果LVP里面藏着Neumann BC，则我们再倒出pde形式的时候test space的满足要分两部分，一个是在H_0^1（test space H^1 + 部分边界给定）下先讨论出PDE的形式，接着再讨论出剩下那一项为0。

Up till now, we haven't seen the PDE already. Then, we discussed how to derive the PDE from the linear variation problems by using integration by parts. we need to do two things. Fristly, we need to shrink our function space and assume more smoothness. Seen as follows:

$$u_* \in H^1(]a,b[) \, , \quad u(a) = u_a, \; u(b) = u_b \; : \quad \int_a^b \sigma(x) \frac{du}{dx}(x) \frac{dv}{dx}(x) \, \mathrm{d}x = \int_a^b f(x) v(x) \, \mathrm{d}x \quad \forall v \in H_0^1(]a,b[) \, , \tag{1.4.2.1}$$

with a uniformly positive stiffness coefficient function $\sigma \in C^0_{\mathrm{pw}}([a,b])$ and $f \in L^2(]a,b[)$.

We require/assume more smoothness

Assumption 1.5.1.2. Smoothness required for two-point boundary value problems

The following smoothness requirements have to be satisfied

$$u \in C^2([a,b]) \quad , \quad \sigma \in C^1([a,b]) \quad , \quad f \in C^0([a,b]) \, . \tag{1.5.1.3}$$

vs. : $u \in C^1_{\mathrm{pw}}([a,b]) \quad , \quad \sigma \in C^0_{\mathrm{pw}}([a,b]) \quad , \quad f \in C^0_{\mathrm{pw}}([a,b]) \, .$ (1.5.1.4)

for LVP:

Then by using integration by part, we can derive the PDE which is the strong form of this problem (vs. "weak form" = LVP). ⚠️During this process, we will derive the bounder term, which will be equal to zero due to the test funciton $v \in H_0^1(]a,b[)$. Furthermore, we have meet the situation that the boundary term is not equal to zero automatically. Under such circumustance, we derive a "hidden" boundary condition, which is called Neumann boundary condition.

2.6 Diffusion Models

oritented surface: a surface with crossing direction. Highlight a different approach to get the PDE for BVP. 其实就是单元体的conservation law。然后我们结合了fourier law，推导出了stationary heat conduction。这里最后提到，因为我们推导出来的形式和Membrane的平衡方程很像，所以在其背后应该有相似的Variational form。但是这里老师没继续提了

2.7 Boundary Conditions

提出well-posed的概念，well-posed的problem我们才可以讨论起E&U, stability. PDE + BDC ⇒ BVP 也就是说，光说求解PDE是不准确的，只有所有的boundar上都存在boundary conditions，解才可能存在并且唯一。这里我们通过Heat flow这个例子来说明在这个context下，几种不同的boundary condtion：Dirichlet BC, Neumann BC and Radiation BC.其实我觉得在学习PDE，一定要从model出发，从问题出发。在第一章节中，我们全讨论的是elliptic second-order problem。并且，我们从两个角度进行讨论，第一个角度，我们研究的对象是membrane，我们从能量最小的角度出发（这是很符合我们对这个问题的物理直觉的），得到了quadratic minimization functional，接着给出了等价的variational equation。这些都是方程的弱形式。接着，我们推导了藏在这个variational equation中的PDE，以及相应了引出了sobolev space和BC。而在这两章，我们却换了个角度，对于heat Diffusion这个问题，基本的物理直觉是conservation law（而不是energy minimization），我们基于此，推导出了PDE，发现这个PDE和我们讨论的membrane方程非常相似，因此我们就认为，对于这个方程，应该也具有相似的variationa equation，但老师这里没讲。但是，我们这里直接告诉你，单纯的PDE并不是well-posed的，因为解不是unique。因此这里我们指出需要BC，并且讨论了三种潜在的BC。并且，最后我们给出了一个非常重要的结论（仅对second order elliptic boundary value problems）：For second order elliptic BVP, exactly one boundary condition is needed on every part of $\partial\Omega$.

2.8 Second Order Elliptic BVPs

这一章节的核心就是如何从BVPs推出来等价的linear variational problems。还是以heat diffusion举例，列举了三种不同边角条件下如何推导相应的linear variational form。

2.9 Essential and Natural Boundary Conditions

Week 3

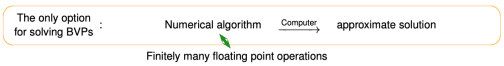
3 Finite Element Method (FEM)

在力学上，我们对有限元的理解和网格划分，我们首先划分网格，基于插值函数，以节点作为interpolation condition然后插值得到单元内的应力或者应变场。如果是线性问题，直接求解LSE。如果不是线性问题，用newton iteration使得week form的residual为零，从而得到节点上的应力应变值。再得到节点上的应力值之后，再插值得到整个域上的应力场，然后再求导以及通过本构得到整个场的其他物理量。

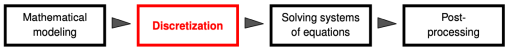
在力学中的有限元，存在单元刚度矩阵组装成域内的节点刚度矩阵的概念，而在数学的角度，在我们定义的sobolev space中，我们直接求得整个体系的Galerkin matric

3.1 Introduction

Our goal is to solve linear scalar second-order elliptic boundary value problem as introduced in Chapter 1 in various forms, as a minimization problem (1.2.3.1), a linear variational problem like (1.4.2.4) ("weak form"), or in strong form (1.5.3.5). [No analytic solution ⇒approximate solution]

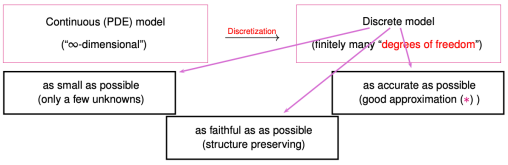


According to our understanding, PDE models inherently rely on infinite-dimensional state spaces. Thus it takes an infinite amount of information to characterize the solution. Since computers are finite automata, (numerical) algorithms can operate only on discrete models.



- **Definition 2.1.1.1. Discrete model**
  A **discrete** model for a physical system/phenomenon is a model, for which both data/parameters and states can be described by a finite number of real numbers.

We will start our discussion from the following prospectives:



It should be noted that we devoted to the discretization of linear second-order elliptic BVP by means of the so-called finite element method.

- 线性泛函，其variational form可以表示成这种形式： $u \in \widehat{V}$: $\mathsf{a}(u,v) = \ell(v) \quad \forall v \in V_0$,

☛ 2nd-order elliptic Dirichlet problem:

$$u \in H^1(\Omega)\,,\quad u = g \text{ on } \partial\Omega: \quad \int_\Omega (\boldsymbol{\alpha}(\boldsymbol{x})\,\mathbf{grad}\,u(\boldsymbol{x}))\cdot\mathbf{grad}\,v(\boldsymbol{x})\,\mathrm{d}\boldsymbol{x} = \int_\Omega f(\boldsymbol{x})v(\boldsymbol{x})\,\mathrm{d}\boldsymbol{x} \quad \forall v \in H^1_0(\Omega)\,, \qquad (1.4.2.4)$$

☛ 2nd-order elliptic Neumann problems:

$$u \in H^1_*(\Omega): \quad \int_\Omega (\boldsymbol{\alpha}(\boldsymbol{x})\,\mathbf{grad}\,u)\cdot\mathbf{grad}\,v\,\mathrm{d}\boldsymbol{x} = \int_\Omega fv\,\mathrm{d}\boldsymbol{x} + \int_{\partial\Omega} h\,v\,\mathrm{d}S \quad \forall v \in H^1_*(\Omega)\,, \qquad (1.8.0.16)$$

Mission of FEM: Discretization → conversion into a problem posed on a finite-dimensional space

3.2 Properties of Galerkin Discretization

In this section we adopt an abstract perspective: Targets of discretization will be linear variational problems (1.4.1.7) of the special form:

$$u \in V_0\!: \quad \mathsf{a}(u,v) = \ell(v) \quad \forall v \in V_0\,, \qquad (2.2.0.2)$$

✦ V_0 ≙ a vector space (Hilbert space) (usually a Sobolev space → Section 1.3) with norm $\|\cdot\|_V$,
✦ $\mathsf{a}(\cdot,\cdot)$ ≙ bilinear form, *continuous* (→ Def. 1.2.3.42) on V_0, which means

$$\exists C > 0\!: \quad |\mathsf{a}(u,v)| \le C\|u\|_V\|v\|_V \quad \forall u,v \in V_0\,. \qquad (2.2.0.3)$$

✦ ℓ ≙ *continuous* linear form in the sense of Def. 1.2.3.42, *cf.* (1.2.3.40),

$$\exists C > 0\!: \quad |\ell(v)| \le C\|v\|_V \quad \forall v \in V_0\,. \qquad (2.2.0.4)$$

- The importance of this continuity is discussed in the begining of Section 1.2.3.4.

- If a is symmetric and positive definite (→ Def. 1.2.3.27), we may choose $\|\cdot\|_V := \|\cdot\|_{\mathsf{a}}$, the "energy norm" of Def. 1.2.3.35. By the Cauchy-Schwarz inequality of Thm. 0.3.1.19 continuity of a w.r.t. $\|\cdot\|_{\mathsf{a}}$ is clear. ⌟

Galerkin Discretization: 1st Step

The simple idea of the first step of Galerkin discretization:

Replace the infinite-dimensional function space V_0 in the linear variational problem (2.2.0.2) with a finite-dimensional subspace $V_{0,h} \subset V_0$.

- Note that a subscript tag h distinguishes "discrete functions/quantities", that is, functions/operators etc. that are associated with a finite dimensional space. The symbol h has its origin in a widely used notation for the stepsize/meshwidth underlying a discretization, see § 2.3.1.3 below.
- The core is to change our configuration space, and solve it in the finite-dimensional space.

Discrete variational problem (DVP) ⇔ Discrete quadratic minimizatin problem (DQMP)

- 如果不化成变分，则是泛函问题的直接解法了。就是假设解函数形式（将解函数参数化）。这时，泛函问题转换成了函数问题，自变量是这几个参数而不是函数了，这时候导数为0直接求解即可。但是困难的地方是：首先假设的解函数set得满足泛函对解的约束条件。其次，你最后求得的解只是在这个解空间中的最优解，不代表其就是本来在所有解空间中的那个真实解。
- 如果化成变分形式，则从最小值问题转化成了方程问题。然后用有限元离散即可求解。对于线性问题，有限元后我们可以直接将方程转化成LSE。对于非线性问题，就是Nonlinear LSE，用数值里面的iterative methods进行求解。
- 如果进一步采用分部积分化解变分形式，我们最后会得到PDE形式。

$$
\begin{aligned}
&u_h \in V_{0,h}\colon\ \mathsf{a}(u_h, v_h) = \ell(v_h) \quad \forall v_h \in V_{0,h}\,. && \text{(2.2.1.1)}\\
&\qquad\Updownarrow\quad \left[\begin{array}{rcl} u_h &=& \mu_1 b_h^1 + \cdots + \mu_N b_h^N, \mu_i \in \mathbb{R} \\ v_h &=& \nu_1 b_h^1 + \cdots + \nu_N b_h^N, \nu_i \in \mathbb{R} \end{array}\right]\\
&\sum_{k=1}^{N}\sum_{j=1}^{N} \mu_k \nu_j \mathsf{a}(b_h^k, b_h^j) = \sum_{j=1}^{N} \nu_j \ell(b_h^j) \quad \forall \nu_1, \ldots, \nu_N \in \mathbb{R}\,,\\
&\qquad\Updownarrow\\
&\sum_{j=1}^{N} \nu_j \left(\sum_{k=1}^{N} \mu_k \mathsf{a}(b_h^k, b_h^j) - \ell(b_h^j) \right) = 0 \quad \forall \nu_1, \ldots, \nu_N \in \mathbb{R}\,,\\
&\qquad\Updownarrow\quad (*)\\
&\sum_{k=1}^{N} \mu_k \mathsf{a}(b_h^k, b_h^j) = \ell(b_h^j) \quad \text{for } j = 1, \ldots, N\,.\\
&\qquad\Updownarrow\quad [\vec{\mu} = (\mu_1, \ldots, \mu_N)^\top \in \mathbb{R}^N]\\
&\boxed{\mathbf{A}\vec{\mu} = \vec{\varphi}}\,, \text{ with } \quad \mathbf{A} = \left[\mathsf{a}(b_h^k, b_h^j)\right]_{j,k=1}^{N} \in \mathbb{R}^{N,N}\,,\\
&\qquad\qquad \vec{\varphi} = \left[\ell(b_h^j)\right]_{j=1}^{N}\,.
\end{aligned}
$$

A linear system of equations (LSE)

- 最后，我们就是在这个基函数下。解出来**存在且唯**一的满足linear variational form的solution。
- 整理理解上看：∞维的函数空间我们不太知道怎么去处理，很难的得到解析解（第一章讨论了挺多直接求解解析解的情况）。这一章则是转换成有限维度，问题不是LSE就是Nonlinear LSE，我们一定可以求解。
- ⚠ **Theorem 2.2.2.6. Independence of Galerkin solution of choice of basis**

  The choice of the basis $\mathfrak{B}_h$ has no impact on the (set of) Galerkin solutions u_h of (2.2.1.1).

👇求解该问题的关键转换为了obtain the Galerkin Matrices (We call it stifiness matrices in the mechanics)

Although we know that the final solution didn't rely on the **basis**. But the Galerkin matrices and right hand side vector is totally determined by the basis.

Lemma 2.2.3.2. Effect of change of basis on Galerkin matrix

Consider (2.2.1.1) *and two bases of* $V_{0,h}$,

$$\mathfrak{B}_h := \{b_h^1, \ldots, b_h^N\} \quad , \quad \widetilde{\mathfrak{B}}_h := \{\widetilde{b}_h^1, \ldots, \widetilde{b}_h^N\} ,$$

related by the basis transformation matrix $\mathbf{S}$ *according to*

$$\widetilde{b}_h^j = \sum_{k=1}^{N} s_{jk} b_h^k \quad \text{with} \quad \mathbf{S} = \left[s_{jk}\right]_{j,k=1}^{N} \in \mathbb{R}^{N,N} \text{ regular.} \tag{2.2.3.3}$$

Then the Galerkin matrices $\mathbf{A}, \widetilde{\mathbf{A}} \in \mathbb{R}^{N,N}$, *the right hand side vectors* $\vec{\varphi}, \widetilde{\vec{\varphi}} \in \mathbb{R}^N$, *and the coefficient vectors* $\vec{\mu}, \widetilde{\vec{\mu}} \in \mathbb{R}^N$, *respectively, satisfy*

$$\widetilde{\mathbf{A}} = \mathbf{S}\mathbf{A}\mathbf{S}^\top \quad , \quad \widetilde{\vec{\varphi}} = \mathbf{S}\vec{\varphi} \quad , \quad \widetilde{\vec{\mu}} = \mathbf{S}^{-\top}\vec{\mu} . \tag{2.2.3.4}$$

✍ notation: $\mathbf{S}^{-\top} := (\mathbf{S}^{-1})^\top = (\mathbf{S}^\top)^{-1}$ for any $\mathbf{S} \in \mathbb{R}^{N,N}$

- The relationship $\widetilde{\mathbf{A}} = \mathbf{S}\mathbf{A}\mathbf{S}^\top$ between matrices has a special name in linear algebra:

Definition 2.2.3.5. Congruent matrices

Two matrices $\mathbf{A} \in \mathbb{R}^{N,N}$, $\mathbf{B} \in \mathbb{R}^{N,N}$, $N \in \mathbb{N}$, are called congruent, if there is a regular matrix $\mathbf{S} \in \mathbb{R}^{N,N}$ such that $\mathbf{B} = \mathbf{S}\mathbf{A}\mathbf{S}^T$.

- What properties are shared by all Galerkin matrices without knowing the basis for a DVP:

matrix properties invariant under congruence :
- regularity → [Hip19, ??]
- symmetry ($\mathbf{A}^\top = \mathbf{A}$)
- positive definiteness → [Hip19, ??]

- Take-home message:

Thm. 2.2.2.6: the choice of $\mathfrak{B}_h$ does not affect u_h ⇒ No impact on discretization error ! (assuming exact arithmetic)

But: Key properties (e.g., conditioning, sparsity) of the Galerkin matrix crucially depend on $\mathfrak{B}_h$!

- The choice of trial/test space $V_{0,h}$ *alone* determines the quality of the Galerkin solution.
- The choice of basis $\mathfrak{B}_h$ determines how well (stably, efficiently) we can compute the Galerkin solution.

总结，就是对于linear form variational的形式，我们无法在无穷维空间中进行求解，因此我们就在有限维空间中进行求解，问题的解最后化成了求解LSE。通过研究这个问题，我们也发现，只要离散的trial/test选定，最后的解就确定了。Galerkin matrices并不影响最后的解，但会影响我们求解最后解过程中是否在数值上稳定可靠（稀疏性、稳定性、高效性）因此，我们还是应该选择好一点的basis。从而使得我们的galerkin matric有规律一些，方便我们后续求解。

3.3 Case Study: Linear FEM for Two-Point Boundary Value Problems

A 2-pt Dirichlet BVP: $-u'' = f$ in $]a,b[$, $u(a) = u(b) = 0$.

LVP form: $u \in H_0^1(]a,b[)$: $\underbrace{\int_a^b \frac{du}{dx}(x)\frac{dv}{dx}(x)\,\mathrm{d}x}_{=:a(u,v)} = \underbrace{\int_a^b f(x)v(x)\,\mathrm{d}x}_{=:\ell(v)} \quad \forall v \in H_0^1(]a,b[)$.

- 这个$f(x)$不是方程本身属性量，相当于是外界输入响应，在neural opertor里，我们通常也是学习这种歌外界输入相应、边界条件对解函数的影响。而$f(x)$ is usually given in prodecural form.

GD step I: choice of discrete trail & test space

Approximation in the FEM

Idea underlying the finite element method:

approximate u by means of continuous, *piecewise polynomial* functions.

Therefore we equip $\Omega = [a,b]$ with $M+1$ **nodes** ($M \in \mathbb{N}$) forming the set

$$\mathcal{V}(\mathcal{M}) := \{a = x_0 < x_1 < \cdots < x_{M-1} < x_M = b\} .$$

The nodes define small intervals that constitute a **mesh/grid**

$$\mathcal{M} := \{]x_{j-1}, x_j[: 1 \le j \le M\} .$$

The intervals $[x_{j-1}, x_j]$, $j = 1, \ldots, M$ are the **cells** of the mesh $\mathcal{M}$, which is often identified with the set of its cells. A special case is an **equidistant** mesh with uniformly spaced nodes:

$$x_j := a + jh , \quad h := \frac{b-a}{M} .$$

The local and global resolution of a mesh/grid is measured through two quantities, the

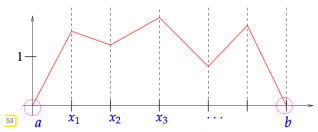
(local) **cell size** $h_j := |x_j - x_{j-1}|, j = 1, \ldots, M$

(global) **meshwidth** $h_{\mathcal{M}} := \max_j |x_j - x_{j-1}|$

- We have **M+1** nodes and **M** meshes.

Here, we consider the simplest situation:

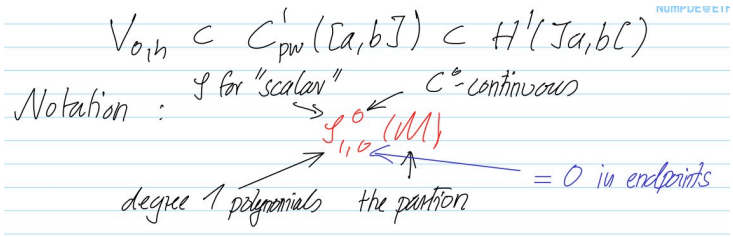
Thus, *continuous* piecewise polynomials provide valid trial and test functions for the variational problem (2.3.0.1). We give a first example.



The simplest space of continuous, $\mathcal{M}$-piecewise polynomial functions in $H_0^1(]a,b[)$:

$$V_{0,h} = \mathcal{S}^0_{1,0}(\mathcal{M}) := \left\{ \begin{array}{l} v \in C^0([a,b]) : v_{|[x_{i-1},x_i]} \text{ linear}, \\ i = 1, \ldots, M, v(a) = v(b) = 0 \end{array} \right\}$$

$\Uparrow$ a function $\in \mathcal{S}^0_{1,0}(\mathcal{M})$ $\quad\rightarrowtail\quad$ $N := \dim \mathcal{S}^0_{1,0}(\mathcal{M}) = M - 1$

- ⚠ Piecewise polynomial必须得是C^0。这是因为其是H_0^1的子空间。在力学上的物理意义就是解不能出现断层或者不连续。
- 

- **Remark**: $\mathcal{S}^0_{1,0}(\mathcal{M}) \not\subset C^1([a,b])$
  - 因为我们研究的是在be used in DVP, since smoothness requirements for variational form relaxed compared with the PDE form. (这也是合理的，其实单纯的PDE考虑的解空间其实是不完全的，PDE通常要求解有更多的连续性，一阶连续，但是这在实际情况中并没有考虑point load。因此，minimization functional才是我们最原始的物理模型，PDE只是增加了更多的约束的导出形式。这也解释了为什么NS方程可能解不存在，因为其解可能在这种连续性假设下不存在。)

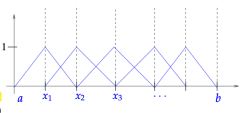
GD step II: Choice of basis

Natural (& convenient) choice: <tent functions

Our choice of the (ordered) basis $\mathfrak{B}_h\{b_h^1, \ldots, b_h^{M-1}\}$ of V_h is the following:

1D "**tent functions**" [Hip19, ??] ▷

$$b_h^i(x) := \begin{cases} (x - x_{j-1})/h_j & \text{, if } x_{j-1} \le x \le x_j , \\ (x_{j+1} - x)/h_{j+1} & \text{, if } x_j \le x \le x_{j+1} , \\ 0 & \text{elsewhere.} \end{cases}$$



(2.3.2.1)

$\blacktriangleright$ $\quad b_h^i(x_i) = \delta_{ij} := \begin{cases} 1 & \text{, if } i = j , \\ 0 & \text{, if } i \ne j . \end{cases}$ (2.3.2.2)

- We point out that (2.3.2.2) qualifies the tent functions as a **cardinal basis** of $\mathcal{S}^0_{1,0}(\mathcal{M})$ with respect to the node set $\mathcal{X} \equiv x_i$. As a consequence the basis expansion coefficients for any function in $\mathcal{S}^0_{1,0}(\mathcal{M})$ are given by the function values at the interior nodes of the mesh: $u_h \in \mathcal{S}^0_{1,0}(\mathcal{M}) \;\Leftrightarrow\; u_h = \sum_{i=1}^{M-1} u_h(x_i) b_h^i$.
- Deriviate:
  - $\blacktriangleright$ $\quad \frac{db_h^i}{dx}(x) = \begin{cases} \frac{1}{h_j} & \text{, if } x_{j-1} \le x \le x_j , \\ -\frac{1}{h_{j+1}} & \text{, if } x_j < x \le x_{j+1} , \\ 0 & \text{elsewhere.} \end{cases}$ (2.3.2.6)

- $\mathcal{M}$-p.w. constant and discontinuous. OK for variational form, because we integrate. (但是这对于PDE form是无法存在的。)

GD step III: Formulas for Galerkin matrices & r.h.s vectors

$$\underbrace{\int_a^b \sum_{l=1}^N \mu_l \frac{db_h^l}{dx}(x)\frac{db_h^k}{dx}(x)\,dx}_{=a(u_h,b_h^k)} = \underbrace{\int_a^b f(x)b_h^k(x)\,dx}_{=\ell(b_h^k)}\ ,\quad k=1,\ldots,N\,.$$

$$\Updownarrow$$

$$\sum_{l=1}^N \left(\int_a^b \frac{db_h^l}{dx}(x)\frac{db_h^k}{dx}(x)\,dx\right)\mu_l = \underbrace{\int_a^b f(x)b_h^k(x)\,dx}_{:=\varphi_k}\ ,\quad k=1,\ldots,N\,.$$

$$\Updownarrow$$

$\boxed{\mathbf{A}\vec{\mu}=\vec{\varphi}}$ with $(\mathbf{A})_{kl} := \int_a^b \frac{db_h^l}{dx}(x)\frac{db_h^k}{dx}(x)\,dx,\quad k,l=1,\ldots,N$,

$\vec{\mu}=[\mu_l]_{l=1}^N \in \mathbb{R}^N$, $\vec{\varphi}=[\varphi_k]_{k=1}^N \in \mathbb{R}^N$.

A linear system of equations!

▷ Galerkin matrix $\mathbf{A}\in\mathbb{R}^{N,N}$, $(\mathbf{A})_{ij} := \int_a^b \frac{db_h^j}{dx}(x)\frac{db_h^i}{dx}(x)\,dx,\quad 1\le i,j\le N$

piecewise derivatives

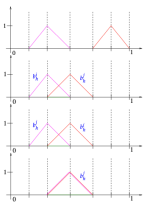
▷ r.h.s. vector $\vec{\varphi}\in\mathbb{R}^N$, $(\vec{\varphi})_k := \int_a^b f(x)b_h^k(x)\,dx,\quad k=1,\ldots,N$.

In addition, we use that the gradients of the tent functions are piecewise constant:

$$\frac{db_h^j}{dx}(x) = \begin{cases} \frac{1}{h_j} & \text{, if } x_{j-1}\le x\le x_j\,,\\ -\frac{1}{h_{j+1}} & \text{, if } x_j < x\le x_{j+1}\,,\\ 0 & \text{elsewhere,}\end{cases} \tag{2.3.2.6}$$

and immediately get

$$\int_a^b \frac{db_h^j}{dx}(x)\frac{db_h^i}{dx}(x)\,dx = \begin{cases} 0 & \text{, if } |i-j|\ge 2 \ \to\\ -\frac{1}{h_{i+1}} & \text{, if } j=i+1 \ \to\\ -\frac{1}{h_i} & \text{, if } j=i-1 \ \to\\ \frac{1}{h_i}+\frac{1}{h_{i+1}} & \text{, if } 1\le i=j\le M-1 \ \to\end{cases}$$



We obtain the following Galerkin matrix, which is symmetric, positive definite (→ Def. 0.3.1.16), and tridiagonal:

$$\mathbf{A} = \begin{bmatrix} \frac{1}{h_1}+\frac{1}{h_2} & -\frac{1}{h_2} & 0 & & 0\\ -\frac{1}{h_2} & \frac{1}{h_2}+\frac{1}{h_3} & -\frac{1}{h_3} & & \\ 0 & \ddots & \ddots & \ddots & \\ & & \ddots & & 0\\ & \ddots & \ddots & & -\frac{1}{h_{M-1}}\\ 0 & & -\frac{1}{h_{M-1}} & \frac{1}{h_{M-1}}+\frac{1}{h_M} & \end{bmatrix} \in \mathbb{R}^{N,N},\quad N:=M-1\,. \tag{2.3.3.2}$$

✍ notation: $h_j := |x_j - x_{j-1}|$ ≙ local meshwidth, cell size ⌟

- S.p.d. tradiagonal matrix.
- Special case: equidistant mesh:

$$= \frac{1}{h}\begin{bmatrix} 2 & -1 & 0 & & 0\\ -1 & 2 & -1 & & \\ 0 & \ddots & \ddots & \ddots & \\ & & \ddots & \ddots & 0\\ & & & -1 & 2 & -1\\ 0 & & & 0 & -1 & 2 \end{bmatrix}\begin{bmatrix}\mu_1\\ \vdots\\ \mu_N\end{bmatrix} = h\begin{bmatrix} f(x_1)\\ \vdots\\ f(x_N)\end{bmatrix}$$

  - Here, we refer it as 1D poisson matrix.

Computation of right hand side vetor

Due to the reason that the basis function is analytic, we can calculate its integral. But for procedural form f, we need to use numerical quadrature to calculate its integral. BUT, if the coefficient function are aslo procedural form, we also need to use the numerical quadrature for the left-hand-side integral.

Replace the integral with an m-point quadrature formula/quadrature rule on $[a,b]$, $m\in\mathbb{N}$ → [Hip19, ??]:

$$\int_a^b \psi(t)\,dt \approx Q_m(\psi) := \sum_{j=1}^m \omega_j^m \psi(\zeta_j^m)\,. \tag{2.3.3.7}$$

ω_j^m: quadrature weights , ζ_j^m: quadrature nodes $\in[a,b]$.

总结来说，下面用数值积分，同时也是neural operator学习的对象：考虑使用$\mathcal{M}$-based composite quadrature rule, which applies a simple quadrature formula on all cells of the mesh, e.g., the **composite trapezoidal rule**:

$$\int_a^b \psi(t)\,\mathrm{d}t \approx \sum_{l=1}^{M} \tfrac{1}{2} h_l (\psi(x_{l-1}) + \psi(x_l)) \,.$$

- Right hand side: source function
- Coefficient function:

$$\varphi_k := (\vec{\varphi})_k = \int_a^b f(x) b_h^k(x)\,\mathrm{d}x \approx \tfrac{1}{2}(h_k + h_{k+1}) f(x_k), \quad 1 \le k \le N \,.$$

Computation for general stiffness coefficient

$$u \in H_0^1(]a,b[): \quad \int_a^b \sigma(x) \frac{du}{dx}(x) \frac{dv}{dx}(x)\,\mathrm{d}x = \int_a^b f(x) v(x)\,\mathrm{d}x \quad \forall v \in H_0^1(\Omega)]a,b[\,.$$

Assumption 2.3.3.11. Smoothness requirement for stiffness coefficient

σ is piecewise continuous, $\sigma \in C^0_{\mathrm{pw}}([a,b])$, with jumps *only* at grid nodes x_j

$$\int_a^b \psi(x)\,\mathrm{d}x \approx \sum_{j=1}^{M} h_j \psi(m_j) \,, \quad m_j := \tfrac{1}{2}(x_j + x_{j-1}) \,.$$

$$\underbrace{\sum_{\ell=1}^{N} \left(\sum_{j=1}^{M} h_j \sigma(m_j) \frac{db_h^\ell}{dx}(m_j) \frac{db_h^k}{dx}(m_j) \right)}_{=(\mathbf{A})_{k,\ell}} \mu_\ell = \underbrace{\tfrac{1}{2}(h_{k+1} + h_k) f(x_k)}_{=:(\vec{\varphi})_k} \,, \quad k = 1, \ldots, N \,,$$

$$\Updownarrow$$

$$\mathbf{A}\vec{\mu} = \vec{\varphi} \,.$$

$$\frac{1}{h} \begin{bmatrix} \sigma_1 + \sigma_2 & -\sigma_2 & 0 & \cdots & & & \cdots & 0 \\ -\sigma_2 & \sigma_2 + \sigma_3 & -\sigma_3 & & \vdots & & & \\ 0 & \ddots & \ddots & \ddots & & & & \\ \vdots & & & & & & & \vdots \\ & & & & \ddots & \ddots & \ddots & 0 \\ \vdots & & & & & -\sigma_{M-2} & \sigma_{M-2} + \sigma_{M-1} & -\sigma_{M-1} \\ 0 & \cdots & & & \cdots & 0 & -\sigma_{M-1} & \sigma_{M-1} + \sigma_M \end{bmatrix} \begin{bmatrix} \mu_1 \\ \vdots \\ \mu_N \end{bmatrix} = h \begin{bmatrix} f(x_1) \\ \vdots \\ f(x_N) \end{bmatrix} \,, \tag{2.3.3.14}$$

with $\sigma_j := \sigma(m_j), j = 1, \ldots, m.$ ⌟

注意，我们没有从节点插值的概念去理解我们的basis function，而是通过有限维空间的角度引入，然后定义了tent function。而tent function的本质也就是用节点处的值去插值单元内部的值。最后求解出来的tent function前的那些系数其实各个节点上的值。

3.4 Case Study: Triangular Linear FEM in Two Dimensions

2D Meshes: Triangulartions

Linear Finite Element Space

Nodal Basis Functions

Galerkin Matrix

本章节的重点是求Galerkin matrix。而根据定义，Galerkin matrix是nodal basis function的bilinear form。然后我们写出定义后发现$A_{i,j}$会由几部分组成。当$i \neq j$时，是i,j连线处左右两侧的单元有贡献。当$i = j$时是以i为节点的所有单元都有贡献。为了便于计算，我们提出了分单元求贡献，然后再组装成Galerkin Matrix的方法。也就是在每个单元里面，我们定义了barcentric function（其实就是Nodal basis function在各个单元中的部分表示）。然后我们求三个node的barcertric function的bilinear form，然后再根据我们提出的一系列组装规则，带回到原Galerkin matrix即可（其实这里也可以把一个单元看成一个小问题，barcentric function其实相当于这个问题的nodal function了，我们求解这个3维小的问题的galerkin matrix）。其中，我们发现barcetric function可以用两种方法得到，分别是几何的和代数的。对于assembly，我们也提出了代码上高效且可行的方案。对于right hand vector，也是先按照基本的方法把单元矩阵组装好，再求linear form的时候，采用数值积分的方式去求即可。

Week 3

3.5 Building Blocks of General Finite Element Methods

Overview: *key features and components* that distinguish the finite element approach to the discretization of linear boundary value problems for partial differential equations:

- variational formulations of a boundary value problem as starting point
- a partitioning of the computational domain Ω by means of a mesh $\mathcal{M}$
- the use of Galerin trial and test spaces based on piecewise polynomials w.r.t. $\mathcal{M}$ 构建我们的Galerkin discretization space，这个space是piecewise continuatity。这个地方需要考虑你决定采用多少的阶数去近似，从而会引出我们的基函数的形式。这个地方基函数我们通常选择以节点的cardinal basis function。⚠️：这里所指的polynomials指的仅仅只是在单元内部是一个polynomials，然后在单元边界上不同polynomials保持连续。此外，在单元内部，polynomials可以看作是以节点进行插值。
  - 节点上的nodal basis function，系数其实就可以看作节点上
- the use of locally supported basis functions for the assembly of the resulting linear system of equations - 在一个单元内部具有单元的型函数。

3.5.1 Meshes

concepts:

Entities refer to the geometric entities “vertex”, ”edge”, ”face” of polygon/polyhedron: meaning of these terms corresponds to geometric intuition.

Entities can be classified by their dimen- sion or co-dimension, which add up to the world dimension/physical dimension d:

mesh entity	dimension	codimension
2D, $d = 2$:		
triangles	2	0
edges	1	1
vertices	0	2
3D, $d = 3$:		
tetrahedra	3	0
faces	2	1
edges	1	2
vertices	0	3

⚠ Non-hanging nodes

3.5.2 Polynomials

The gist of FEMs is approximation by piecewise polynomials.

In FEM:	Galerkin trial/test spaces comprise $\mathcal{M}$-*locally polynomial* functions on Ω

- **uni-variate polynomials**: $\mathcal{P}_p(\mathbb{R}) := \{x \mapsto c_0 + c_1 x + c_2 x^2 + \dots c_p x^p\}$
- **Multivariate Polynomials**:

Definition 2.5.2.2. Multivariate polynomials

Space of multivariate (d-variate) polynomials of (total) degree $p \in \mathbb{N}_0$:

$$\mathcal{P}_p(\mathbb{R}^d) := \{\boldsymbol{x} \in \mathbb{R}^d \mapsto \sum_{\boldsymbol{\alpha} \in \mathbb{N}_0^d, |\boldsymbol{\alpha}| \le p} c_{\boldsymbol{\alpha}} \boldsymbol{x}^{\boldsymbol{\alpha}},\ c_{\boldsymbol{\alpha}} \in \mathbb{R}\}\,.$$

  - $\boldsymbol{\alpha} = (\alpha_1, \dots, \alpha_d)$: $\boldsymbol{x}^{\boldsymbol{\alpha}} := x_1^{\alpha_1} \cdot \dots \cdot x_d^{\alpha_d}$, $|\boldsymbol{\alpha}| = \alpha_1 + \alpha_2 + \dots + \alpha_d$.
  - $d = 2$: $\mathcal{P}_p(\mathbb{R}^2) = \left\{ \sum_{\substack{\alpha_1, \alpha_2 \ge 0 \\ \alpha_1 + \alpha_2 \le p}} c_{\alpha_1, \alpha_2} x_1^{\alpha_1} x_2^{\alpha_2},\ c_{\alpha_1, \alpha_2} \in \mathbb{R} \right\}$
  - Examples: $\mathcal{P}_2(\mathbb{R}^2) = \mathrm{Span}\{1, x_1, x_2, x_1^2, x_2^2, x_1 x_2\}$, $\mathcal{P}_1(\mathbb{R}^2)$ = affine linear functions $\mathbb{R}^2 \mapsto \mathbb{R}$, see Section 2.4.2
  - **Lemma 2.5.2.5. Dimension of spaces of polynomials**

    $$\dim \mathcal{P}_p(\mathbb{R}^d) = \binom{d+p}{p} \quad \textit{for all } p \in \mathbb{N}_0, d \in \mathbb{N}$$

    - $= C_{d+p}^p$
- **Tensor product polynomials**: another way to extend uni-variate polynomials to higher dimensions.

Definition 2.5.2.7. Tensor product polynomials

Space of tensor product polynomials of degree $p \in \mathbb{N}$ in each coordinate direction

$$\mathcal{Q}_p(\mathbb{R}^d) := \left\{ \boldsymbol{x} \mapsto \sum_{\ell_1=0}^{p} \cdots \sum_{\ell_d=0}^{p} c_{\ell_1,\dots,\ell_d} x_1^{\ell_1} \cdot \dots \cdot x_d^{\ell_d},\ c_{\ell_1,\dots,\ell_d} \in \mathbb{R} \right\}$$
$$= \mathrm{Span}\{\boldsymbol{x} \mapsto p_1(x_1) \cdot \dots \cdot p_d(x_d),\ p_i \in \mathcal{P}_p(\mathbb{R}), i = 1, \dots, d\}\,.$$

  - Example: $\mathcal{Q}_2(\mathbb{R}^2) = \mathrm{Span}\{1, x_1, x_2, x_1 x_2, x_1^2, x_1^2 x_2, x_1^2 x_2^2, x_1 x_2^2, x_2^2\}$

Lemma 2.5.2.8. Dimension of spaces of tensor product polynomials

$$\dim \mathcal{Q}_p(\mathbb{R}^d) = (p+1)^d \quad \textit{for all} \quad p \in \mathbb{N}_0, d \in \mathbb{N}$$

⌟

Terminology: $\mathcal{P}_p(\mathbb{R}^d)/\mathcal{Q}_p(\mathbb{R}^d)$ = *complete* spaces of polynomials/tensor product polynomials

  - 对于tensor product polynomials，虽然degree2。但是最高阶会出现四阶，

- 可以看到，随着问题维度的增加，Dimension of space是呈现指数级上升的，这一点和有限差分比较像。这一点是为了说明存在curse of dimensionality.

这两种polynomial就是polynomial的两种不同引入方式，只不过表述不一样，从而导致引入后系统的维度不同。

3.5.3 Basis Functions

Third main ingredient of FEM: locally supported basis functions
(see Section 2.2 for role of bases in Galerkin discretization)

Basis functions $b_h^1, \ldots, b_h^N$ for a finite element trial/test space $V_{0,h}$ built on a mesh $\mathcal{M}$ *must* satisfy:

($\mathfrak{B}_1$) $\mathfrak{B}_h := \{b_h^1, \ldots, b_h^N\}$ is basis of $V_{0,h}$ $\quad\succ\quad$ $N = \dim V_{0,h}$,
($\mathfrak{B}_2$) each b_h^i is associated with a single geometric entity (cell/edge/face/vertex) of $\mathcal{M}$,
($\mathfrak{B}_3$) $\operatorname{supp}(b_h^i) = \bigcup\{\overline{K}: K \in \mathcal{M}, \boldsymbol{p} \subset \overline{K}\}$, if b_h^i associated with cell/edge/face/vertex $\boldsymbol{p}$.

- The basis functions b_h^i are also called **global shape functions** (GSF)/**global basis functions/degrees of freedom** (DOFs)

Often finite element spaces are directly defined by specifying a set of basis functions:

Mesh $\mathcal{M}$ + global shape functions ➨ complete description of finite element space

Supports of global shape functions on triangular mesh

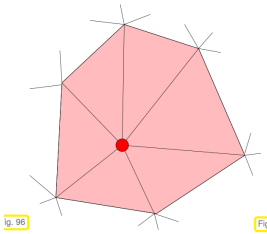


Fig. 96

Support of node-associated basis function, *cf.* Fig. 99

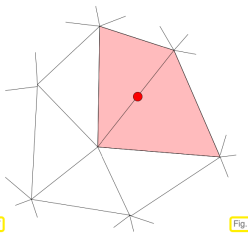


Fig. 97

Support of edge-associated basis function

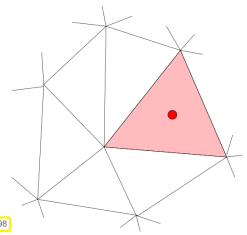


Fig. 98

Support of cell-associated basis function

Local Shape function

Definition 2.5.3.4. Local shape functions (LSF)

Given a finite element function space on a mesh $\mathcal{M}$ with global shape functions b_h^i, $i = 1, \ldots, N$, for every mesh entity K we define

$$\left\{b_K^j\right\}_{j=1}^{Q(K)} := \{b_{h|K}^j,\ K \subset \text{interior of } \operatorname{supp}(b_h^j)\} := \text{set of } \textbf{local shape functions} \text{ (LSF)},$$

that is the local shape functions are the basis functions that *cover* K, restricted to K.

Global shape functions $\xrightarrow{\text{Restriction to entity}}$ local shape functions (2.5.3.5)

Example:

Global basis function for $\mathcal{S}_1^0(\mathcal{M})$ ▷

Example: On the "unit triangle" $\widehat{K}$ with vertices

$$\mathbf{a}^1 = \begin{bmatrix}0\\0\end{bmatrix},\ \mathbf{a}^2 = \begin{bmatrix}1\\0\end{bmatrix},\ \mathbf{a}^3 = \begin{bmatrix}0\\1\end{bmatrix}:$$

we find the local shape functions

$$b_{\widehat{K}}^1(\boldsymbol{x}) := 1 - x_1 - x_2\,,$$
$$b_{\widehat{K}}^2(\boldsymbol{x}) := x_1\,,$$
$$b_{\widehat{K}}^3(\boldsymbol{x}) := x_2\,.$$

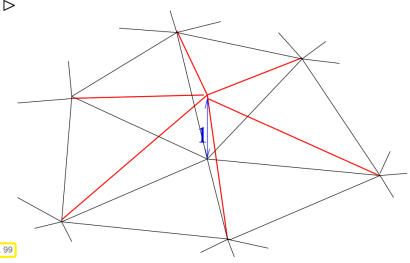


Fig. 99

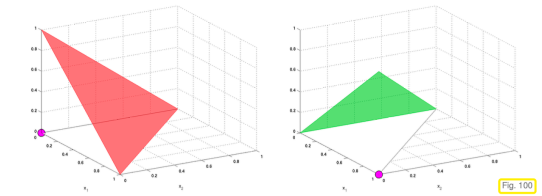


Fig. 100

- 在真实问题中，可以看到shape function其实和节点的值有关，但是我们前面证明过三角形单元的shape function只与角度有关。即使换种思路走，我们也可以把任意的三角形单元转换到unity triangle上面，然后再求单元矩阵。也就是说，至始至终，我只需要转换矩阵（与坐标有关）unity triangle的local shape function即可，就可以考虑进行单元计算。

3.6 Lagrangian Finite Element Methods

In previous chapters, we had only the simple case of $\mathcal{S}_1^0(\mathcal{M})$, piecewise linear finte element functions on simplicial meshes $\mathcal{M}$ as a concrete realization. In this section we will see an important family of finite element spaces contained in $H^1(\Omega)$, the Lagrangian finite element spaces.

这一章节主要就是将前面讨论的情况更general化，提出一类叫做lagrangian finite element spaces。其包含在sobolev空间里面（就是必须得piece-wise连续）。而这个方法就可以帮助我们构建单元内部任意阶数的插值函数。而保证单元与单元在边界上连续的默认前提就是以节点作为interpolate nodes（也就是在节点上得满足interpolation condition）。为了和力学中的概念对应起来，我们要清楚nodal basis function，barcentric function的具体概念。而这里的插值就是拉格朗日基做插值。

Parlance: finite element spaces that are contained in $H^1(\Omega)$, are often called "H^1-conforming".

Notation:
(Lagrangian FE spaces)

$$\mathcal{S}_p^0(\mathcal{M})$$

- continuous functions, *cf.* $C^0(\Omega)$
- locally polynomials of degree p , *e.g.* $\mathcal{P}_p(\mathbb{R}^d)$

Our goal is the construction of finite element spaces and global shape functions of higher polynomials degrees, generalizing the space $\mathcal{S}_1^0(\mathcal{M})$ introduced in the previous section.

3.6.1 Simplicial Lagrangian FEM

Configuration

- Simplicial mesh $\Rightarrow$ triangles in 2D, tetrahedra in 3D.

Generalization

In last chapter, we have already discussed about $\mathcal{S}_1^0(\mathcal{M})/\mathcal{S}_{1,0}^0(\mathcal{M})$. Next, we generalize them to higher polynomial degree $p \in \mathbb{N}_0$.

Definition 2.6.1.1. Simplicial Lagrangian finite element spaces

The space of p-**th degree Lagrangian finite element** functions on simplicial mesh $\mathcal{M}$ is defined as

$$\mathcal{S}_p^0(\mathcal{M}) := \{v \in C^0(\overline{\Omega}): v_{|K} \in \mathcal{P}_p(K) \quad \forall K \in \mathcal{M}\} .$$

- It means that the degree of the locally polynomial function is p, not 1 as we discussed before.

We express the local shape functions in terms of barycentric coordinate functions:

$$
\begin{aligned}
b_K^1 &= (2\lambda_1 - 1)\lambda_1 \,, \\
b_K^2 &= (2\lambda_2 - 1)\lambda_2 \,, \\
b_K^3 &= (2\lambda_3 - 1)\lambda_3 \,, \\
b_K^4 &= 4\lambda_1\lambda_2 \,, \\
b_K^5 &= 4\lambda_2\lambda_3 \,, \\
b_K^6 &= 4\lambda_1\lambda_3 \,.
\end{aligned}
\qquad (2.6.1.6)
$$

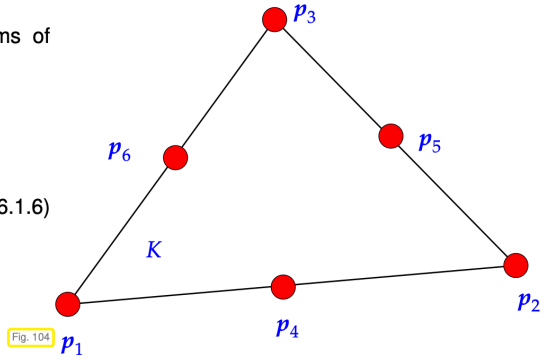
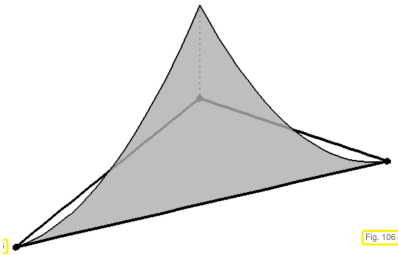


Fig. 104

We remark that it is generally true for Lagrangian finite elements that local shape functions are linear combinations of (products of) barycentric coordinate functions, see (2.7.5.1) below.

- $$b_K^i = \sum_{\alpha\in\mathbb{N}_0^3,|\alpha|=p} \kappa_\alpha \lambda_1^{\alpha_1}\lambda_2^{\alpha_2}\lambda_3^{\alpha_3} \,, \quad \kappa_\alpha \in \mathbb{R} \,, \quad |\alpha| := \alpha_1+\alpha_2+\alpha_3 \,, \qquad (2.7.5.1)$$
- Local shape function 和 barycentric coordinate function之间的关系。

Let us look at the graphs of selected local shape functions for $\mathcal{S}_2^0(\mathcal{M})$ over a triangle:


b_K^1

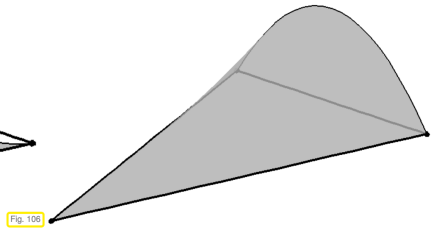
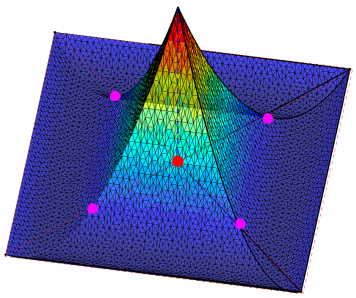


Fig. 106

b_K^6

- own the property of cardinal basis

Global basis function:



◁ Global basis function for $\mathcal{S}_2^0(\mathcal{M})$ associated with a vertex

The cardinal basis property (2.6.1.4) is satisfied, because this function attains value $= 1$ at a vertex (●) and vanishes at the midpoints (●) of the edges of adjacent triangles, as well as at any other vertex.

3.6.2 Tensor-Product Lagrangian FEM

Now we consider tensor product meshes (grids) as follows:

Another special type are tensor product meshes, also called **grids** ▷

in 2D: $a = x_0 < x_1 < \ldots < x_n = b$,
$c = y_0 < y_1 < \ldots < y_m = d$.

$\mathcal{M} = \{]x_{i-1},x_i[\times]y_{j-1},y_j[: \quad (2.5.1.4)$
$1 \le i \le n, 1 \le j \le m\}$.

☞ Restricted to tensor product domains

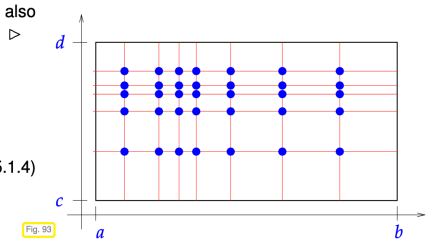
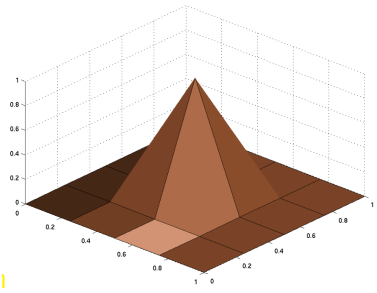


Fig. 93

Terminology: Simplicial mesh ≙ triangular mesh in 2D
tetrahedral mesh in 3D

2D tensor product tent function:



$$b_K^1(\boldsymbol{x}) = (1-x_1)(1-x_2)\,,$$
$$b_K^2(\boldsymbol{x}) = x_1(1-x_2)\,,$$
$$b_K^3(\boldsymbol{x}) = x_1x_2\,,$$
$$b_K^4(\boldsymbol{x}) = (1-x_1)x_2\,.$$

(2.6.2.3)

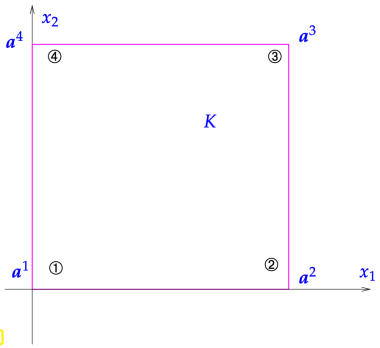


Fig. 115

$\blacktriangleright$ $b_K^i(\boldsymbol{a}^j) = \delta_{ij}$, $1 \le i,j \le 4$,

that is, these basis functions satisfy a local version of the cardinal basis propoerty (2.6.1.4).

- Tensor product construction → bilinear local shape functions, e.g. on $K =]0,1[^2$. 通常来说，local shape function应该和节点坐标有关系，在这里我们考虑一种unity的情况。之所以考虑这种情况也是为了后边为等参单元做铺垫。

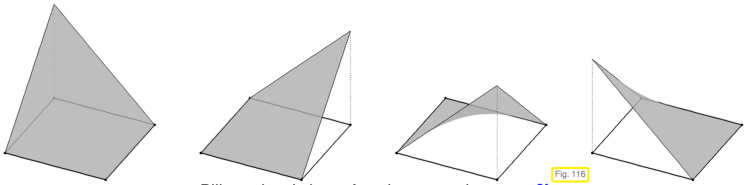


Fig. 116

Bilinear local shape functions on unit square K

$\blacktriangleright$ $\mathrm{Span}\{b_K^1, b_K^2, b_K^3, b_K^4\} = \mathcal{Q}_1(\mathbb{R}^2)$.

Bilinear Lagrangian finite element space on 2D tensor product mesh $\mathcal{M}$:

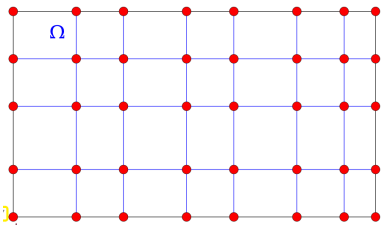
$$\mathcal{S}_1^0(\mathcal{M}) := \{v \in C^0(\Omega)\colon v_{|K} \in \mathcal{Q}_1(\mathbb{R}^2)\ \forall K \in \mathcal{M}\}\,. \qquad (2.6.2.4)$$

- 和2.6.1.1做比较，其实差异就在多项式的定义。这个地方的一阶多项式有四个monimial，而前面那个是只有三个，因此对于一维问题所需要的插值点数目是不一样的。二维的同理。

Definition 2.6.2.5. Tensor product Lagrangian finite element spaces

Space of p-th degree Lagrangian finite element functions on tensor product mesh $\mathcal{M}$

$$\mathcal{S}_p^0(\mathcal{M}) := \{v \in C^0(\overline{\Omega})\colon v_{|K} \in \mathcal{Q}_p(K)\ \forall K \in \mathcal{M}\}\,.$$



◁ — ≙ grid lines of a 2D tensor product mesh

• ≙ interpolation nodes for $\mathcal{S}_1^0(\mathcal{M})$

For $p = 1$ the interpolation nodes agree with the nodes of the mesh.

Note: A function $\in \mathcal{Q}_1(\mathbb{R}^2)$ is uniquely determined by its values in the corners of a rectangle.

- Note: A function $\in \mathcal{Q}_1(\mathbb{R}^2)$ is uniquely determined by its values in the corners of a rectangle. 其实以cardinal basis做基，function就是加权求和。其中各个基前的权就是对应node上的values。

Choice of interpolation nodes for tensor product Lagrangian finite elements of higher degree:

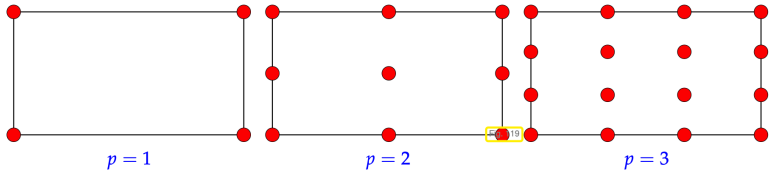


Fig. 119

((Bi)-linear Lagrangian finite elements on hybrid meshes)

$\mathcal{M}$: 2D hybrid mesh comprising triangles & rectangles ▷

Idea: use

- linear functions (→ Def. 2.5.2.2, $p = 1$) on triangular cells,
- bi-linear functions (→ Def. 2.6.2.5, $p = 1$) on rectangles.

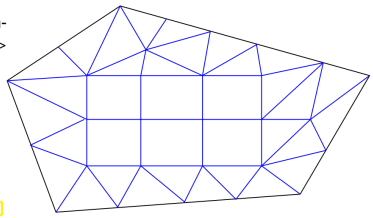


Fig. 120

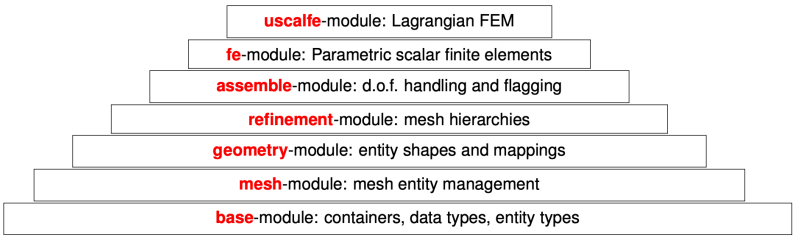
$$\mathcal{S}_1^0(\mathcal{M}) = \left\{ v \in H^1(\Omega) \colon v_{|K} \in \begin{cases} \mathcal{P}_1(\mathbb{R}^2) & \text{, if } K \in \mathcal{M} \text{ is triangle,} \\ \mathcal{Q}_1(\mathbb{R}^2) & \text{, if } K \in \mathcal{M} \text{ is rectangle} \end{cases} \right\}. \tag{2.6.2.11}$$

3.7 Implementation of Finite Element Methods

Guiding principle behind the implementation of finite element codes is:

to rely on *local* computations as much as possible!

Modules of LehrFEM++: according to a hierarchy of dependency

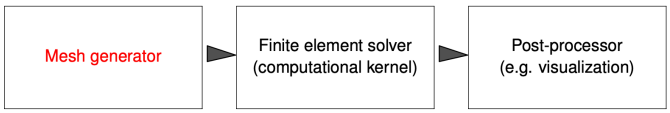


These are supplemented by the

- **io**-module: reading and writing from/to files meshes and finite element functions
- **quad**-module: supplying pre-defined quadrature rules.

3.7.1 Mesh generation

- We use Gmesh for mesh generation, but we won't discuss it here. See the lecture documents.



building blocks of the mesh (nodes, edges, cells)

In LEHRFEM++ mesh objects are to be derived from the interface class `lf::mesh::Mesh`

3.7.2 Mesh Information an Mesh Data Structures

Mesh Data Structures

- Containers: traversal of entities
- Topology: incidence/adjacency
- Geometry: node locations, entity shapes

Where Mesh = sequential container (container) of mesh entity objects (topology, geometry).

3.7.2.1 LF++ Mesh: Container Functionality

From an algorithmic and software point of view:

- a mesh is a collection of (geometric) entities (→ § 2.5.1.2) of particular shapes and location, connected by adjacency/incidence relations
  - An example for an incidence relation (关联关系) on the set of mesh entities is "is sub-entity of".

Purposes of mesh data structures

Mesh data structures must
1. offer unique identification of mesh entities [= cells/(faces)/(edges)/vertices] (for instance, by an integer index)
2. make possible traversal of cells of the mesh (→ global numbering)
3. represent mesh topology (= incidence relationships of cells/faces/edges/vertices)
4. allow sequential access to edges/faces of a cell (→ traversal of local shape functions/degrees of freedom)
5. describe mesh geometry (= location/shape of cells/faces/edges/vertices)

- 总体上来说，一个mesh有极大成员，取一个三维的问题来进行举例：cells - faces - edges - vertices。我们mesh data structure的重要几个成分就是：这些entities的访问，cells（单元）层面的traversal (global numbering)，单元、节点、etc这些之间的连接关系组成mesh topology。在一个单元内部实现edges/faces的traversal。以及mesh的geometry，也就是形状和坐标。

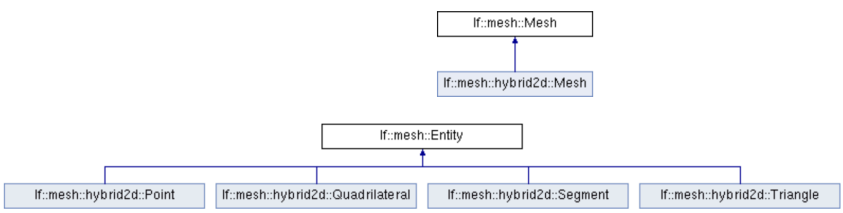
order (global numbering) of the geometric entities is very important

Global objects	↔	*Local objects*
• single instance exists • container for geometric entities		• many instances exist • store geometry/topology

In LEHRFEM++ this classification directly manifests itself in two fundamental interface classes (abstract base classes with purely virtual functions)

global: **lf::mesh::Mesh** → (documentation)	⟷	local: **lf::mesh::Entity** → (documentation)

Inheritance diagram:



Using entity iterators in LehrFEM++

Implement loops over the entities of a **lf::mesh::Mesh** object and prints their indices. These loops are fundamental for almost every task in a finite element code:

C++ code 2.7.2.7: Traversal of entities of a mesh ⇝ GITHUB

```
int traverseEntities(const lf::mesh::Mesh &mesh, dim_t codim) {
  LF_ASSERT_MSG((codim <= mesh.DimMesh()), "codim " << +codim << " too large");
  std::cout << "Mesh dimension = " << mesh.DimMesh()
            << ", iterating over entities of co-dim. " << codim << ": "
            << mesh.NumEntities(codim) << " exist" << std::endl;
  size_type cnt = 0;
  // Typical loop for running through all entities of a specific
     co-dimension
  for (const lf::mesh::Entity *entity : mesh.Entities(codim)) {
    // Print entity information including its unique index
    std::cout << cnt << ": Entity #" << mesh.Index(*entity) << ": " << *entity
              << std::endl;
    cnt++;
  }
  return cnt;
}
```

Note that the loop variable entity is a *pointer* to **lf::mesh::Entity** objects.

- 接着目前，我们看到mesh的功能如下：各个codimension的entities数目，entity的indexing。访问特定entity通过codim和索引index。

3.7.2.2 LF++ Mesh: Topology layer

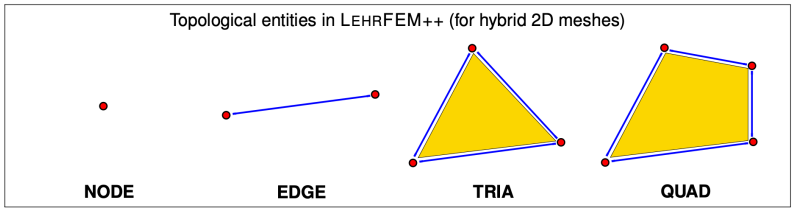
We examine the adjacency/incidence relations restricted to a single entity. In LEHRFEM++ the topology of an entity can be queried by the method **RefEl()** of **lf::mesh::Entity**. It returns a static object of type **lf::base::RefEl** of which there are four different versions:

(I) NODE-type (= `lf::base::RefEl::kPoint()`): 0-dimensional entity without sub-entities,

(II) EDGE-type (= `lf::base::RefEl::kSegment()`): a 1-dimensional entity with two 0-dimensional NODE-type subentities

(III) TRIA-type (= `lf::base::RefEl::kTria()`): a 2-dimensional entity with three sub-entities of EDGE-type and further three sub-entities of NODE-type

(IV) QUAD-type (= `lf::base::RefEl::kQuad()`): another 2-dimensional entity possessing four sub-entities of EDGE-type and NODE-type, respectively.



The following C++ function demonstrates how to test for a particular topological type of an entity in LEHRFEM++:

C++ code 2.7.2.11: Counting cells of topological type QUAD and TRIA ⇝ GITHUB

```
std::pair<size_type, size_type> countCellTypes(const lf::mesh::Mesh &mesh) {
  size_type tria_cnt = 0;
  size_type quad_cnt = 0;  // Counters
  // Loop over all cells (= co-dimension-0 entities) of the mesh
  for (const lf::mesh::Entity* cell : mesh.Entities(0)) {
    // Fetch type information
    lf::base::RefEl ref_el{cell->RefEl()};
    // Test, if current cell is of a particular type
    switch (ref_el) {
      case lf::base::RefEl::kTria(): { tria_cnt++; break; }
      case lf::base::RefEl::kQuad(): { quad_cnt++; break; }
      default: { LF_VERIFY_MSG(false, "Unknown cell type"); }
    }}
  return {tria_cnt, quad_cnt};
}
```

- Incidence relations: 1. (boundary) contains (edge -> triangular) 2. "is part of" (node -> triangular)

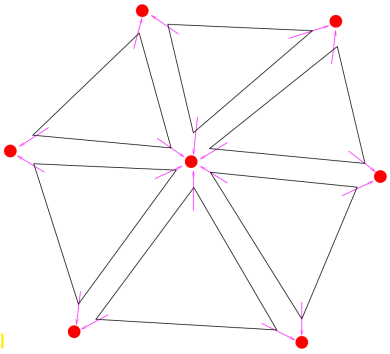
对于每一个entity，有一个‼️的点是local topology concerns the local numbering/local ordering of sub-entities, which underlies the implementation of the fundamental method.

```
nonstd::span<const lf::mesh::Entity* const>
SubEntities(unsigned rel_codim) const
```

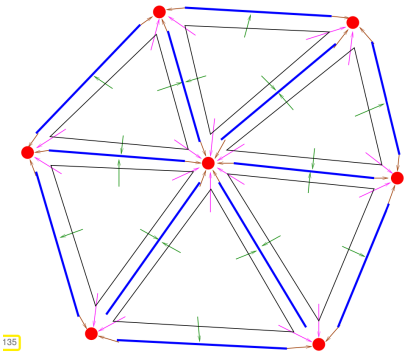
- It returns a random access container, an array, of sub-entities of a particular *relative co-dimension*, which is the dimension gap between the current entity object and the requested sub-entity.
- 注意最后返回的同样是一个指针数组 array of sub-entity pointer.

Storing topology of triangular mesh in 2D

topology有在不同的库中具有不同的存储格式，我自己采用的是第一种minimal scheme。即只存储vertices，不存储edges。



(A) Minimal scheme: triangles hold lists/vectors of references to their vertices. Edges not stored.
Realized in the mesh data structure discussed in Section 2.4.1, see the class **TriaMesh2D** listed in Code 2.4.1.2. This is sufficient for linear Lagrangian finite elements, if no special boundary conditions have to be dealt with.
Note that this scheme already provides complete topological information (edges can be reconstructed)!
This scheme is used by MFEM.



(C) Full unidirectional topology representation:

- All geometric entities are stored as (virtual) "objects".
- Elements hold lists/vectors of references to their vertices and edges.
- Edges have references to their endpoints.

Topology representation in LEHRFEM++

- 还有三种存储形式，见课件P210页。

C++ code 2.7.2.19: Printing topology information for a mesh in LEHRFEM++ ➙ GITHUB

```
void scanTopology(const lf::mesh::Mesh &mesh, dim_t codim) {
  LF_ASSERT_MSG((codim <= mesh.DimMesh()),
          "codim " << +codim << " too large");
  // loop over all entities of the specified codimension
  for (const lf::mesh::Entity* ent : mesh.Entities(codim)) {
    // Fetch topology type (TRIA or QUAD so far)
    const lf::base::RefEl ref_el{ent->RefEl()};
    // Print topological type and global index of the ent
    const glb_idx_t ent_idx = mesh.Index(*ent);
    std::cout << ref_el << ": idx = " << ent_idx << std::endl;
    // Inspect sub-entities of any co-dimension
    for (dim_t sub_codim = 1; sub_codim <= mesh.DimMesh() - codim;
         ++sub_codim) {
      // Obtain iterator over sub-entities
      nonstd::span<const lf::mesh::Entity* const>
     sub_ent_range { ent->SubEntities(sub_codim) };
      size_type sub_cnt = 0;  // Counter for sub-entities
      // Loop over sub-entities, whose types and indices will be output
      for (const lf::mesh::Entity* subent : sub_ent_range) { //
        std::cout << "\t rel. codim " << +sub_codim << " sub-ent "
            << sub_cnt << ": " << *subent << ", idx = "
            << mesh.Index(*subent) << std::endl;
        sub_cnt++;
      }}}}
```

3.7.2.3 LF++Mesh: Geometry Layer

LF++ stores the shape of any entity in objects derived from the interface class **lf::geometry::Geometry**

> In LEHRFEM++ coordinate vectors are small *fixed size* EIGEN *vector types*, see [Hip19, **??**] and Section 2.7.3. Thus, all of EIGEN's linear algebra operations and functions are available for them.

At this point the main method for learning the shape of an entity is the function

```
Eigen::MatrixXd
lf::geometry::Corners(const lf::geometry::Geometry& geo);
```

It returns the Cartesian coordinates of the corner points of a (geometric) entity as the columns of a matrix. For instance, if the shape of the entity is a 2D triangle with vertices $\boldsymbol{a}^1 = \begin{bmatrix} a_1^1 \\ a_2^1 \end{bmatrix}$, $\boldsymbol{a}^2 = \begin{bmatrix} a_1^2 \\ a_2^2 \end{bmatrix}$, and $\boldsymbol{a}^3 := \begin{bmatrix} a_1^3 \\ a_2^3 \end{bmatrix}$, then this matrix reads

$$\begin{bmatrix} a_1^1 & a_1^2 & a_1^3 \\ a_2^1 & a_2^2 & a_2^3 \end{bmatrix} \in \mathbb{R}^{2,3}.$$

Accessing corner coordinate in LF++

C++ code 2.7.2.23: Output of locations of entity corners ➙ GITHUB

```
void PrintGeometryInfo(const lf::mesh::Mesh &mesh, dim_t codim) {
  LF_ASSERT_MSG((codim <= mesh.DimMesh()),
          "codim " << +codim << " too large");
  // loop over all entities of the specified codimension
  for (const lf::mesh::Entity* ent : mesh.Entities(codim)) {
    // Number of nodes = number of corner points
    const size_type num_nodes = ent->RefEl().NumNodes();
    // Obtain pointer to geometry object associated with entity
    const lf::geometry::Geometry *geo_ptr = ent->Geometry();
    LF_ASSERT_MSG(geo_ptr != nullptr, "Missing geometry!");
    // Fetch coordinates of corner points in packed format § 2.7.2.21
    Eigen::MatrixXd corners = lf::geometry::Corners(*geo_ptr);
    LF_ASSERT_MSG(corners.rows() == geo_ptr->DimGlobal(),
                  "dimension mismatch for coordinate vectors");
    LF_ASSERT_MSG(corners.cols() == num_nodes, "#corners mismath");
    std::cout << ent->RefEl() << "(" << mesh.Index(*ent) << ") pts: ";
    for (int l = 0; l < num_nodes; ++l) {
      std::cout << l << " =[" << corners.col(l).transpose() << "], ";
    }
    std::cout << std::endl;
  }}
```

两个点明白就行：

- 先ran through all the entity（通过codimension来决定，也就是单元层面）。用`mesh.Entities(codim)`取出来的每个entity是一个对象。用`lf::mesh::Entity`来进行存放。之后再调用`.RefEI()`来获得topology信息。通过调用`.Geometry()`开获得geometry信息。需要注意的是。这里的entity通常是一个指针。

– 在每个entity里面，实现对相关topology or geometry information的访问。

3.7.3 Assembly Algorithm

首先提出求解localization的概念，也就是Galerkin matrix是integral，localization的概念就是setting integrals into sums of local contributions. 即得到local element matrix。element matrix的大小其实是受到这个单元的order决定的。因为不同的order有不同的节点数，每个节点数又对应单元刚度矩阵中的一行一列。而这过程中，我们要建立映射：local-global index mapping。这个映射建立后可以让我们把gloal shape function转成local shape function。

这里提到了单元节点标号规则。接着讨论实现这种mapping的一个关键元素是d.o.f. Handler类。这个类的初始化和定义，这个类就是让我们把全局shape function映射到单元上，之后我们在单元上做local operation，之后再反映射会全局对应节点。在全局上进行组装。

Review

Because the order of the local polynomial will give rise to the difference of the total nodes. Let's review the dimension of spaces of polynomials:

Lemma 2.5.2.5. Dimension of spaces of polynomials

$$\dim \mathcal{P}_p(\mathbb{R}^d) = \binom{d+p}{p} \quad \text{for all } p \in \mathbb{N}_0, d \in \mathbb{N}$$

Lemma 2.5.2.8. Dimension of spaces of tensor product polynomials

$$\dim \mathcal{Q}_p(\mathbb{R}^d) = (p+1)^d \quad \text{for all} \quad p \in \mathbb{N}_0, d \in \mathbb{N}$$

Assembly: Index Mappings

☞ Local→global index map (“d.o.f. mapper”/”d.o.f. handler”)

$$\texttt{locglobmap} : \{\text{co-dim.-}m \text{ entities}\} \times \mathbb{N} \to \mathbb{N} ,$$
$$\texttt{locglobmap}(K,i) = j , \quad \text{if} \quad b^j_{h\,|K} = b^i_K , \quad i \in \{1,\ldots,Q(K)\} . \qquad (2.7.4.8)$$

global shape function　　local shape function

The mapping locglobmap generalizes the device of the index mapping array dofh introduced in (2.4.5.21) on Page 178 for linear Lagrangian finite elements on 2D triangular meshes and also used in Code 2.4.5.23: Precisely, they are related by $\texttt{dofh}(k,l) = \texttt{locglobmap}(K,l)$, if K has index k , $l \in \{1,2,3\}$.

Data structure: $\texttt{dofh} \in \mathbb{N}^{\sharp\mathcal{M},3}$: local→global *index mapping array*: “d.o.f. mapper”

$\texttt{dofh}(k,l)$ = global number of vertex l of k-th cell $\in \{1,\ldots,N\}$
$x_{\texttt{dofh}(k,l)} = a^l$ when a^1, a^2, a^3 are the vertices of K_k , (2.4.5.21)

for $l \in \{1,2,3\}, k \in \{1,\ldots,M\}, M := \sharp\mathcal{M}, N := \sharp\mathcal{V}(\mathcal{M})$ (“mathematical indexing”!).

➢ simple realization of index mapping:

```
dofh(k,l) := Mesh._elements(k-1,l-1)
```

mathematical indexing　　C++ indexing

坐标转换的一系列操作（函数）全部由一个对象来管理。这个类是 **lf::assemble::DofHandler**, which suuplies the following main methods:

- `size_type lf::assemble::DofHandler::NumDofs () const;`

which returns the total number of global basis functions, that is, the dimension of the finite element space.

- `size_type lf::assemble::DofHandler::NumLocalDofs( const lf::mesh::Entity & ) const;`

which returns the number of global/local shape function *covering* any geometric entity of the mesh of any co-dimension.

- `nonstd::span<const lf::assemble::gdof_idx_t> lf::assemble::DofHandler::GlobalDofIndices( const lf::mesh::Entity &entity) const;`

which returns an array of index numbers $\in \{0,\ldots,$`NumDofs()`$-1\}$ for the global shape functions *covering* a particular entity of *any* dimension. The convention (LN) for the numbering of local shape functions applies.

- `size_type lf::assemble::DofHandler::NumInteriorDofs( const lf::mesh::Entity & ) const;`

which tells us the number of global/local shape functions *associated with* a particular geometric entity of any co-dimension.

- `nonstd::span<const lf::assemble::gdof_idx_t> lf::assemble::DofHandler::InteriorGlobalDofIndices( const lf::mesh::Entity &entity) const;`

which provides the array of global indices $\in \{0,\ldots,$`NumDofs()`$-1\}$ of the global shape functions *associated with* a specific geometric entity of any co-dimension.

- `const lf::mesh::Entity &lf::assemble::DofHandler::Entity( gdof_idx_t dofnum) const;`

which returns a reference to the unique geometric entity to which the global shape function with index `dofnum` is *associated*.

- 这个对于每个entity的interior dof非常有用，尤其是在计算问题组装后的单元非零元素个素以及整个问题的维度的时候

⚠️: 这里的DofHandler是需要做initialization的

```
lf::assemble::UniformFEDofHandler dof_handler(
mesh_p, {{lf::base::RefEl::kPoint(), ndof_node},
  {lf::base::RefEl::kSegment(), ndof_edge},
  {lf::base::RefEl::kTria(), ndof_tria},
  {lf::base::RefEl::kQuad(), ndof_quad}});
```

where the variables `ndof_*` contain the numbers of global/local shape functions associated with the entity type in front of them. For the 2D finite element spaces $\mathcal{S}_p^0(\mathcal{M})$ the table beside gives the numbers:

	$p=1$	$p=2$	$p=3$
ndof_node	1	1	1
ndof_edge	0	1	2
ndof_tria	0	0	1
ndof_quad	0	1	4

- 用这个interior node的概念我们可以很快计算出单元的local shape function的数量。也可以很方便计算出non-zero entities的数量。
- Example:

  The **lf::assemble::DofHandler** studied in this example was initialized as follows:

  ```
  lf::assemble::UniformFEDofHandler dof_handler(
  mesh_p, {{lf::base::RefEl::kPoint(), 1},
    {lf::base::RefEl::kSegment(), 2},
    {lf::base::RefEl::kTria(), 1},
    {lf::base::RefEl::kQuad(), 4}});
  ```

  This means that every NODE carries one d.o.f., every EDGE two of them, every TRIA 1, and there are four global shape functions associated to every QUAD. Thus we compute

  ```
  (dof_handler.NumDofs() == 22) == true;
  ```

```
QUAD 0: 16 dofs = [0 1 3 2 5 6 9 10 13 14 7 8 17 18 19 20]
int = [17 18 19 20]
TRIA 1: 10 dofs = [1 3 4 9 10 15 16 11 12 21 ] int = [21]
EDGE 0: 4 dofs = [0 1 5 6 ] int = [5 6]
EDGE 1: 4 dofs = [2 0 7 8 ] int = [7 8]
EDGE 2: 4 dofs = [1 3 9 10 ] int = [9 10]
EDGE 3: 4 dofs = [4 1 11 12 ] int = [11 12]
EDGE 4: 4 dofs = [3 2 13 14 ] int = [13 14]
EDGE 5: 4 dofs = [3 4 15 16 ] int = [15 16]
NODE 0: 1 dofs = [0 ] int = [0]
NODE 1: 1 dofs = [1 ] int = [1]
NODE 2: 1 dofs = [2 ] int = [2]
NODE 3: 1 dofs = [3 ] int = [3]
NODE 4: 1 dofs = [4 ] int = [4]
```

Distribute Assembly Schemes

- We often use Cell-oriented assembly of finite element Galerkin matrix and right hand side vector, which relys on:

  — loops only over mesh cells combined with purely local operations.

Pseudocode 2.7.4.20: Abstract assembly routine for finite element Galerkin matrices

```
Sparse Matrix ← assembleGalMat(Mesh M) {
  A = N × N sparse matrix; // Allocated zero sparse matrix
  foreach K ∈ M { // loop over all cells
    Qk  = no_loc_shape_functions(K);
    // Local operation:  compute Qk × Qk element matrix → Def. 2.7.4.5,
    // usually incurs cost of only "O(1)"
    Ak = getElementMatrix(K);
    // Get vector of global indices (length Qk);
    // Usage of locglobmap as in Ex. 2.7.4.10
    Vector idx = {locglobmap(K,1),...,locglobmap(K,Qk)};
    // Add local contributions to global matrix
    for i :=1 to Qk {
      for j :=1 to Qk {
        // Update entry of FE Galerkin matrix
        A(idx(i),idx(j)) += Ak(i,j);
      }}
  } // end main loop
  return(A);
}
```

LF++ Examples for assembly

C++ code 2.7.4.23: Assembly function of LehrFEM++ ⇝ GITHUB

```
template <typename TMPMATRIX, class ENTITY_MATRIX_PROVIDER>
void AssembleMatrixLocally(dim_t codim, const DofHandler &dof_handler_trial,
const DofHandler &dof_handler_test,
ENTITY_MATRIX_PROVIDER &entity_matrix_provider,
TMPMATRIX &matrix) {
  // Fetch pointer to underlying mesh
  auto mesh = dof_handler_trial.Mesh();
  // Central assembly loop over entities of co-dimension specified by
  // the function argument codim
  for (const lf::mesh::Entity *entity : mesh->Entities(codim)) {
    // Some entities may be skipped
    if (entity_matrix_provider.isActive(*entity)) {
      // Size, aka number of rows and columns, of element matrix
      const size_type nrows_loc = dof_handler_test.NumLocalDofs(*entity);
      const size_type ncols_loc = dof_handler_trial.NumLocalDofs(*entity);
      // row indices of for contributions of cells
      nonstd::span<const gdof_idx_t> row_idx(
      dof_handler_test.GlobalDofIndices(*entity));
      // Column indices of for contributions of cells
      nonstd::span<const gdof_idx_t> col_idx(
      dof_handler_trial.GlobalDofIndices(*entity));
      // Request local matrix from ENTITY_MATRIX_PROVIDER object.  In the
      // case codim = 0, when entity is a cell,
      // this is the element matrix
      const auto elem_mat{entity_matrix_provider.Eval(*entity)}; //
      // Assembly double loop over element matrix
      for (int i = 0; i < nrows_loc; i++) {
        for (int j = 0; j < ncols_loc; j++) {
          // Add the element at position (i,j) of the local matrix
          // to the entry at (row_idx[i], col_idx[j]) of the global
             matrix
          matrix.AddToEntry(row_idx[i], col_idx[j], elem_mat(i, j));
        }}
```

- `codim` is used for boundary integral
- Variational problems can be with different trial and test spaces. This is the reason why we use two `doh_handler` object. See the page P226 for details of the method.
- Assembly schemes:

- Loops over entities
- DofHandler queries
- Local operation

3.7.4 Local Computations

这一章的核心是计算element matrix。而element matrix的计算指望于local shape function的积分。第一种是可以给出解析解的analytic computation。第二种是像激励函数f或者系数函数μ是prodecural form的时候，我们采用数值积分。然后介绍了数值积分的时候，我们需要将单元转换到标准单元中。这涉及到一个转换矩阵（affline function）。通常是jacobian和其det()。然后再进行积分。

First Option: Direct analytic evaluations - closed form expressions

- Only for locally constant coefficients/data

两种思路来引入，第一种是，我们的目的是计算element matrix。然后我们知道element matrix中就是计算两个basis function的bilinear form。接着可以将积分面积Ω展开成单元K内积分,从而我们就引出了barycentric coordinate function的概念，从而引出了element matrix的概念。而最终的计算就是单元矩阵对Galerkin Matrix的组装。

Notation:

- Basis:
  - 

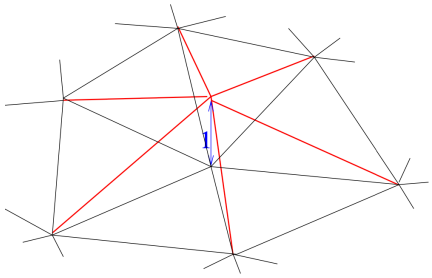
    Idea: ✦ choose (ordered) basis $\mathfrak{B}_h = \{b_h^1, \ldots, b_h^N\}$, $N := \dim V_{0,h}$, of $V_{0,h}$:

    $$V_{0,h} = \operatorname{Span}\{\mathfrak{B}_h\}$$

    ✦ insert basis representation into the variational equation (2.2.1.1)

    $$v_h \in V_{0,h} \;\Rightarrow\; v_h = \nu_1 b_h^1 + \cdots + \nu_N b_h^N\,, \quad \nu_i \in \mathbb{R}\,, \qquad (2.2.2.1)$$
    $$u_h \in V_{0,h} \;\Rightarrow\; u_h = \mu_1 b_h^1 + \cdots + \mu_N b_h^N\,, \quad \mu_i \in \mathbb{R}\,. \qquad (2.2.2.2)$$

  - Usually, we opt to use nodal basis of $\mathcal{S}_1^0(\mathcal{M})$:
    - $$b_h^i \in \mathcal{S}_1^0(\mathcal{M})\,,$$
      $$b_h^i(\boldsymbol{x}_j) = \begin{cases} 1 & \text{, if } i = j\,, \\ 0 & \text{else,} \end{cases}$$
      $$i,j \in \{1,\ldots,N\}\,.$$

    - Note that the defining relations amount to the **cardinal basis** property of $\mathcal{B}_h$ with respect to the node set $\mathcal{V}(\mathcal{M}) = \{\boldsymbol{x}_1, \ldots, \boldsymbol{x}_N\}$. basis expansion coefficients → nodal values
      - $u_h \in \mathcal{S}_1^0(\mathcal{M})$: $u_h = \sum_{i=1}^N \mu_i b_h^i \;\Leftrightarrow\; \mu_i = u_h(\boldsymbol{x}_i) \quad \forall i = 1,\ldots,N\,.$
- Galerkin matrices: $\mathbf{A} = \left[\mathsf{a}(b_h^k, b_h^j)\right]_{j,k=1}^N \in \mathbb{R}^{N,N}\,,$ $j \hat{=}$ row index, $k \hat{=}$ column index,
  - We learn about that Galerkin matrix is a sparse matrix. For sake of simplicity we consider the bilinear form for the Poisson equation: $(\mathbf{A})_{i,j} = \mathsf{a}(b_h^j, b_h^i) = \int_\Omega \mathbf{grad}\, b_h^j \cdot \mathbf{grad}\, b_h^i \,\mathrm{d}\boldsymbol{x}\,.$

- 然后我们就提出了在单元内部进行计算，最后组装的概念

Idea:

"Assembly"
(add up *cell contributions*)

$$(\mathbf{A})_{ij} = \int_{K_1} \mathbf{grad}\, b^j_{h|K_1} \cdot \mathbf{grad}\, b^i_{h|K_1}\, \mathrm{d}\boldsymbol{x} + \int_{K_2} \mathbf{grad}\, b^j_{h|K_2} \cdot \mathbf{grad}\, b^i_{h|K_2}\, \mathrm{d}\boldsymbol{x}$$

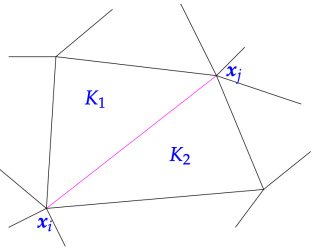


Fig. 73

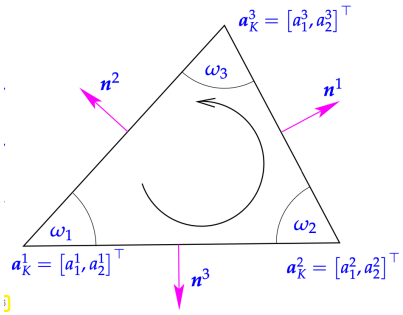
- Barycentric coordinate functions:
  - $\mathsf{a}_K(b^j_h, b^i_h) := \int_K \mathbf{grad}\, b^j_{h|K} \cdot \mathbf{grad}\, b^i_{h|K}\, \mathrm{d}\boldsymbol{x}$, x_i, x_j nodes $\in$ vertices of K .

  §2.4.5.2 (Barycentric coordinate functions) We aim to find analytic formulas for the restrictions $b^i_{h|K}$. If $\boldsymbol{a}^1_K, \boldsymbol{a}^2_K, \boldsymbol{a}^3_K$ are the vertices of the triangle K with coordinates $\boldsymbol{a}^1_K = \begin{bmatrix} a^1_1 \\ a^1_2 \end{bmatrix}$, $\boldsymbol{a}^2_K = \begin{bmatrix} a^2_1 \\ a^2_2 \end{bmatrix}$, and $\boldsymbol{a}^3_K = \begin{bmatrix} a^3_1 \\ a^3_2 \end{bmatrix}$,
  - we write

$$\lambda_i := b^j_{h|K} \quad \text{with} \quad \boldsymbol{a}^i_K = \boldsymbol{x}_j \qquad \left[\begin{array}{l} i \leftrightarrow \text{local vertex number} \\ j \leftrightarrow \text{global node number} \end{array} \right] \qquad (2.4.5.3)$$

  - The barycentric coordinate functions owe their name to the fact that they can be regarded as "coordinates of a point with respect to the vertices of a triangle" in the sense that: $\boldsymbol{x} = \lambda_1(\boldsymbol{x})\boldsymbol{a}^1_K + \lambda_2(\boldsymbol{x})\boldsymbol{a}^2_K + \lambda_3(\boldsymbol{x})\boldsymbol{a}^3_K$.

- Element stiffness matrix: $\mathbf{A}_K = \left[\int_K \mathbf{grad}\, \lambda_i \cdot \mathbf{grad}\, \lambda_j\, \mathrm{d}\boldsymbol{x} \right]_{i,j=1}^{3} \in \mathbb{R}^{3,3}$.
  -



$$\mathbf{A}_K = \frac{1}{2} \begin{bmatrix} \cot\omega_3 + \cot\omega_2 & -\cot\omega_3 & -\cot\omega_2 \\ -\cot\omega_3 & \cot\omega_3 + \cot\omega_1 & -\cot\omega_1 \\ -\cot\omega_2 & -\cot\omega_1 & \cot\omega_2 + \cot\omega_1 \end{bmatrix} .$$

  - 对于三角形单元，local computation非常容易，似乎local computation只与三个角的角度有关，而与三个单元节点的坐标无关。
  - 引入local computation的缘由是我们定义了galerkin discretization space之后，我们引入nobal basis function.然后用其计算galerkin matrix。然后通过研究那个积分我们发现为了高效计算，我们可以现在单元内部计算，之后再整体上组装。这个过程中我们引入了barycentric coordinate function来帮助我们进行计算。

- **Mesh/grid**: $\mathcal{M} := \{]x_{j-1}, x_j[: 1 \le j \le M\}$.
- Galerkin discretization space: $\mathcal{S}^0_{1,0}(\mathcal{M})$ 0指的是在整个离散的空间上，解是continuous C^0，1指的是在每个单元上polynomial order是1，也就是linear。0指的是边界条件是0

barycentric coordinate representations of local shape functions, in 2D: $b^i_K = \sum_{\boldsymbol{\alpha} \in \mathbb{N}^3_0, |\boldsymbol{\alpha}| = p} \kappa_{\boldsymbol{\alpha}} \lambda_1^{\alpha_1} \lambda_2^{\alpha_2} \lambda_3^{\alpha_3}$, $\kappa_{\boldsymbol{\alpha}} \in \mathbb{R}$, $|\boldsymbol{\alpha}| := \alpha_1 + \alpha_2 + \alpha_3$

Lemma 2.7.5.5. Integration of powers of barycentric coordinate functions

For any non-degenerate d-simplex K with barycentric coordinate functions $\lambda_1, \ldots, \lambda_{d+1}$ and exponents $\alpha_j \in \mathbb{N}$, $j = 1, \ldots, d+1$,

$$\int_K \lambda_1^{\alpha_1} \cdot \cdots \cdot \lambda_{d+1}^{\alpha_{d+1}} \, \mathrm{d}\boldsymbol{x} = d!|K| \, \frac{\alpha_1! \alpha_2! \cdot \cdots \cdot \alpha_{d+1}!}{(\alpha_1 + \alpha_2 + \cdots + \alpha_{d+1} + d)!} \quad \forall \boldsymbol{\alpha} \in \mathbb{N}_0^{d+1} \, . \tag{2.7.5.6}$$

Local Quadrature

necessary for coefficients/data in procedural form

C++ code 2.7.5.41: Entity-based composite numerical quadrature ➙ GITHUB

```
template <typename FUNCTOR>
auto localQuadFunction(
    const lf::mesh::Mesh &mesh,
    std::map<lf::base::RefEl, lf::quad::QuadRule> quadrules, FUNCTOR &&f,
    dim_t codim,
    std::function<bool(const lf::mesh::Entity &)> pred =
        [](const lf::mesh::Entity & /*entity*/) -> bool { return true; }) {
  LF_ASSERT_MSG(mesh.DimMesh() >= codim, "Illegal codim = " << codim);
  // Variable for summing the result
  using value_t = std::invoke_result_t<FUNCTOR, Eigen::VectorXd>;
  value_t sum_var{};
  // Loop over entities of co-dimension codim
  for (const lf::mesh::Entity *entity : mesh.Entities(codim)) {
    // Obtain geometry information for entity
    const lf::geometry::Geometry &geo{*entity->Geometry()};
    // obtain quadrature rule suitable for entity type
    auto tmp = quadrules.find(entity->RefEl());
    if (tmp != quadrules.end()) {
      // A quadrature rule has been found
      const lf::quad::QuadRule &qr{tmp->second};
      // Number of quadrature points
      const size_type P = qr.NumPoints();
      // Quadrature points
      const Eigen::MatrixXd zeta_ref{qr.Points()};
      // Map quadrature points to physical/world coordinates
      const Eigen::MatrixXd zeta{geo.Global(zeta_ref)};
      // Quadrature weights
      const Eigen::VectorXd w_ref{qr.Weights()};
      // Gramian determinants
      const Eigen::VectorXd gram_dets{geo.IntegrationElement(zeta_ref)};
      // Iterate over the quadrature points
      for (int l = 0; l < P; ++l) {
        sum_var += w_ref[l] * f(zeta.col(l)) * gram_dets[l];
      }
    } else {
      LF_VERIFY_MSG(false, "Missing quadrature rule for " << entity->RefEl());
    }
  }
  return sum_var;
}
```

3.7.5 Treatment of Essential Boundary Conditions

这一章介绍的就是我们如何处理essential (Dirichlet BVPs) boundary condition。最后其实是通过修正Galerkin Matrix和r.h.s vector。然后再求解这个LSE即可得到结果。

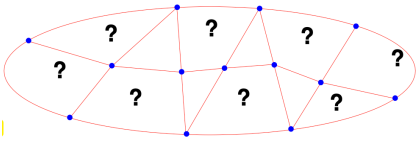
3.8 Parametric Finite Element Methods

等参单元的概念。tranformations to reference cells, which enable us to extend the range of Lagrangian finite element spaces signigficantly.

Φ^* maps {functions on Ω} $\to$ {functions on $\hat{\Omega}$}

- A linear mapping between function spaces

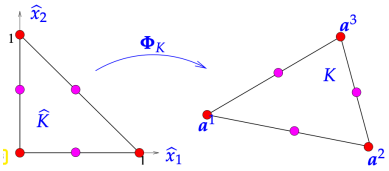
Motivation:



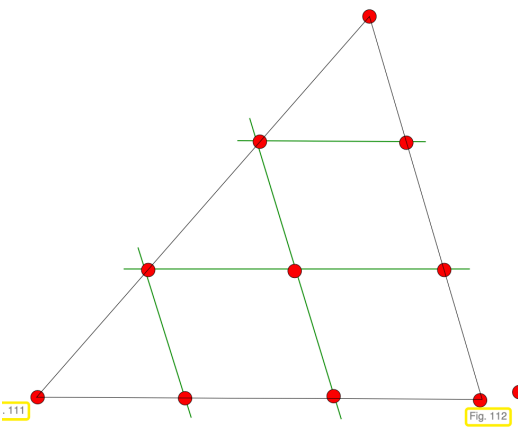
◁ 2D hybrid mesh $\mathcal{M}$ with curvilinear triangles and general quadrilaterals

How to build $\mathcal{S}_1^0(\mathcal{M})$?

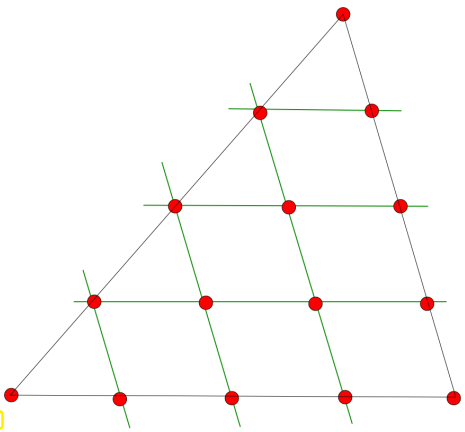
3.8.1 Affine Equivalence



- 建立了local shape function之间的联系（reference domain → actual domain)



(local) interpolation nodes for $\mathcal{S}_3^0(\mathcal{M})$



(local) interpolation nodes for $\mathcal{S}_4^0(\mathcal{M})$

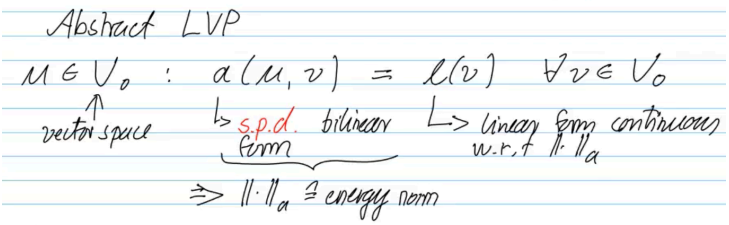
4 FEM: Convergence and Accuracy

convergence: How fast does u_h get closer to u as we vary our discretization scheme?

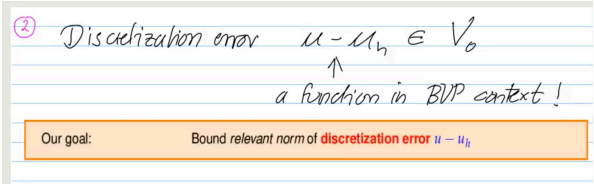
⇓ How to define "close"? We need accuracy to define metrics!

Accuracy: How close is u_h to u?

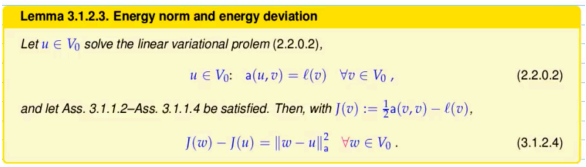
4.1 Abstract Galerkin Error Estimates



Necessary condition for the existence of the solution!



- find a bound or relative norm for this discretization error.



The energy norm of the discretization error squares is equal to the deviation of the energy:

It still has other norm in other applications, but let's focus energy norm:

In this chapter, we defined the conception of errror, found the energy norm and energy deviation, Galerkin orthogonality and cea's lemma. cea's lemma told us the galerkin discretization error is determined by the Galerkin discretization space, which can be determined by refinement - p-refinement/h-refinement.

4.2 Empirical (Asymptotic) Convergence of Lagrangian FEM

Summary

- h-refinement $\rightarrow$ algebratic convergence
- L^2 - convergence faster than $\|\cdot\|$-convergence
- Asymptotic convergence depends on the smoothness of u

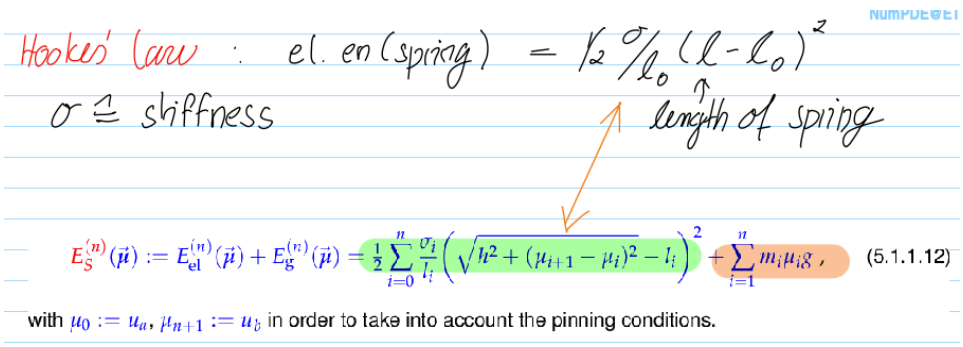
4.3 A Priori (Asymptotic) Finite Element Error Estimates

5 Non-Linear Elliptic Boundary Value Problem

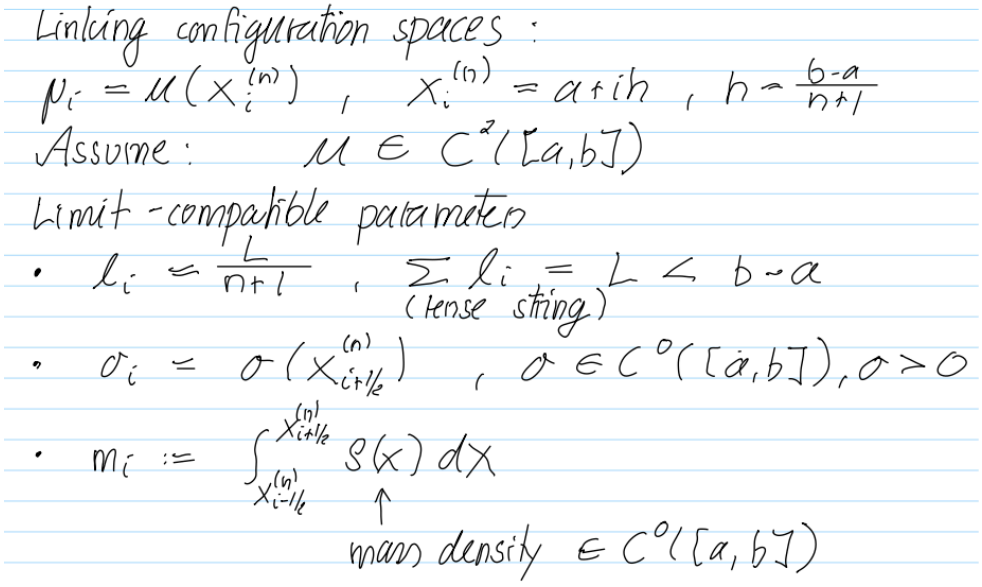
In this chapter we go beyond the linear boundary value problems exclusively discussed so far in the previous chapters. To begin with, the chapter revisits the problems of finding the equilibrium configuration for a thin elastic membrane. It will turn out that a refined mathematical model will be inherently non-linear and that model will serve as a starting point for the discussion of non-linear variational calculus. This will yield non-linear variational problems, to which (finite-element) Galerkin discretization can be applied. Finally, we will take a brief look at iterative schemes for the solution of the resulting non-linear systems of equations.

5.1 Non-linear Elastic Membrane Models

这一章节核心讨论的问题就是如何推导出elastic model的energy。以一维的问题举例，我们首先考虑一个simplifed model with n d.o.f (using vertical displacement of point masses with equidistant mesh). In this setting, the potential energy is equals to elastic energy + gravitational energy (in the discreted form). For the elastic energy, it has such form:



Next, we applied continuum limit to link the continuous function space and discrete space with sampling idea.



$$F_{el}^{(n)}(u) = \frac{1}{2}\sum_{i=0}^{n}\frac{n+1}{L}\sigma(x_{i+1/2}^{(n)})\left(\sqrt{h^2+(u(x_{i+1}^{(n)})-u(x_i^{(n)}))^2}-\frac{L}{n+1}\right)^2, \quad (5.1.1.24)$$

$$E_g^{(n)}(u) = g\cdot\sum_{i=1}^{n}\int_{x_{i-1/2}^{(n)}}^{x_{i+1/2}^{(n)}}\rho(x)\,dx\cdot u(x_i^{(n)})\,. \quad (5.1.1.25)$$

Then, let $n \to \infty$. Note that the limit of elastic energies via Taylor expansion.

Limit of elastic energies via Taylor expansion

$$\begin{aligned}\sqrt{h^2+(u(x+\tfrac{1}{2}h)-u(x-\tfrac{1}{2}h))^2} &= \sqrt{h^2+(u'(x)h+O(h^2))^2}\\ &= \sqrt{h^2+|u'(x)|^2h^2+O(h^3)}\\ &= h\sqrt{1+|u'(x)|^2}\sqrt{1+O(h)}\\ &= h\sqrt{1+|u'(x)|^2+O(h^2)} \quad \text{for } h\to 0\,,\end{aligned} \tag{5.1.1.29}$$

$X \leftrightarrows X_{i+1/2}^{(n)}$

$\sqrt{1+\delta} = (1+\frac{1}{2}\delta+O(\delta^2)) \to$

$h = \frac{b-a}{n+1}$

最后会得到一个形式。Same trick for 2D cases。但是我们发现这个形式不是quadratic energy形式。接着，我们采用了taut membrane limit，也就是假设膜本来就是拉紧的，external force只能产生small displacement。这也意味着 $L \to 0$，从而我们可以将我们的energy functional做进一步的化简，从而得到

A string/membrane is called taut, when it is strongly pre-stretched only due to the boundary conditions, that is, the elastic energy is large already without acting gravitational force. Quantitatively speaking, this means that $L \ll b-a$ for either the elastic string model of Section 5.1.1 or the two-dimensional membrane model of Section 5.1.2. This also means that even a considerable force will only bring about a *small deformation* of the elastic string/membrane. Hence, the situation considered in this section is also known as the small displacement setting.

Formally the taut membrane limit consists of two steps targeting the expressions for the elastic energy:

❶ Replace the stiffness coefficient function $\sigma = \sigma(x)$ with a **renormalized** stiffness $\widehat{\sigma} := (b-a)\sigma/L$ (string), $\widehat{\sigma} := \sigma/L$ (membrane), and consider the latter independent of L.

❷ Set $L \leftarrow 0$ in the formulas for the elastic energy.

5.1.3. Taut Membrane Limit $\hat{=}$ $L \to 0$

Sect 1.2.1 : Quadratic energies ?

$$J_S(u) := J_{el}(u) + J_g(u) = \int_a^b \frac{1}{2}\frac{b-a}{L}\sigma(x)\left(\sqrt{1+|u'(x)|^2}-\frac{L}{b-a}\right)^2 + g\rho(x)u(x)\,dx\,. \tag{5.1.1.33}$$

Renormalized stiffness : $\hat{\sigma}(x) = \frac{b-a}{L}\sigma(x)$: independent of L

Set $L := 0$

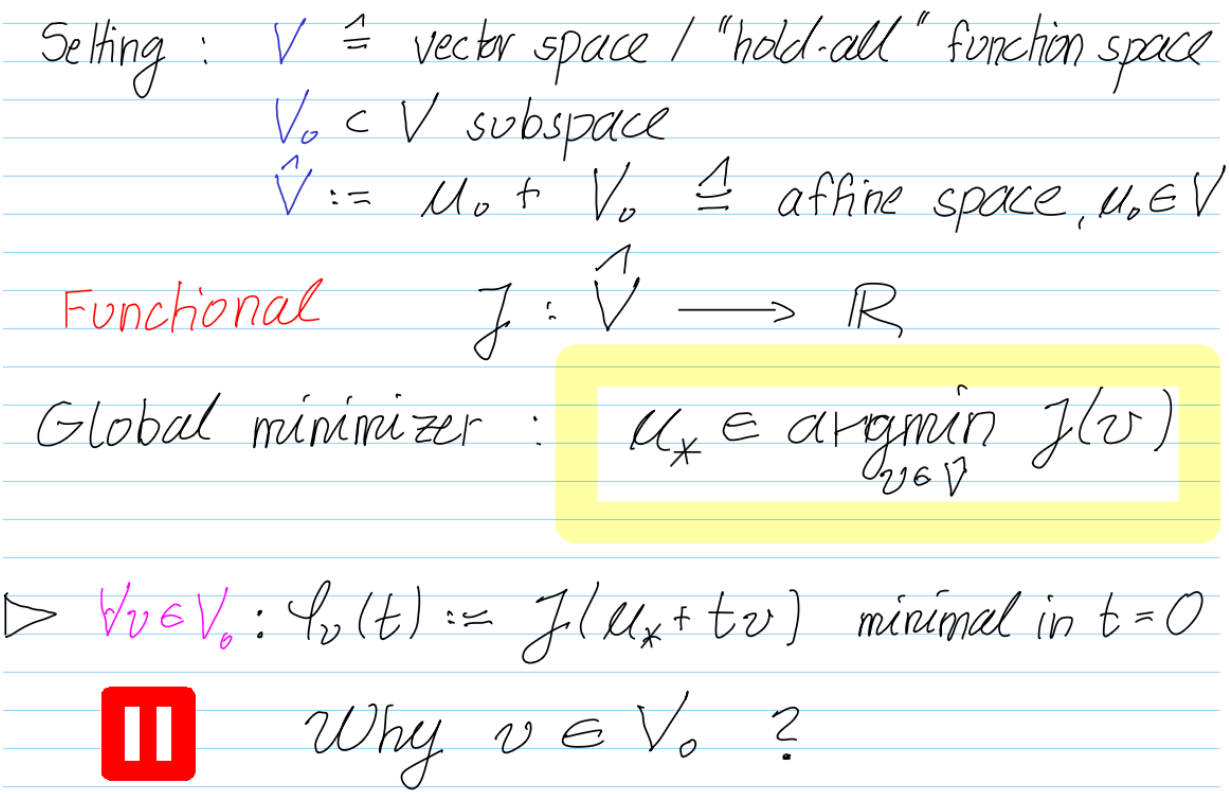
$$\widetilde{J}_S(u) = \frac{1}{2}\int_a^b \widehat{\sigma}(x)\left(1+|u'(x)|^2\right) + g\rho(x)u(x)\,dx\,. \tag{5.1.3.1}$$

→ A quadratic functional

irrelevant for minimizer

5.2 Calculus of Variations

5.2.1 Calculus of Variations: Fundamental Idea



- 如果不等于V_0就会破坏解的结构，因为$u_* \in \hat{V} \to u_* = u_0 + V_0$. If $u_* \notin V_0 \to u_* + tv \notin \hat{V}$。即解不在我们的functional space中了。
- 而通常，如果有Dirichlet BC，V_0通常是在这些边界上等于0的function space。因为一个function space是subspace的前提是里面存在zero space，而在我们施加了Dirichlet BC后，我们将无法实现function在subspace中这种情况，因此我们才考虑将我们的两个space拆开。解空间是一个affline space，由一个solution satisfying BC + subspace
- 上述公式讨论了我们两个space概念的由来，即为什么我们会讨论test space的概念。其原始由来是从functional。

Next, we give the equation for computing the global minimizers, this is called characterization of global minimizers. 注意这是必要条件，也就是如果u_*是global minimizer，一定得满足这个条件。但是只满足这个条件却不一定能推出u_* is global minimizer。

Minimization of functional ⇒ Variational Equation [referred to as calculus of variations]。而这一章节的核心就是从minimization problem如何推出等价的variational equation。（in chapter 1, We have discussed the U&E for the quadratic minimization problem. The conclusion hold for general cases.)。而我们主要依赖的是下面这个theorem。

② $\Rightarrow$ $\frac{d\varphi_v}{dt}(0) = 0 \quad \forall v \in V_0$

Theorem 5.2.1.5. Characterization of global minimizers

Let $J : \widehat{V} \to \mathbb{R}$ a continuous functional and

$$u_* \in \operatorname*{argmin}_{v \in \widehat{V}} J(v)$$

a global minimizer of J over the affine space $\widehat{V} = v_0 + V_0 \subset V$. Then, if $\varphi_v(t) := J(u_* + tv)$ is differentiable in $t = 0$ for all $v \in V_0$,

$$\frac{d\varphi_v}{dt}(0) = 0 \quad \forall v \in V_0 \,. \tag{5.2.1.6}$$

↑ A necessary condition

- generalize sect. 1.4.1, where J was a quadratic functional
  - quadratic minimization problem: Seek $u_* \in \widehat{V}$: $u_* = \operatorname*{argmin}_{v \in \widehat{V}} J(v)$

    Assume that $u_* \in \widehat{V}$ solves (1.4.1.2) and, for arbitrary $v \in V_0$ define the function.

    $$\varphi_v : \mathbb{R} \to \mathbb{R} \quad , \quad \varphi_v(t) = J(u_* + tv) \,, \quad t \in \mathbb{R} \,. \tag{1.4.1.3}$$

  - Invoking the bilinearity and symmetry of a and the linearity of ℓ, we easily find

    $$\varphi_v(t) = \tfrac{1}{2}t^2 a(v,v) + t a(u_*, v) + \tfrac{1}{2} a(u_*, u_*) - t\ell(v) - \ell(u_*) \,, \quad t \in \mathbb{R} \,.$$

    Note that φ_v is a quadratic polynomial, its graph a parabola. By definition of u_* the function φ_v has a global minimum at $t = 0$. Since φ_v is continuously differentiable, we conclude that

  - $$\frac{d\varphi_v}{dt}(0) = a(u_*, v) - \ell(v) = 0 \,. \tag{1.4.1.4}$$

    This will hold for any $v \in V_0$ and we end up with a new form of equation, a linear variational equation,

    $$u_* \quad \text{solves (1.4.1.2)} \quad \Rightarrow \quad a(u_*, v) = \ell(v) \quad \forall v \in V_0 \,. \tag{1.4.1.5}$$

  - 可以得到variational equation的具体形式是一个LSE.
- 注意，一个泛函的极值问题等价于这里的偏导数等于0,见下面这个例子：

Ex. 5.2.1.8: $V = \mathbb{R}^n$, $n \in \mathbb{N}$

$J : \mathbb{R}^n \longrightarrow \mathbb{R}$ smooth

- Known: $x_* \in \operatorname*{argmin}_{z \in \mathbb{R}^n} J(z) \Rightarrow \operatorname{grad} J(x_*) = 0$

  $(5.2.1.6) \Leftrightarrow \frac{d}{dt}\{t \mapsto J(x_* + tv)\}_{|t=0} = \operatorname{grad} J(x_*) \cdot v = 0 \quad \forall v \in \mathbb{R}^n$

而先在问题的关键是如何求这个函数关于t的导数？

而这里，我们发现，这其实就是在v方向上的方向导数：

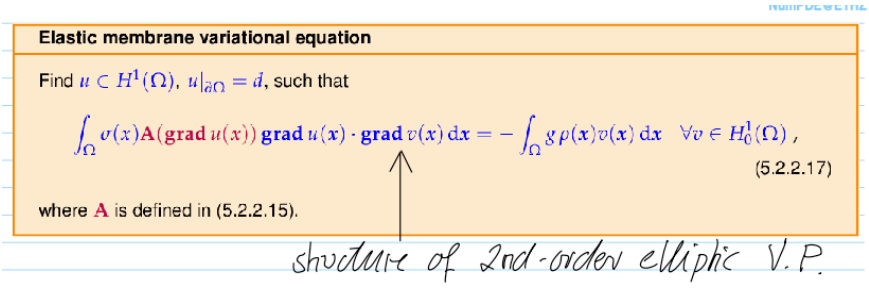
$\frac{d\varphi_v}{dt}(0) = \lim_{t \to 0} \frac{\varphi_v(t) - \varphi_v(0)}{t} = \lim_{t \to 0} \frac{J(u_* + tv) - J(u_*)}{t}$

$=$ directional derivative (in direction v): $\langle DJ(u_*), v \rangle$

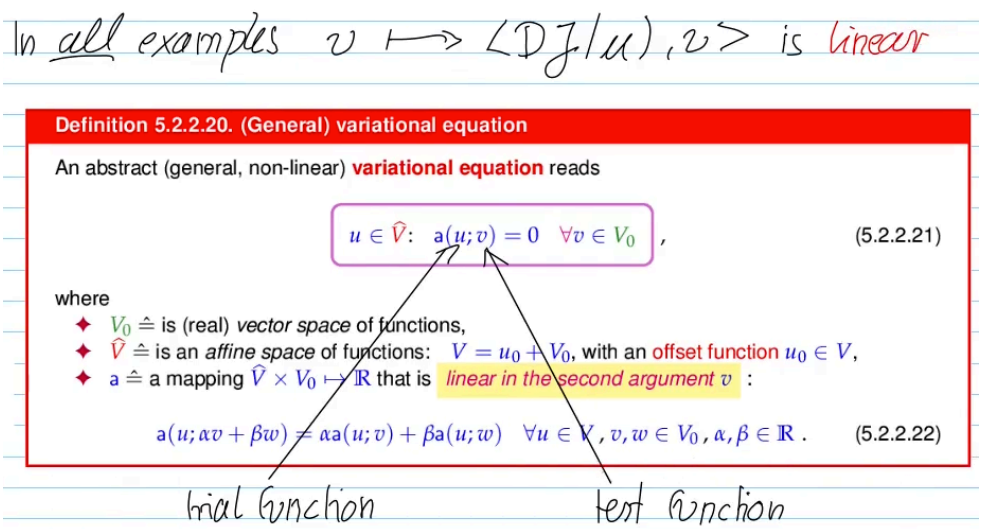
- 因此，大致的思路就是我们可以基于方向导数=0列出来一个方程，这个方程里面有u，也有方向导数v。而最终的解就是在$\forall v \in H_0^1$,使得该方程都成立的u即为最小值解。

一个泛函的minimization problem其实就等价于寻找各个方向上v(test function)的方向导数都为0点u^*，得出的这个方程即为我们的variational equation。因此我们重点讨论如何定义方向导数。如果是个linear functional的话，我们只需要把u替换成v带进去即为v方向上的方向导数（利用binlinearity可以很容易推导出来variational equation。）。而对于nonlinear functional,我们则需要用到Taylor expansion，只取一阶近视，而二阶等高阶这些不要(掌握multidimensional taylor expansion。 从而我们可以得到minimizaiton problem解的necessary condition（其实这里感觉是等价的，不一定只是necessary condition）

然后，我们先举了一些具体的例子：directional derivate for elastic string potential energy， variational equation for 2d elastic membrane.



最后，我们给出general variational equation：



这里我们得到了general structure。同时我们也发现，无论functional线不线性，对于v，始终是线性的。见5.2.2.22。感觉挺合理的，就是假定u不变的情况下，方程关于v是一个线性的。而如果functional是线性的，最后得到的variational equation可能对u，也就是trial function也是线性的。也就是整个a就是一个bilinear form。

然后我们又聚了一个在数学领域很重要的问题，我觉得这个问题可能在最优输运里面提及，即minimial surface problem for graphs.

For $u : \Omega \subset \mathbb{R}^2 \longrightarrow \mathbb{R}$, $u|_{\partial\Omega} = d$

$$G_u := \left\{ \boldsymbol{x} = [x_1, x_2, x_3]^\top : \begin{bmatrix} x_1 \\ x_2 \end{bmatrix} \in \Omega,\ x_3 = u\left(\begin{bmatrix} x_1 \\ x_2 \end{bmatrix}\right) \right\}, \tag{5.2.2.25}$$

$$\text{area}(G_u) = \int_\Omega \sqrt{1 + \|\mathbf{grad}\, u(x)\|_2^2}\, dx \longrightarrow \min \tag{5.2.2.27}$$

$J(u) :=$

Directional derivative by Taylor expansion

$$\begin{aligned}\sqrt{1 + \|\mathbf{grad}\, u(x) + t\,\mathbf{grad}\, v(x)\|_2^2} &= \sqrt{1 + \|\mathbf{grad}\, u(x)\|_2^2 + 2t\,\mathbf{grad}\, u(x) \cdot \mathbf{grad}\, v(x) + O(t^2)} \\ &= \sqrt{1 + \|\mathbf{grad}\, u(x)\|_2^2} \cdot \sqrt{1 + 2t \frac{\mathbf{grad}\, u(x) \cdot \mathbf{grad}\, v(x)}{1 + \|\mathbf{grad}\, u(x)\|_2^2} + O(t^2)} \\ &= \sqrt{1 + \|\mathbf{grad}\, u(x)\|_2^2} \left(1 + t \frac{\mathbf{grad}\, u(x) \cdot \mathbf{grad}\, v(x)}{1 + \|\mathbf{grad}\, u(x)\|_2^2} + O(t^2)\right),\end{aligned}$$

▷ Variational equation

$$u \in C^1_{pw}(\Omega),\ u|_{\partial\Omega} = d \quad : \quad \int_\Omega \frac{\mathbf{grad}\, u(x) \cdot \mathbf{grad}\, v(x)}{1 + \|\mathbf{grad}\, u(x)\|_2^2}\, dx = 0 \quad \forall v \in C^1_{pw,0}(\overline{\Omega}) \,. \tag{5.2.2.29}$$

V $\quad \rightsquigarrow \langle Dj(u), v \rangle$ $\quad V_0$

接着，通过structure of a 2nd-order elliptic V.P.,我们又介绍了不同程度的nonlinear问题：

Special (simpler) cases

Ex. 5.2.2.30 (Quasi-linear 2nd-order elliptic V.P.)

$$u \in X \subset H^1(\Omega): \quad \int_\Omega \mathbf{A}(x, u(x))\,\mathbf{grad}\, u(x) \cdot \mathbf{grad}\, v(x) + G(x, u(x))v(x)\, dx = \int_\Omega f(x)v(x)\, dx \quad \forall v \in X_0 \subset H^1(\Omega) \,. \tag{5.2.2.31}$$

$A : \Omega \times \mathbb{R} \longrightarrow \mathbb{R}^{2,2}$ indep. of derivatives of u

$G : \Omega \times \mathbb{R} \longrightarrow \mathbb{R}$

Ex. 5.2.2.32 (Semi-linear 2nd-order elliptic V.P.)

$$u \in X \subset H^1(\Omega): \quad \int_\Omega \mathbf{A}(x)\,\mathbf{grad}\, u(x) \cdot \mathbf{grad}\, v(x) + G(x, u(x))v(x)\, dx = \int_\Omega f(x)v(x)\, dx \quad \forall v \in X_0 \subset H^1(\Omega) \,, \tag{5.2.2.33}$$

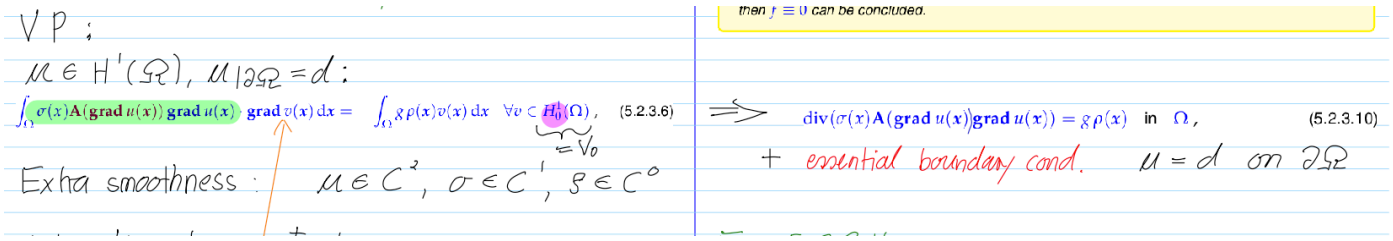
No derivatives in non-linear term !

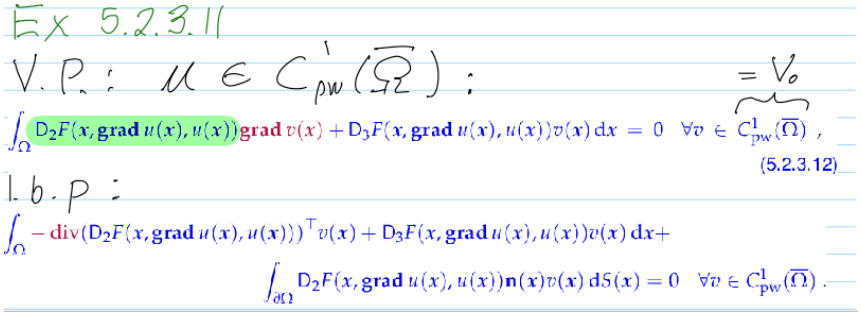
Also :

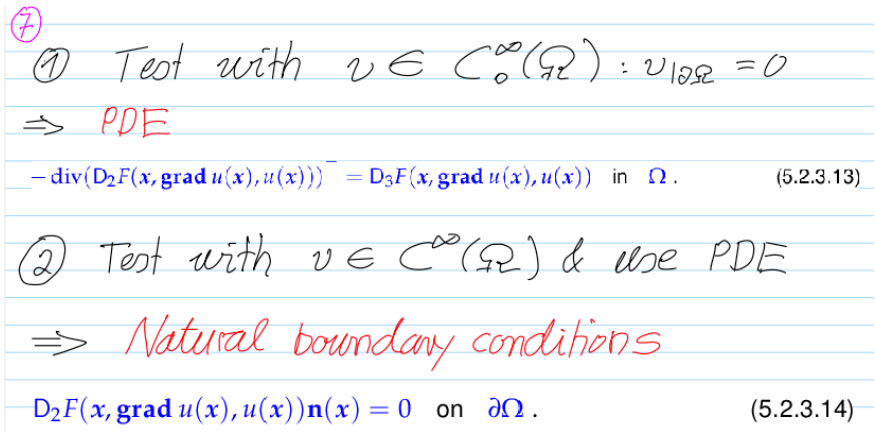
$$u \in H^1(\Omega): \quad \int_\Omega \kappa(x)\,\mathbf{grad}\, u \cdot \mathbf{grad}\, v\, dx + \int_{\partial\Omega} \Psi(u)\, v\, dS = \int_\Omega fv\, dx \quad \forall v \in H^1(\Omega) \,, \tag{1.8.0.8}$$

这里的非线性是怎么体现的呢？

最后，则是讨论如何从V.P推导出nonlinear BVPs，和第一章的方法完全一致，直接green's first formula做分布积分就行了，这里不做赘述。我们看几个例子



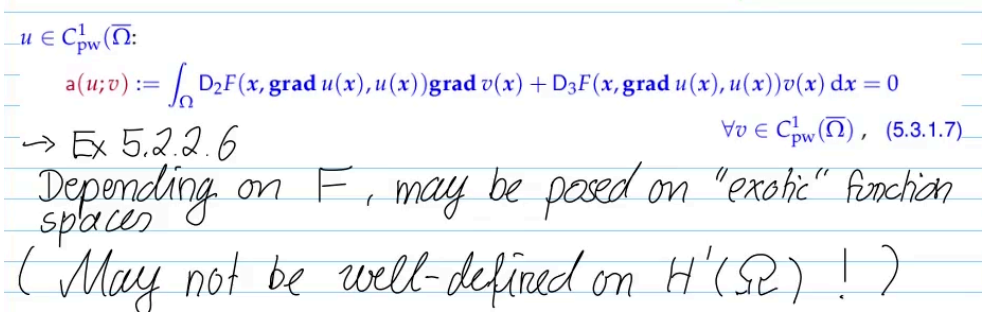




5.3 Galerkin Discretization of Non-Linear BVPs

5.3.1 Abstract Galerkin Discretization of Non-Linear Variational Problems

这个地方有一个点需要注意:



- 当我们遇到这种非线性PDE时（尤其当$u(x)$非常次数非常高的时候），我们通过PDE的理论去分析U&E从而得到我们合理的function space,通常会得到的一个非常"exotic" function space。这个function space may not be well-defined on $H^1(\Omega)$。我们上面研究的都是elliptic PDE的情况，而所谓的非线性也一般都是coefficient里面带有u^2这种，所以整个PDE还是在elliptic的framework下的，问题不是很复杂。
- 但是，当我们用finite element space去近似他的解时，在这个reduced的space种，almost always suitable for Galerkin discretization。

Note : Lagr. FE space $S_0^p(\mathcal{M}) \subset C^1_{pw}(\overline{\Omega})$

↓

Almost always suitable for Gal. disc.

⇓

$u_h \in \mathcal{S}_p^0(\mathcal{M})$:

$$\int_\Omega \mathsf{D}_2 F(\boldsymbol{x}, \mathbf{grad}\, u_h(\boldsymbol{x}), u_h(\boldsymbol{x}))\mathbf{grad}\, v_h(\boldsymbol{x}) + \mathsf{D}_3 F(\boldsymbol{x}, \mathbf{grad}\, u_h(\boldsymbol{x}), u_h(\boldsymbol{x}))v_h(\boldsymbol{x})\,\mathrm{d}\boldsymbol{x} = 0$$

$\forall v_h \in \mathcal{S}_p^0(\mathcal{M})$, (5.3.1.8)

[Discrete variational problem]

接着，after discretization，我们可以用numerical methods for cse里面的技巧去对这个离散的问题做newton iteraton。但是，我们接下来要介绍的这种方法，在我们做discretization之前，我们仍可以用iterative method去做近似。

5.3.2 Iterative Methods in Function Space

Option : Solve NLSE $F(\vec{v}) = 0$ by some iterative method.

Alternative : Idea: Design the iterative method for (5.2.2.21) before discretization!

The usual warnings apply

In many cases iterative method will only converge, if the initial guess is chosen *"sufficiently" close* to the (desired) solution.

Solutions of (general) discrete variational problems need *not* be *unique*: the (approximate) solution computed by a convergent iterative method may depend on the intial guess.

在介绍具体的方法之前，我们想要强调的一点是通常迭代方法并不是求解问题里面的难点，求解的难点在于initial guess的选择。以及对于nonlinear problems，解通常不是唯一的，因此最后迭代收敛的解是否是你想要的也需要仔细斟酌。

Fixed-Point Iterations

下面，我们举一个VP类，然后主要用iterative method解决这类问题：model problem

Model problem, $\Omega \subset \mathbb{R}^d$:

$u \in \hat{V} \subset H^1(\Omega)$: $\int_\Omega \mathbf{A}(\boldsymbol{x}, \mathbf{grad}\, u(\boldsymbol{x}))\, \mathbf{grad}\, u(\boldsymbol{x}) \cdot \mathbf{grad}\, v(\boldsymbol{x}) + c(u(\boldsymbol{x}))u(\boldsymbol{x})v(\boldsymbol{x})\,\mathrm{d}\boldsymbol{x}$

$= \int_\Omega f(\boldsymbol{x})v(\boldsymbol{x})\,\mathrm{d}\boldsymbol{x} \quad \forall v \in V_0 \subset H^1(\Omega)$.

$A : \Omega \times \mathbb{R}^d \to \mathbb{R}^{d,d}$, $c : \mathbb{R} \to \mathbb{R}$

Examples:

(i) Elastic membrane V.P.

Find $u \in H^1(\Omega)$, $u|_{\partial\Omega} = d$, such that

$$\int_\Omega v(x)\mathbf{A}(\mathbf{grad}\, u(x))\, \mathbf{grad}\, u(x) \cdot \mathbf{grad}\, v(x)\,\mathrm{d}x = -\int_\Omega g\rho(x)v(x)\,\mathrm{d}x \quad \forall v \in H^1_0(\Omega)\,, \tag{5.2.2.17}$$

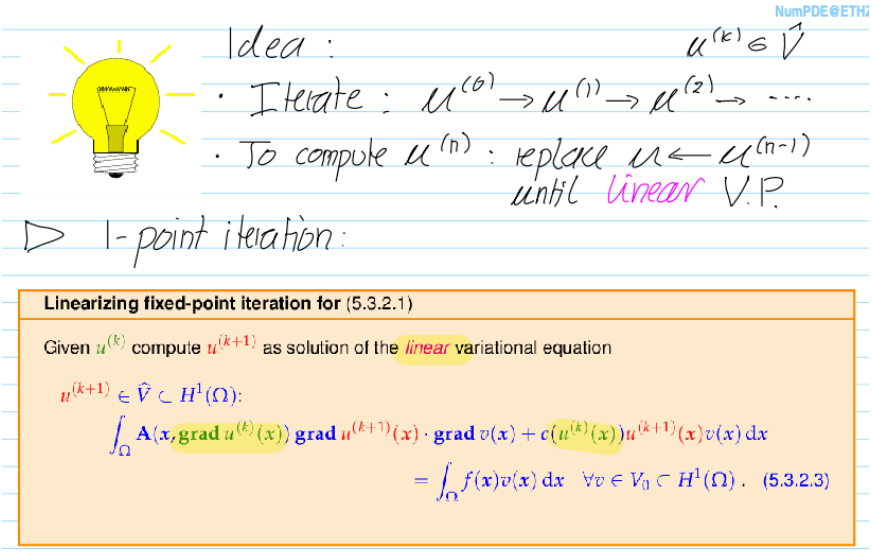
where A is defined in (5.2.2.15).

(ii) Graph minimal surface V.P.

$u \in C^1_{\mathrm{pw}}(\Omega)$, $u_{\partial\Omega} = d$: $\int_\Omega \frac{1}{1 + \|\mathbf{grad}\, u(x)\|_2^2}\, \mathbf{grad}\, u(x) \cdot \mathbf{grad}\, v(x)\,\mathrm{d}x = 0 \quad \forall v \in C^1_{\mathrm{pw},0}(\overline{\Omega})$. (5.2.2.29)

- 之前的nonlinear elastic membrane和graph minimal surface都是model problem这个类下的。

下面开始解绍fixed-point iteration。其核心的思想就是把nonlinear term中的解用上一步代替，从而构造一个linear variational problem，然后再用galerkin discretization离散化求解这个linear variational problem



Linearizing fixed-point iteration for (5.3.2.1)

Given $u^{(k)}$ compute $u^{(k+1)}$ as solution of the *linear* variational equation

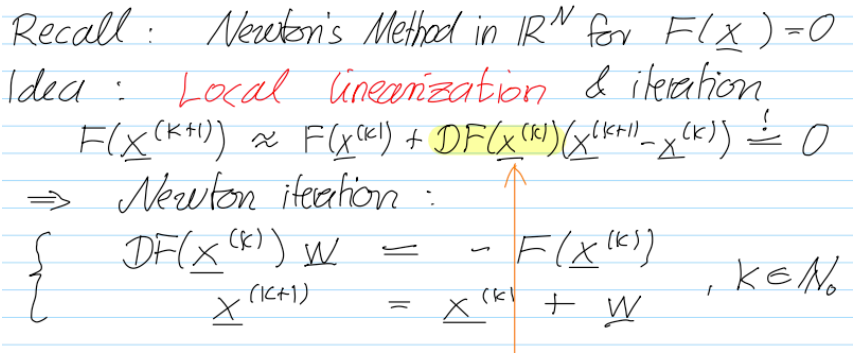
$u^{(k+1)} \in \hat{V} \subset H^1(\Omega)$:

$$\int_\Omega \mathbf{A}(\boldsymbol{x}, \mathbf{grad}\, u^{(k)}(\boldsymbol{x}))\, \mathbf{grad}\, u^{(k+1)}(\boldsymbol{x}) \cdot \mathbf{grad}\, v(\boldsymbol{x}) + c(u^{(k)}(\boldsymbol{x}))u^{(k+1)}(\boldsymbol{x})v(\boldsymbol{x})\,\mathrm{d}\boldsymbol{x} = \int_\Omega f(\boldsymbol{x})v(\boldsymbol{x})\,\mathrm{d}\boldsymbol{x} \quad \forall v \in V_0 \subset H^1(\Omega)\,. \tag{5.3.2.3}$$

- 按我的理解，不动点迭代是要构造格式的，不知道他这里咋就可以直接这样操作了。在研究研究

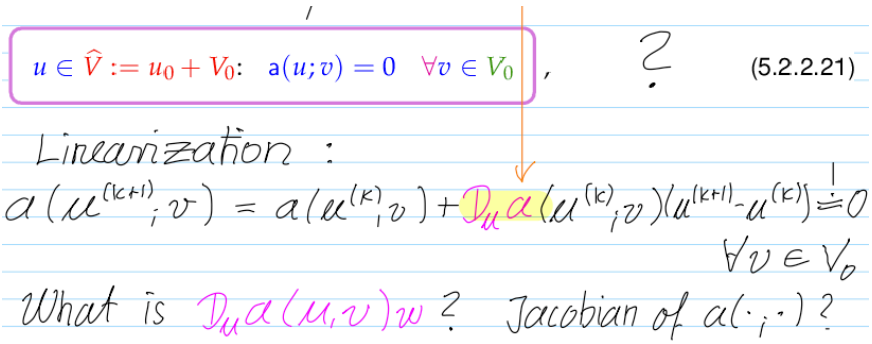
Newton's Method

另一种方法是牛顿法。我们先来看看在有限维空间中牛顿法是怎么做的：



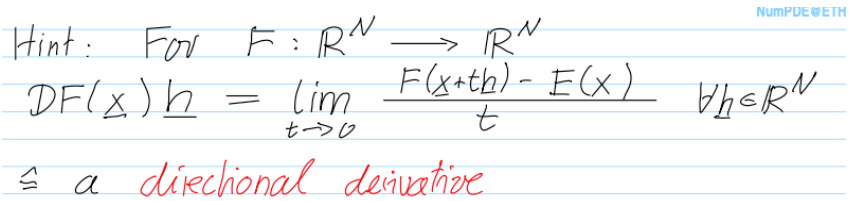
- 核心是在迭代是做local linearization
- 这里的w称为newtwon correction/update

接着，我们看看如何在function space中用，同样的，我们用关于变量u的导做一阶linearization：

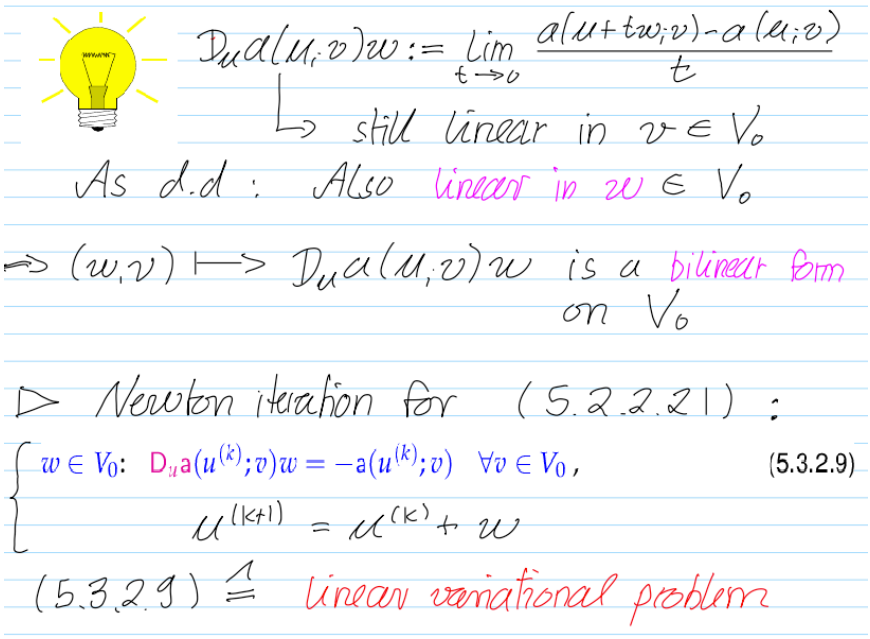


- 这里的$u^{k+1} - u^k$就是我们的newton correction，也是我们要求的

接下来的核心问题是如何算这个对u的jacobian，其实，回顾一下，我们可以知道，这也是directional derivative，而且是沿着t方向的方向导数。所谓的方向导数引入t逼近于0。

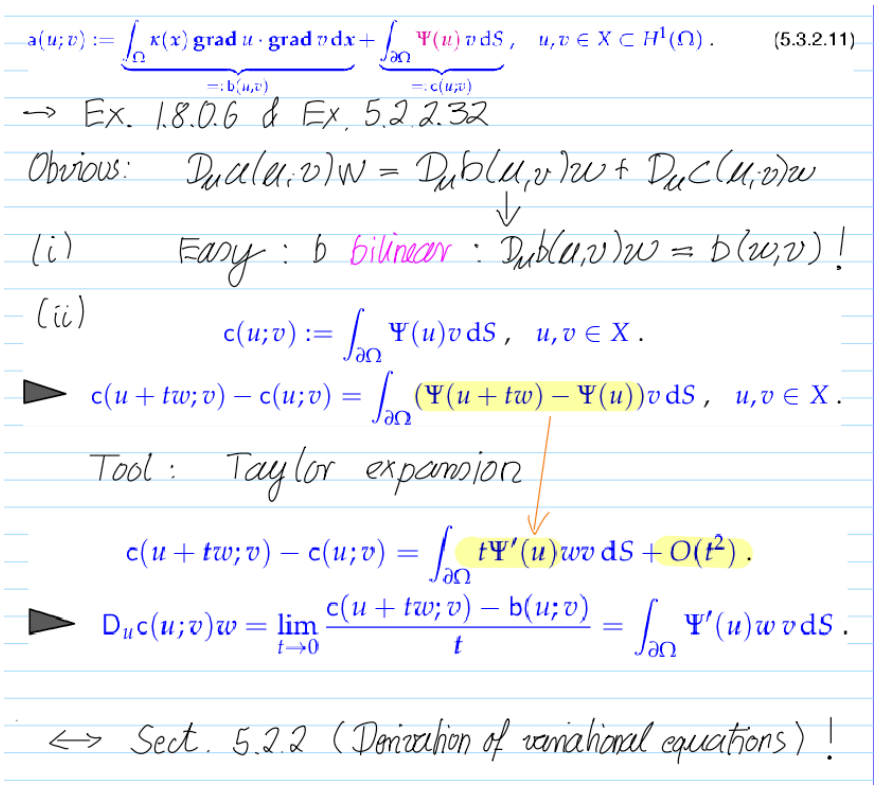


在上一章节，我们已经讨论了大量计算directional derivative的理论。



- 基于前面的讨论，我们知道求完倒后的项其实就是关于v, w的bilinear form。见5.3.2.9。因此，我们的问题通常变换成了一个linear variational problem，我们离散化进行求解。

接下来，我们看一个semi-linear v.p的例子：



5.3.3 Galerkin Discretization of Linearized Variational Equations

接下来，我们想问，在discretization前离散的优势是什么?

Benefit of "iterations in function space":

Step-dependent (adaptive) choice of Galerkin trial/test spaces.

For iterative methods in function space different Galerkin trial/test spaces can be chosen differently in every step.

- 就是每一步迭代我们可以选择不同的discretization space。比如最开始迭代时，解离真实解很远，我们可以用粗糙的discretization space做快速估测。之后随着迭代靠近真实解，我们再选用更加精细的discretization space做逼近。我个人目前是如果discretization后做迭代，那么每一次迭代向量的维数固定。而在函数空间迭代，两个不同维度函数似乎也可以计算

Galerkin discretization for fixed-point iteration

这两个例子直接看lecture notes更清楚

$u^{(k-1)} \in \hat{V} \subset H^1(\Omega)$:

$$\int_\Omega \mathbf{A}(x, \operatorname{grad} u^{(k)}(x)) \operatorname{grad} u^{(k+1)}(x) \cdot \operatorname{grad} v(x) + c(u^{(k)}(x)) u^{(k+1)}(x) v(x) \,\mathrm{d}x = \int_\Omega f(x) v(x) \,\mathrm{d}x \quad \forall v \in V_0 \subset H^1(\Omega) . \tag{5.3.2.3}$$

Discrete test space $V_{0,h}^{(k)} := S_p^0(\mathcal{M}^{(k)})$ (Lagrange FE space)

Discrete trial space $V_h^{(k)} := u_{0,h}^{(k)} + V_{0,h}^{(k)}$ ← offset function

⑥ Choose bases $\mathcal{B}_h^{(k)} := \{b_k^1, \ldots, b_k^{N_k}\}$ of $V_{0,h}^{(k)}$, $N_k := \dim V_{0,h}^{(k)}$

▷ Iterates $\vec{\mu}^{(k)} \in \mathbb{R}^{N_k}$ from

$$\mathbf{M}(\vec{\mu}^{(k-1)}) \vec{\mu}^{(k)} = \vec{\varphi}(\vec{\mu}^{(k-1)}) \tag{5.3.3.4}$$

with $u_h^{(k)} = u_{0,h}^{(k)} + \sum_{\ell=1}^{N_k} \left(\vec{\mu}^{(k)}\right)_\ell b_k^\ell$, $u_h^{(k-1)} = u_{0,h}^{(k-1)} + \sum_{\ell=1}^{N_{k-1}} \left(\vec{\mu}^{(k-1)}\right)_\ell b_{k-1}^\ell$,

$$\mathbf{M}(\vec{\mu}^{(k-1)}) := \left[\int_\Omega \mathbf{A}(x, \operatorname{grad} u_h^{(k-1)}(x)) \operatorname{grad} b_k^j(x) \cdot \operatorname{grad} b_k^i(x) + c(u_h^{(k-1)}) b_k^j(x) b_k^i(x) \,\mathrm{d}x \right]_{i,j=1}^{N_k} \in \mathbb{R}^{N_k, N_k},$$

$$\vec{\varphi}(\vec{\mu}^{(k-1)}) := \left[\int_\Omega f(x) b_k^i(x) \,\mathrm{d}x - \int_\Omega \mathbf{A}(x, \operatorname{grad} u_h^{(k-1)}(x)) \operatorname{grad} u_{0,h}^{(k)}(x) \cdot \operatorname{grad} b_k^i(x) + c(u_h^{(k-1)}) u_{0,h}^{(k)}(x) b_k^i(x) \,\mathrm{d}x \right]_{i=1}^{N_k} \in \mathbb{R}^{N_k}.$$

Galerkin discretization for Newton iteration

§ 5.3.3.5 (Galerkin discretization for Newton's method)

V.P.: $u \in \widehat{V} := u_0 + V_0$: $\mathsf{a}(u;v) = 0 \quad \forall v \in V_0$, (5.2.2.21)

▷ Newton iteration for (5.2.2.21):

$$\begin{cases} w \in V_0: \ \mathsf{D}_u\mathsf{a}(u^{(k)};v)w = -\mathsf{a}(u^{(k)};v) \quad \forall v \in V_0, & (5.3.2.9) \\ u^{(k+1)} = u^{(k)} + w \end{cases}$$

Galerkin test space: $V_{0,h}^{(k)} \subset V_0$

Galerkin trial space: $\widehat{V}_h^{(k)} := u_{0,h}^{(k)} + V_{0,h}^{(k)}$

▷ Discrete linear variational equation for Newton update

$$w_h \in V_{0,h}^{(k)}: \ \mathsf{D}_u\mathsf{a}(u_h^{(k-1)};v_h)w_h = -\mathsf{a}(u_h^{(k-1)};v_h) \quad \forall v_h \in V_{0,h}^{(k)}, \qquad (5.3.3.6)$$

+ Computation of next iterate → projection/interpolation

$$u_h^{(k)} := \mathsf{P}_h^{(k)}(u_h^{(k-1)} + w_h). \quad \left[u_h^{(k-1)} + w_h \notin \widehat{V}_h^{(k)} \text{ possible}\right]$$

⑦ Choose bases $\mathcal{B}_h^{(k)} := \{b_k^1, \ldots, b_k^{N_k}\}$ of $V_{0,h}^{(k)}$, $N_k := \dim V_{0,h}^{(k)}$

⇒ Newton updates $\vec{\omega} \in \mathbb{R}^{N_k}$

LSE: $\mathbf{M}(\vec{\mu}^{(k-1)})\vec{\omega} = \vec{\varphi}(\vec{\mu}^{(k-1)})$, (5.3.3.7)

with $w_h = \sum_{\ell=1}^{N_k} (\vec{\omega})_\ell b_k^\ell$, $u_h^{(k-1)} = u_{0,h}^{(k-1)} + \sum_{\ell=1}^{N_{k-1}} \left(\vec{\mu}^{(k-1)}\right)_\ell b_{k-1}^\ell$,

$\mathbf{M}(\vec{\mu}^{(k-1)}) := \left[\mathsf{D}_u\mathsf{a}(u_h^{(k-1)};b_k^j)b_k^i\right]_{i,j=1}^{N_k} \in \mathbb{R}^{N_k,N_k}$, $\vec{\varphi}(\vec{\mu}^{(k-1)}) := \left[-\mathsf{a}(u_h^{(k-1)};b_k^i)\right]_{i=1}^{N_k} \in \mathbb{R}^{N_k}$.

⇒ Next iterate $\mu^{(k)} \in \mathbb{R}^{N_k}$

$$\sum_{\ell=1}^{N_k} \left(\vec{\mu}^{(k)}\right)_\ell b_k^\ell = \mathsf{P}_h^{(k)}(u_h^{(k-1)} + w_h) - u_{0,h}^{(k)}. \qquad (5.3.3.8)$$

- 这个地方的课件出错了，projection部分：

$$\underbrace{u_h^{(k)}}_{\in \widehat{V}_h^{(k)}} = P_h^{(k)} \underbrace{u_h^{(k-1)}}_{\in \widehat{V}_h^{(k-1)}} + \underbrace{w_h}_{\in V_{h,0}^{(k)}}$$

- 核心的关键是要注意这里的projection，上一次迭代和下一次迭代函数维度不一定相同。不是太清楚这里映射怎么构造，再看看课件
- 看看习题里面有没有关于projection operator的介绍

6 Second-Order Linear Evolution Problems

This chapter exclusively deals with linear evolution problems. We can distinguish two fundamental classes, dissipative and conservative evolutions. This is reflected by the structure of the chapter, which comprises two sections, Section 6.2 devoted to **dissipative** (parabolic) evolutions, Section 6.3 addressing **conservative** (hyperbolic) evolutions. Each section first develops variational formulations, then discretization in space, and, finally, discretization in time. Results on convergence are reviewed and discussed.

> Even though it is derived from the conservation of energy, it captures the "dissipative" nature of heat. This means that if you have a localized "hot spot" or any kind of non-uniform distribution of temperature in a medium, the heat equation dictates that this hot spot will spread out and dissipate over time. High temperature regions decrease while low temperature regions increase until a uniform temperature is achieved. This spreading out and equilibration is what is referred to as "dissipative" behavior.
>
> The classification of PDEs into categories like "elliptic", "parabolic", and "hyperbolic" is based on the mathematical form of the equation, and not necessarily its physical interpretation. These terms originate from the discriminant of the characteristic polynomial of the PDE, which, in simpler terms, gives insight into the nature of the solutions of the equation. For the heat equation, the PDE is parabolic. Solutions to parabolic PDEs exhibit behaviors similar to heat flow: smoothness, diffusion, and dissipation of singularities.

首先，介绍了本章节研究的一类方程。二阶的线性evolution problems。并提到该问题的configuration space定义在时间和空间上。是一个**Space-time cylinder**。然后指出时间域上只有initial condiiton，没有final conditions。另外，时间domain上has a direction(causality)也就是有因果性。

接着，我们通过一个特别的问题引出我们的研究对象。Heat Equation，我们通过balance law（conservation of energy）来写出物理上合理的方程（相比于stationary equation，唯一变化的是多了一个energy stored项。也就是温度✖heat copacity对时间的导数（有点迷惑这一项的合理性）。我们写方程的时候使用积分来写能量守恒的。然后我们运用格林定理以及做一些简单的推导以及用上fourier law，即可得到我们的微分方程。也就是localization方程。

$$\frac{\partial}{\partial t}(\rho u) - \operatorname{div}(\kappa(x)\,\mathbf{grad}\,u) = f \quad \text{in} \quad \tilde{\Omega} := \Omega \times]0, T[$$

然后我们讨论时空边界条件。对于空间边界条件，结论和second-order elliptic boundary value problem一致，在任何空间Boundary上得有BC且唯一。

For second order parabolic evolutions we can/must use the *same* spatial boundary conditions as for stationary second order elliptic boundary value problems.

On $\partial\Omega]0,T[$ we can impose any of the boundary conditions discussed in Section 1.7:
- Dirichlet boundary conditions $u(\boldsymbol{x},t) = g(\boldsymbol{x},t)$, see (7.1.1.7) (fixed surface temperature),
- Neumann boundary conditions $\mathbf{j}(\boldsymbol{x},t)\cdot\boldsymbol{n} = -h(\boldsymbol{x},t)$ (fixed heat flux through surface),
- radiation boundary conditions $\mathbf{j}(\boldsymbol{x},t)\cdot\boldsymbol{n} = \Psi(u(\boldsymbol{x},t))$,

and any combination of these as discussed in Ex. 1.7.0.10, yet, *only one* of them at any part of $\partial\Omega\times]0,T[$, see Rem. 1.7.0.9.

same as in stationary setting

- Initial conditions : ("temporal b.c.")

$$u(x,0) = u_0(x)$$

IBVP

$$\frac{\partial}{\partial t}(\rho(\boldsymbol{x})u) - \operatorname{div}(\kappa(\boldsymbol{x})\,\mathbf{grad}\,u) = f \quad \text{in} \quad \tilde{\Omega} := \Omega\times]0,T[\,, \tag{9.2.1.6}$$

$$u(\boldsymbol{x},t) = g(\boldsymbol{x},t) \quad \text{for} \quad (\boldsymbol{x},t)\in\partial\Omega\times]0,T[\,, \tag{9.2.1.7}$$

$$u(\boldsymbol{x},0) = u_0(\boldsymbol{x}) \quad \text{for all} \quad \boldsymbol{x}\in\Omega\,. \tag{9.2.1.8}$$

Comp. cond. :

$$u_0(x) = g(x,0),\ x\in\partial\Omega$$

- ✏️ 这个compatible condition很重要，也就是边界需要满足这个条件才是合理的

6.1 Time-Dependent Boundary Value Problems

Posed on space-time cylinder

Domain $\tilde{\Omega} = \Omega\times]0,T[$

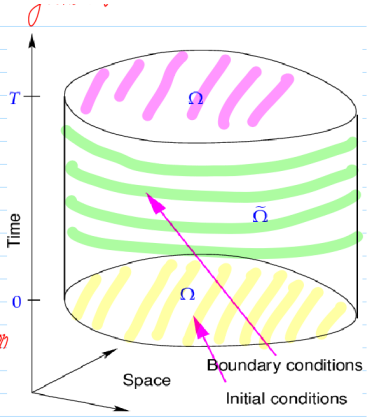
$\Omega \hat{=}$ spatial domain

$T \hat{=}$ final time, $T>0$

solution : $u = u(x,t)$

$x\in\Omega,\ 0\le t\le T$

- $\Omega\times\{0\}$ → initial conditions
- $\partial\Omega\times]0,T[$ → (spatial) boundary condition
- NO "final conditions"



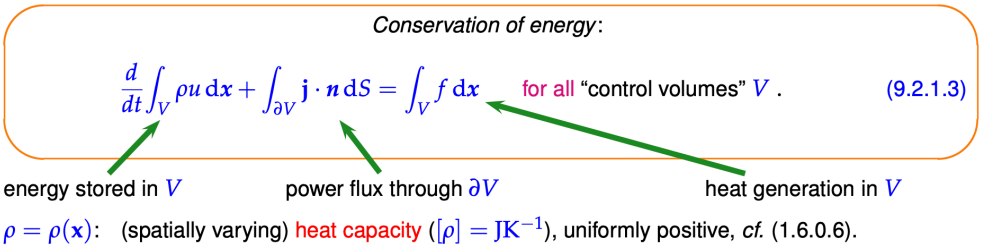
- Special about time ⇒ has a direction [causality]

6.2 Parabolic Initial Boundary Value Problems

dissipative evolutions → parabolic

6.2.1 Heat Equation

Compared with the stationary heat equation, the transient heat conduction's energy balance has to be supplemented by a storage term reflecting the fact that heat can accumulate:



By applying Gauss's Theorem, we can derived the local form of energy balance law:

$$\frac{\partial}{\partial t}(\rho u)(\boldsymbol{x},t) + (\mathrm{div}_{\boldsymbol{x}}\,\mathbf{j})(\boldsymbol{x},t) = f(\boldsymbol{x},t) \quad \text{in} \quad \widetilde{\Omega}\,.$$

Use the Fourier's law to express the flux $\mathbf{j}(\boldsymbol{x}) = -\kappa(\boldsymbol{x})\,\mathbf{grad}\,u(\boldsymbol{x})\,,\quad \boldsymbol{x}\in\Omega$. We can derive the local form of the energy balance law (heat equation)

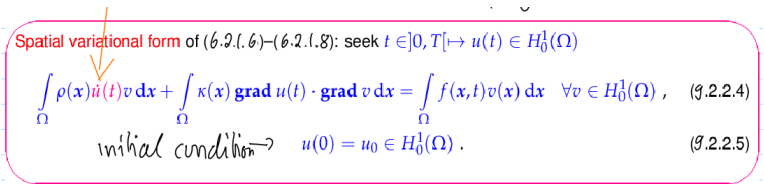
$$\frac{\partial}{\partial t}(\rho u) - \mathrm{div}(\kappa(\boldsymbol{x})\,\mathbf{grad}\,u) = f \quad \text{in} \quad \widetilde{\Omega} := \Omega\times]0,T[$$

- In orde to well-posed this equaiton, we need to supplement the boundary conditions (same as the elliptic BC) and the initial condition. Finally, we derive the 2nd-order **parabolic** initial-boundary value problem (IBVP).

> For second order parabolic evolutions we can/must use the *same* spatial boundary conditions as for stationary second order elliptic boundary value problems.

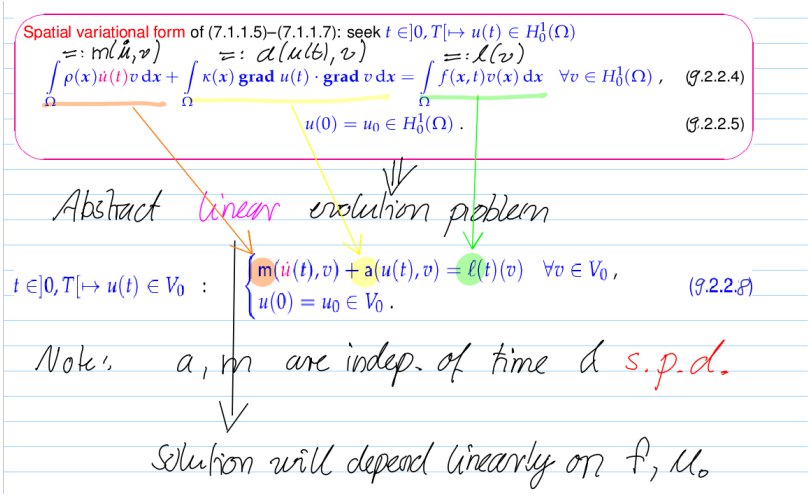
6.2.2 Heat Equation: Spatial Variational Formulation

核心思路就是我们将t treated as "passive parameter"和考虑一些简单的setup（$u(x,t)=0$, heat conductivity/ campacity independent of t)从而整个问题在空间上简化成了spatial dirichlet BVP (homogeneous).然后我们再将整个问题转化成variational formulation。（其思路就是先乘test function - not depend on time，再做分布积分，再chose sobolev space framework使得energy norm make sense并且最大），这里需要注意的是test function不能在解存在的地方有数值。



- 整体上感觉来说就是第一项那一热容项多了出来而已，其他的都与elliptic boundary condition完全一样。
- ⚠，这里的$u:]0,T[$ function-space-value function of time ["ODE in function space"] 之前的elliptic最后$u(t)$与时间无关，只是一个空间函数。而这里，我们将空间场看成一个function space，从而变成了function space的ODE，也就是PDE。

我们进而将这个问题抽象出来，就变成了abstract linear evolution problem（之所以称linear，因为solution will depend linearly on f, u_0):



- 随着时间的evolution，a, m 都不变，并且他们都是s.p.d.两个bilinear form。

6.2.3 Stability of Parabolic Evolution Problems

> "Stability" asks the question whether for a mathematical model small changes in the input data cause only small changes of the solution. This is a central concern, because data will usually have been obtained by measurement or some approximation and are inevitably tainted by "small" errors. If those triggered massive changes of the solution, the latter would not be useful.

首先研究这个问题的前提是我们要明白为什么我们要研究stability。因为像$l(f), u_0$这些不是材料固有属性，需要我们在求解方程需要依据这些data。而这些data通常会存在loss，解对这些loss是否robotness是我们要考虑的对象。这个其实也可以迁移到fourier neural operator。其实他想要拟合的operator就是f和u_0与输出u之间的关联。此外，stability研究也可以让我们研究解的structural properties(这个概念不是很懂)

Definition 1.3.2.3. Space $L^2(\Omega)$ → Def. 0.3.2.27

The function space defined in (1.3.2.2) is the space of square-integrable functions on Ω and denoted by $L^2(\Omega)$.

It is a normed space with norm $\left(\|v\|_0 :=\right) \|v\|_{L^2(\Omega)} := \left(\int_\Omega |v(x)|^2\,dx\right)^{1/2}$.

- 注意norm是要加根号的。

This reduces the problem to bounding $\|u\|_0$ in terms of $|u|_{H^1(\Omega)}$.

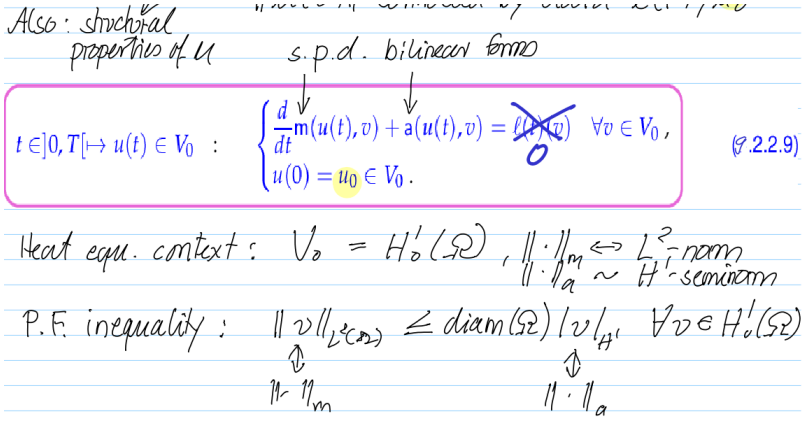
- **Theorem 1.3.4.17. First Poincaré-Friedrichs inequality**

  If $\Omega \subset \mathbb{R}^d$, $d \in \mathbb{N}$, is bounded, *then*

  $$\|u\|_0 \le \operatorname{diam}(\Omega)\,\|\mathbf{grad}\,u\|_0 \quad \forall u \in H_0^1(\Omega).$$

Data: u_0, f

我们在这里考虑一个简化的事例（source term=0，唯一的data即使u_0）



最后得出的结论是解是被u_0很好的限制在，并且这种扰动在时间的演化过程中也会被逐渐的缩写以至于0。

ChatGPT: 看这个解释

Lemma 9.2.3.8. Decay of solutions of parabolic evolutions

For $f \equiv 0$ the solution $u(t)$ of (9.2.2.4) satisfies

$$\|u(t)\|_m \le e^{-\gamma t}\|u_0\|_m \quad , \quad \|u(t)\|_a \le e^{-\gamma t}\|u_0\|_a \quad \forall t \in]0,T[\,,$$

where $\gamma > 0$ is the constant from (9.2.3.6), and $\|\cdot\|_a$, $\|\cdot\|_m$ stand for the energy norms induced by $\mathsf{a}(\cdot,\cdot)$ and $\mathsf{m}(\cdot,\cdot)$, respectively.

- 注意L^1, which is give by $\int_{-\infty}^{\infty}|u(x,t)|dx$, represents the "total mass" of the function keep the same as time goes on.

这个结论同时也说明了我们的解在function space中的norm也是在逐渐减少的。即energy norm是逐渐递减的。

- 这里refer to的是unbounded domain. 对于bounded domain，energy norm还是保持一致的对于zero Neumann BC.

Dissipation of energy in parabolic evolutions

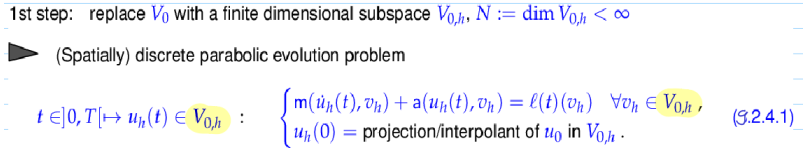
Exponential decay of energy during parabolic evolution without excitation
("Parabolic evolutions dissipate energy")

- 这个地方我们采用了一个axuliary function来完成推导，并且同时推导了在两个不同norm（metric度量）下的bound。具体推导过程有点不懂，再看看研究一下

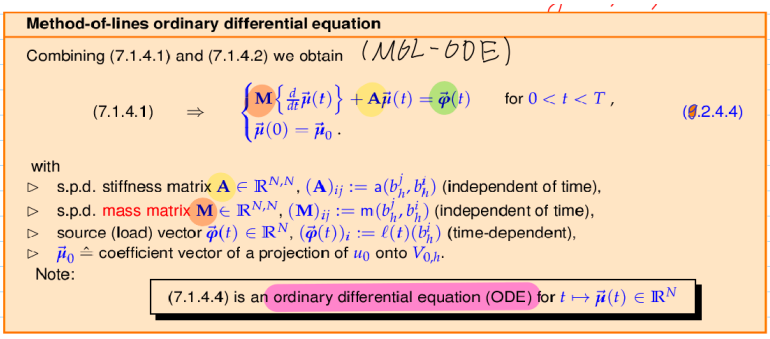
6.2.4 Spatial Semi-Discretization: Method of Lines

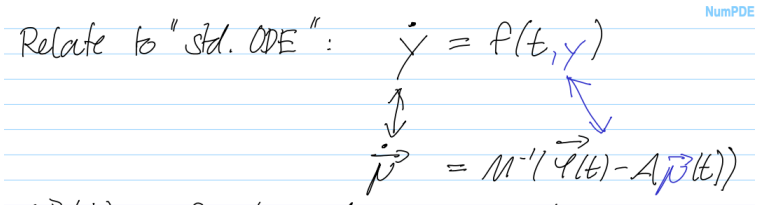
之前我们说解是一个ODE在function space上面，而这样的连续性问题计算机是不能够求解的，因为我们这里先考虑把空间离散化。对于evolution problem，这样的离散称为spatial semi-discretization:

具体步骤就是先做spatial galerkin discretization，replace V_0 with $V_{0,h}$ ($\dim V_0 = N < \infty$, independent of time) 其实这个地方也可以考虑随着时间变化，离散格式相应发生变化。



接着，$u_h(t)$: a function-space valued function of time. 我们选取这个离散空间的basis，把问题转化成求基函数前面的系数。在这里，我们的离散空间不随时间变化的话，那么我们的基也就不随时间变化，从而我们的基函数前面的系数应该是t-dependent coefficients。通过使用bi-linearity of m&a,我们得到如下形式：



- 

- $\mu(t)$ a function of continuous time: MOL-ODE is semi-discrete. Multi-variate ODEs

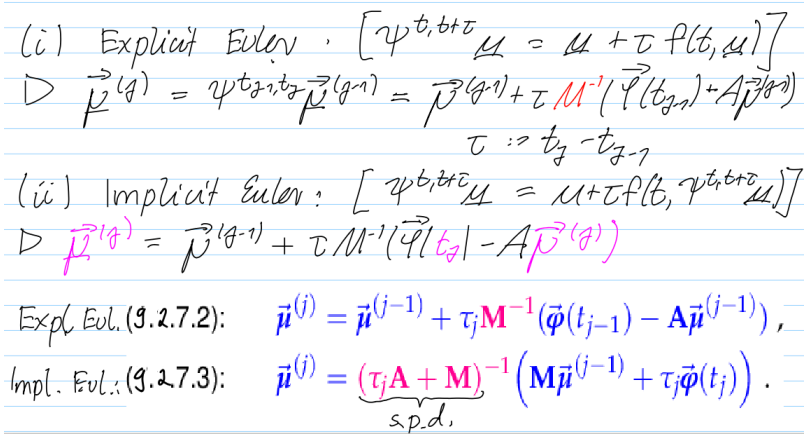
6.2.5 Timestepping for Method-of-Lines ODE

From the method of lines (MOL) presented in Section 9.2.4 we get only a semi-discrete problem in the form of an ODE. However, for implementation we need a fully discrete evolution problem. This requires additional discretization in time:

semi-discrete evolution problem (last chapter) + timestepping ⇒ fully discrete evolution problem

- A good thing is that we can apply the already known integrators for IVP for ODEs to semi-discrete evolution problem.

首先，我们先像一般的ODE那样，把方程化成标准形式。接着我们讨论explicit euler, implicit euler以及Runge-Kutta SSM:



- 其实这样来看，隐式也就是可以直接迭代计算的。但实际中，implicit euler右端还是有$\vec{\mu}^{(j)}$，所以我们得求解一个LSE。

Definition 7.3.3. 1. General Runge-Kutta method →

For coefficients $b_i, a_{ij} \in \mathbb{R}$, $c_i := \sum_{j=1}^{s} a_{ij}$, $i,j = 1,\dots,s$, $s \in \mathbb{N}$, the discrete evolution $\Psi^{s,t}$ of an s-stage Runge-Kutta single step method (RK-SSM) for the ODE $\dot{u} = \mathbf{f}(t,\mathbf{u})$, is defined by

$$\mathbf{k}_i := \mathbf{f}(t + c_i\tau, \mathbf{u} + \tau \sum_{j=1}^{s} a_{ij}\mathbf{k}_j)\,, \quad i = 1,\dots,s \quad , \quad \Psi^{t,t+\tau}\mathbf{u} := \mathbf{u} + \tau \sum_{i=1}^{s} b_i \mathbf{k}_i\,.$$

The $\mathbf{k}_i \in V_0$ are called increments. Implicit increment equations

$\vec{\kappa}_i \in \mathbb{R}^N$

$$\vec{\kappa}_i = M^{-1}\left(\vec{q}(t+c_i\tau) - A\left(\vec{\mu}^{(j)} + \tau \sum_{j=1}^{s} a_{ij}\vec{\kappa}_j\right)\right)$$

$$\vec{\kappa}_i \in \mathbb{R}^N: \quad \mathbf{M}\vec{\kappa}_i + \sum_{m=1}^{s} \tau a_{im}\mathbf{A}\vec{\kappa}_m = \vec{\varphi}(t_j + c_i\tau) - \mathbf{A}\vec{\mu}^{(j)}\,, \quad i = 1,\dots,s\,, \tag{9.2.7.7}$$

$$\vec{\mu}^{(j+1)} = \vec{\mu}^{(j)} + \tau \sum_{m=1}^{s} \vec{\kappa}_m b_m\,. \tag{9.2.7.8}$$

$\hat{=}$ $N \cdot s \times N \cdot s$ linear system of equations

Compact notation via Kronecker product

$$(\,.\,.7.7) \;\Leftrightarrow\; \underbrace{(\mathbf{I}_s \otimes \mathbf{M} + \tau \mathfrak{A} \otimes \mathbf{A})}_{\in \mathbb{R}^{Ns,Ns}} \begin{bmatrix} \vec{\kappa}_1 \\ \vdots \\ \vec{\kappa}_s \end{bmatrix} = \begin{bmatrix} \vec{\varphi}(t_j + c_1\tau) - \mathbf{A}\vec{\mu}^{(j)} \\ \vdots \\ \vec{\varphi}(t_j + c_s\tau) - \mathbf{A}\vec{\mu}^{(j)} \end{bmatrix}. \tag{9.2.7.9}$$

- 至于RK-ssm，由于这里是隐式的格式，也就是不能通过前面的increments推出后面的increment，而是需要求解一下LSE才能推出所有的increments。关于这部分的RK之后再复习一下。

接着，又介绍了采用这样的数值格式去解这个PDE，系统稳定性如何。通过一些数值试验来研究。而误差是通过如下式子来度量：$error^2 = \tau h \sum_{i=1}^{N}\sum_{j=1}^{M} |(\mu_h - \mu)(h_i, j\tau)|^2$ $L^2 - norm$

我们发现，对于explicit Euler timestepping. 如果空间离散很多，那么时间离散得更多才能保证系统稳定。即blow-up for small h_M, large t.这里有一个列表，放上来有助于加深理解。空间

Space-time (discrete) L^2-norm of error for explicit Euler timestepping:

$N \backslash M$	50	100	200	400	800	1600	3200
5	Inf	0.009479	0.006523	0.005080	0.004366	0.004011	0.003834
10	Inf	Inf	Inf	Inf	0.001623	0.001272	0.001097
20	Inf	Inf	Inf	Inf	Inf	Inf	0.000405
40	Inf	Inf	Inf	Inf	Inf	Inf	Inf
80	Inf	Inf	Inf	Inf	Inf	Inf	Inf
160	Inf	Inf	Inf	Inf	Inf	Inf	Inf
320	Inf	Inf	Inf	Inf	Inf	Inf	Inf

Here Inf indicates that the method suffered an exponential blow-up.

- Where meshwidth $h := \frac{1}{N+1}$, timestep $\tau := \frac{1}{M}$ (1D heat equation).

而对于implicit Euler timestepping: unconditional stability

Space-time (discrete) L^2-norm of error for implicit Euler timestepping:

$N\backslash M$	50	100	200	400	800	1600	3200
5	0.007025	0.001828	0.000876	0.002257	0.002955	0.003306	0.003482
10	0.009641	0.004500	0.001826	0.000461	0.000228	0.000575	0.000749
20	0.010303	0.005175	0.002509	0.001149	0.000461	0.000116	0.000058
40	0.010469	0.005345	0.002681	0.001321	0.000634	0.000289	0.000116
80	0.010511	0.005387	0.002724	0.001364	0.000677	0.000332	0.000159
160	0.010521	0.005398	0.002734	0.001375	0.000688	0.000343	0.000170
320	0.010524	0.005400	0.002737	0.001378	0.000691	0.000346	0.000172

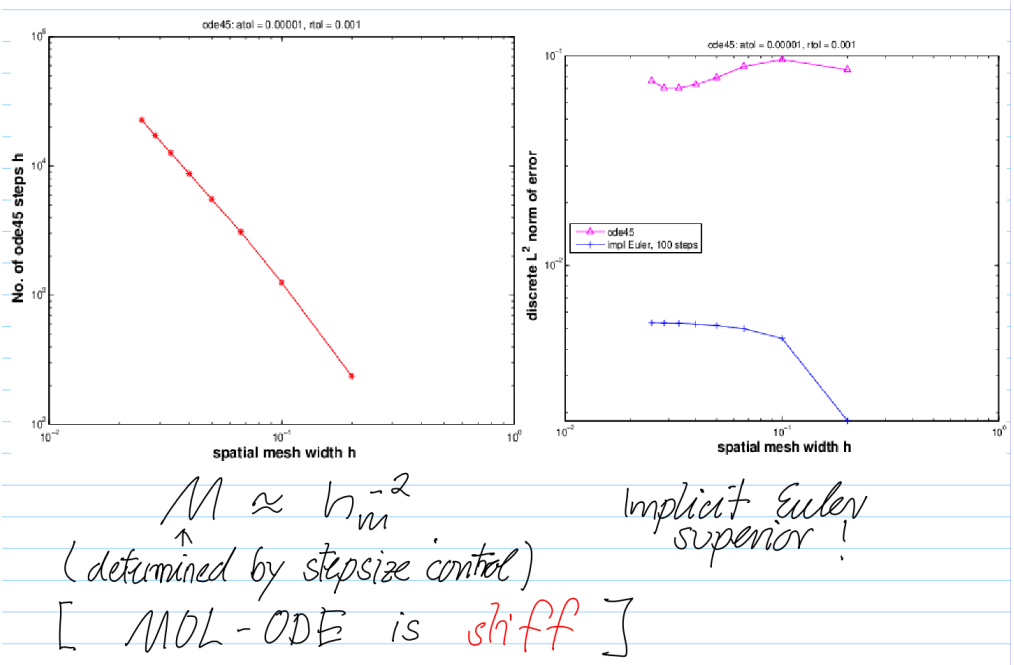
我们有如下的观察：

For explicit Euler timestepping we observe a glaring instability (exponential blow-up) in case of *large timestep combined with fine mesh*.

Implicit Euler timestepping incurs no blow-up for any combination of spatial and temporal mesh width.

因此我们想问：this is just a shortcoming of expl. Euler?（这是显示方法的唯一缺点吗？）

然后我们又看了另一个数值试验，就是我们用Adaptive explicit RK-SSM 4 (stage 4, order 5)



- 在这里，我们遵循的adaptive 原则是总的时间步离散是空间步网格的平方的倒数（这可以确保我们的integrator not blow-up），这里ODE45里面自带的步长自适应。但是，我们发现，和implicit的方法，增加spatial mesh并不能让我们获得更好的精度！即使我们用了更多的时间和空间网格（这真是离谱）。而这种现象被称为stiffness，MOL-ODE is a stiff problem)

接下来我们想要分析这个问题，从而得到更稳定的算法，我们分析的工具是Diagonalization (general conclusion: symmetric matrix can be diagonalized.)。diagonalization可以让我们的ODE decoupling (如果线性的话应该，应为这里考虑的Heat Equation是线性问题，我们也不知道非线性是啥情况)

$\dot{u} = Au$, $A \in \mathbb{R}^{N,N}$ diagonalizable

$AT = TD$, $D = \mathrm{diag}(\lambda_1, \ldots, \lambda_N)$

$w = T^{-1}u \Rightarrow \dot{w} = Dw$: decoupling

$\Leftrightarrow w_i = \lambda_i w_i$ ⊣

而对于我们这种evolution problem，我们引入了generalized eigenvectors和eigenvalues，同样decoupled into scalar ODEs.这里是将solution投影到eigenvector这些basis上，从个人得到decouped equation

s.p.d. matrices $\in \mathbb{R}^{N,N}$

$$\mathbf{M}\left\{\frac{d}{dt}\vec{\mu}(t)\right\} + \mathbf{A}\vec{\mu}(t) = \vec{\varphi}(t)\,. \tag{9.2.4.4}$$

Let $\vec{\psi}_1, \ldots, \vec{\psi}_N \in \mathbb{R}^N$ denote the N linearly independent generalized eigenvectors satisfying

$$\mathbf{A}\vec{\psi}_i = \lambda_i \mathbf{M}\vec{\psi}_i \quad , \quad \vec{\psi}_j^{\top}\mathbf{M}\vec{\psi}_i = \delta_{ij}\,, \quad 1 \le i,j \le N\,. \tag{9.2.7.16}$$

with positive eigenvalues $\lambda_i > 0$. Introducing the regular square matrices

$$\mathbf{T} = [\vec{\psi}_1, \ldots, \vec{\psi}_N] \in \mathbb{R}^{N,N}\,, \tag{9.2.7.17}$$

$$\mathbf{D} := \mathrm{diag}(\lambda_1, \ldots, \lambda_N) \in \mathbb{R}^{N,N}\,, \tag{9.2.7.18}$$

we can rewrite (7.1.7.18) as

$$\mathbf{AT} = \mathbf{MTD} \quad , \quad \mathbf{T}^{\top}\mathbf{MT} = \mathbf{I}\,. \tag{9.2.7.19}$$

$\to (MT)^{-1} = T^{\top}$

New coefficient vector: $\vec{\eta}(t) = T^{-1}\vec{\mu}(t)$

MOL-ODE $\Rightarrow$ $T^{\top}\cdot\,|\, MT\dot{\vec{\eta}} + AT\vec{\eta} = \vec{\varphi}$

然后我们将其带入到explicit euler method中，我们发现，integrator的稳定性取决于lambda的大小，从而又间接影响了我们的timestep size的control。

Diagonalizing explicit Euler method ($\vec{\varphi} \equiv 0$)

$$\vec{\mu}^{(j)} = \vec{\mu}^{(j-1)} - \tau\mathbf{M}^{-1}\mathbf{A}\vec{\mu}^{(j-1)} \quad \xrightarrow{\vec{\eta} := \mathbf{T}^{\top}\mathbf{M}\vec{\mu}} \quad \vec{\eta}^{(j)} = \vec{\eta}^{(j-1)} - \tau\mathbf{D}\vec{\eta}^{(j-1)}\,,$$

$\Updownarrow$

$\eta_i^{(j)} = \eta_i^{(j-1)} - \tau\lambda_i\eta_i^{(j-1)}$

$$\eta_i^{(j)} = \eta_i^{(j-1)} - \tau\lambda_i\eta_i^{(j-1)} \;\Rightarrow\; \boxed{\eta_i^{(j)} = (1-\tau\lambda_i)^j\eta_i^{(0)}}\,. \tag{9.2.7.24}$$

$$|1-\tau\lambda_i| < 1 \;\Leftrightarrow\; \lim_{j\to\infty}\eta_i^{(j)} = 0\,.$$

$|1-\tau\lambda_i| > 1 \Rightarrow$ blow-up

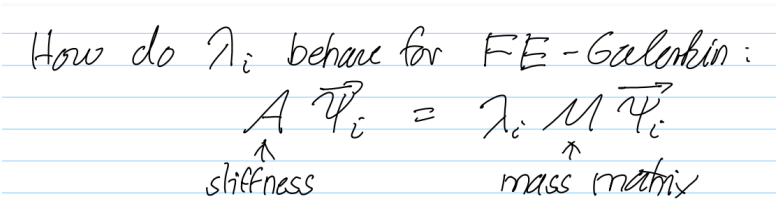
The condition $|1-\tau\lambda_i| < 1$ enforces a

timestep size constraint: $\tau < \frac{2}{\lambda_i}$ $\forall i$

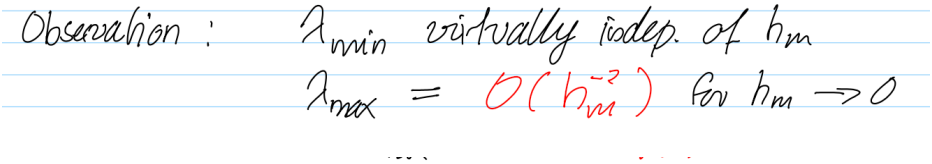
If $\exists \lambda_i : |1-\tau\lambda_i| > 1 \Rightarrow$ Blow-up of $\vec{\mu}^{(j)}$

- 这里我们为什么只考虑$1-\tau\lambda_i>0$会爆炸而不小于小于0会vanishing是因为我们知道我们的解是满足structural properties，也就是我们的解是指数衰减的，所以小于0是合理情况。
- 此外为什么不考虑时而大于0，时而小于0这种情况是因为那是另一种稳定性讨论，我们称之为A-stability。这里讨论的是L-stability。

而λ的值取决于我们的stiffness matrix A和mass matrix M：



但是实际情况中我们肯定不知道λ的，因为我们又做了个实验。用来monitoring generalized eigenvalue的变化当我们refined our mesh。从而我们观察到



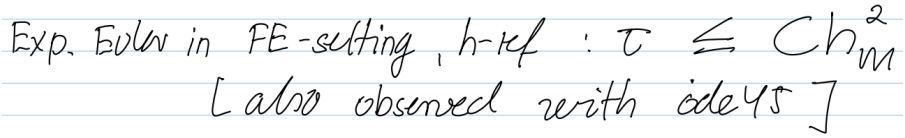
Lemma 3.2.7.30. Behavior of of generalized eigenvalues

Let $\mathcal{M}$ be a simplicial mesh and $\mathbf{A}$, $\mathbf{M}$ denote the Galerkin matrices for the bilinear forms $a(u,v) = \int_\Omega \mathbf{grad}\, u \cdot \mathbf{grad}\, v \,\mathrm{d}\boldsymbol{x}$ and $m(u,v) = \int_\Omega u(x)v(x)\,\mathrm{d}\boldsymbol{x}$, respectively, and $V_{0,h} := \mathcal{S}^0_{p,0}(\mathcal{M})$. Then the smallest and largest generalized eigenvalues of $\mathbf{A}\vec{\mu} = \lambda\mathbf{M}\vec{\mu}$, denoted by $\lambda_{\min}$ and $\lambda_{\max}$, satisfy

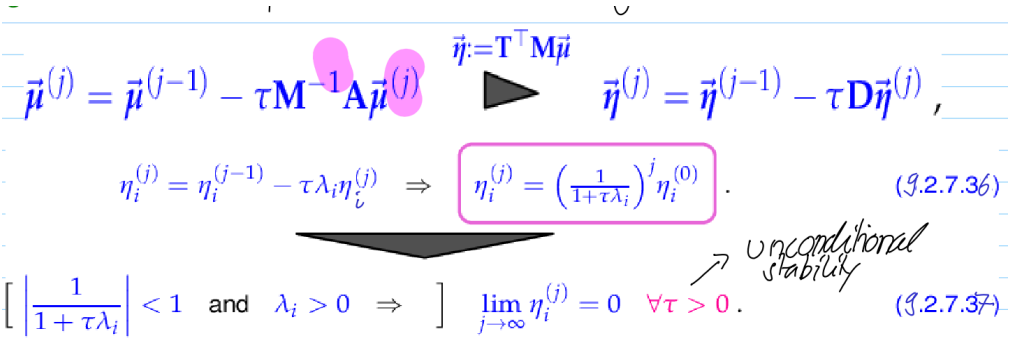
$$\frac{1}{\operatorname{diam}(\Omega)^2} \le \lambda_{\min} \le C \quad , \quad \lambda_{\max} \ge Ch_{\mathcal{M}}^{-2},$$

where the "generic constants" (→ Rem. 3.3.5.10) depend only on the polybomial degree p, the domain Ω, and the shape regularity measure $\rho_{\mathcal{M}}$.

因此在显示欧拉格式中，采用h-ref，如果我们不想发生爆炸，我们采用的约束即为：

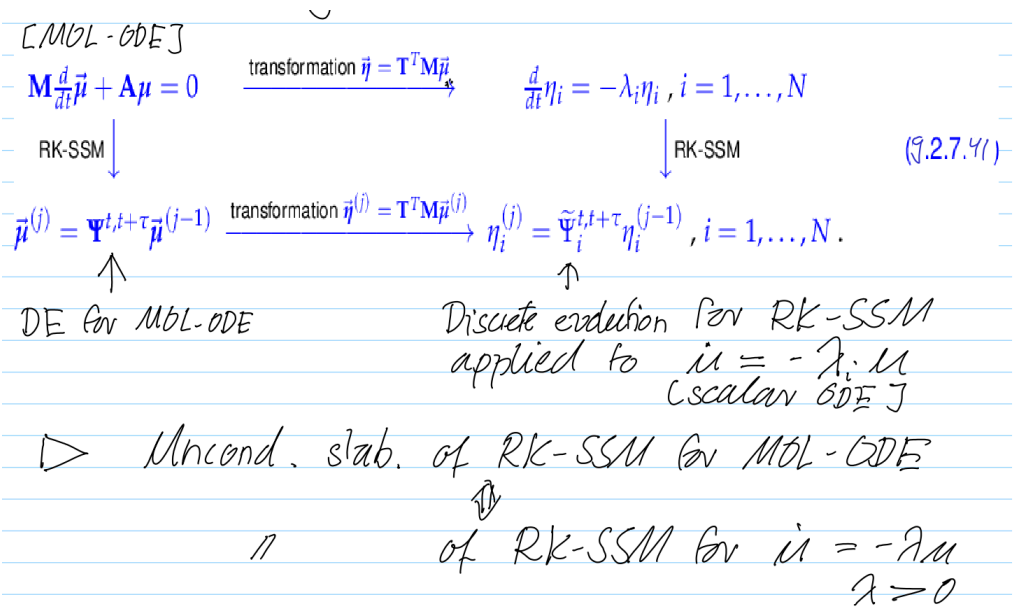


下面，对于Implicit Euler,我们采用diagonalization来看看：



然后我们发现是unconditional stability.

接着我们对general RK使用来看看RK满足什么形式时可以保证不爆炸。并且，我们将整个问题写成了更general的结论：



Unconditional stability of single step methods

Necessary condition for *universal unconditional stability* of a single step method for semi-discrete parabolic evolution problem (7.1.4.4)("method of lines"):

The discrete evolution $\Psi_\lambda^\tau : \mathbb{R} \mapsto \mathbb{R}$ of the single step method applied to the scalar ODE $\dot{u} = -\lambda u$ satisfies

$$\lambda > 0 \quad \Rightarrow \quad \lim_{j\to\infty} (\Psi_\lambda^\tau)^j u_0 = 0 \quad \forall u_0 \in \mathbb{R}, \quad \forall \tau > 0\,. \tag{9.2.7.43}$$

对于RK来说，我们可以用Scheme来写出演化过程中的stability function

RK-SSM for $\dot{u} = -\lambda u \Rightarrow u^{(y)} = S(-\lambda\tau)u^{(y-1)}$

$\begin{array}{c|c} c & A \\ \hline & b^T \end{array}$

↑ Stability functions

$$\Psi_\lambda^{t,t+\tau} u = S(-\lambda\tau)u\,, \qquad (6.2.7.47a)$$

with rational stability function

$$S(z) = 1 + zb^T(\mathbf{I} - z\mathfrak{A})^{-1}\mathbf{1} = \frac{\det(\mathbf{I} - z\mathfrak{A} + z\mathbf{1}b^T)}{\det(\mathbf{I} - z\mathfrak{A})}\,, \quad \mathbf{1} = [1,\ldots,1]^T \in \mathbb{R}^s\,. \qquad (6.2.7.47b)$$

↳ a rational function in z

RK-SSM uncond. stable for MOL-ODE

if $|S(z)| \leq 1 \quad \forall z < 0$

Even desirable $|S(z)| \ll 1$ for $z < 0$, $|z| \gg 1$

[fast exponential decay]

Definition 6.2.7.46. L(π)-stability

A Runge-Kutta single-step method satisfying (7.1.7.45) is called L(π)-stable, if its stability function $S(z)$ according to (7.1.7.47b) satisfies

(i) $|S(z)| < 1$ for all $z < 0$, and

(ii) "$S(-\infty)$" $:= \lim_{z\in\mathbb{R}\to-\infty} S(z) = 0$.

- 这里介绍了$L(\pi)$-stability。
- 并且给出了常用的scheme，并且这些scheme都必须是隐式：

$$\begin{array}{c|cc} \frac{1}{3} & \frac{5}{12} & -\frac{1}{12} \\ 1 & \frac{3}{4} & \frac{1}{4} \\ \hline & \frac{3}{4} & \frac{1}{4} \end{array} \qquad (6.2.7.52)$$

RADAU-3 scheme (order 3)

$$\begin{array}{c|cc} \lambda & \lambda & 0 \\ 1 & 1-\lambda & \lambda \\ \hline & 1-\lambda & \lambda \end{array}\,, \quad \lambda := 1 - \frac{1}{2}\sqrt{2}\,, \qquad (6.2.7.53)$$

SDIRK-2 scheme (order 2)

Note: Necessarily implicit

6.2.6 Fully Discrete Method of Lines: Convergence

现在，我们综合前面几章节的内容，考虑和研究fully discreted PDE。问题的引出通过下面：

Definition 7.3.3.1. General Runge-Kutta method →

For coefficients $b_i, a_{ij} \in \mathbb{R}$, $c_i := \sum_{j=1}^{s} a_{ij}$, $i,j = 1,\ldots,s$, $s \in \mathbb{N}$, the discrete evolution $\Psi^{s,t}$ of an s-stage Runge-Kutta single step method (RK-SSM) for the ODE $\dot{u} = \mathbf{f}(t,\mathbf{u})$, is defined by

$$\mathbf{k}_i := \mathbf{f}(t + c_i\tau, \mathbf{u} + \tau \sum_{j=1}^{s} a_{ij}\mathbf{k}_j)\,, \quad i = 1,\ldots,s \quad , \quad \Psi^{t,t+\tau}\mathbf{u} := \mathbf{u} + \tau \sum_{i=1}^{s} b_i \mathbf{k}_i\,.$$

The $\mathbf{k}_i \in V_0$ are called increments.

[Asymptotic alg. cvg. with order q for $\tau \to 0$]

- Explicit RK-SSM for MOL-ODE & fixed-degree Lagr. FE
  ▷ Stability-induced timestep constraint $\tau \le Ch_m^2$
- $L(\pi)$-stable RK-SSM : unconditionally stable
  ↳ Use these!

unknown constant

- 这里要首先知道这些integrator都是有order的，阶数是q。而当ODE45不用在stiffiness problem上面时，其可以实现algebric convergence with order 5.
- 为什么我们要考虑$L(\pi)$-stable RK-SSM?
  - 第一个理由是stability-induced timestep control中，C是不知道的（unknown stability bound!)。虽然我们可以通过做一系列的数值试验得到，但这是postori的，没有$L(\pi)$这个priori那么好用。
  - 第二个理由是convergence theory，也就是在h-refinement setting中，我们有两个parameters，我们想知道error随着这两个parameters (spatial step and temporal step) 如何变化。但是这里要注意的是我们并不能得出error的具体数值，但是我们可以给出error的trend，即asymptotic perspective，这是很有吸引力的。在下面的部分内容中，我们重点讨论的就是这个内容

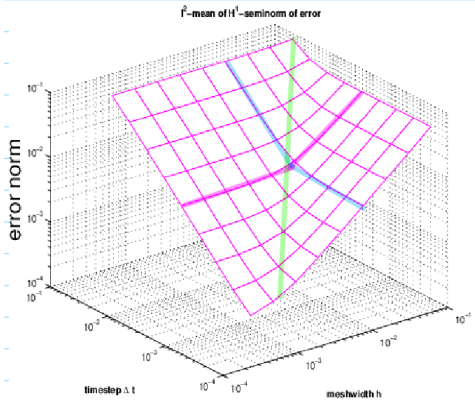
我们做了个数值试验，在这个实验中，我们的norm（度量是这样定义的，空间维度上我们用的H^1 norm，时间维度上我们用的L^2 norm：

- 1D parabolic evolution problem: $\frac{d}{dt}u - u'' = f(t,x)$ on $]0,1[\times]0,1[$
- exact solution $u(x,t) = (1+t^2)e^{-\pi^2 t}\sin(\pi x)$, source term accordingly
- Linear finite element Galerkin discretization equidistant mesh, see Section 2.3, $V_{0,h} = \mathcal{S}^0_{1,0}(\mathcal{M})$,
- piecewise linear spatial approximation of source term $f(x,t)$
- implicit Euler timestepping (→ Ex. 7.1.7.1) with uniform timestep $\tau > 0$

Monitored: ↓ uncond. stable 1-st order

error norm $\left(\tau \sum_{j=1}^{M} |u - u_h(\tau j)|^2_{H^1(\Omega)}\right)^{\frac{1}{2}}$: L^2 in time, H^1 in space

通过数值试验，我们发现一个接近线性的行为：



◁ $h_{\mathcal{M}}$- and τ-dependence of error norm

Obervation:

τ small: error norm $\approx h_{\mathcal{M}}$
$h_{\mathcal{M}}$ small: error norm $\approx \tau$

} alg. cvg. rate 1

The error seems to behave like

$$\text{error norm} \approx C_1 h_{\mathcal{M}} + C_2 \tau \,. \qquad (9.2.8.2)$$

total error = spatial error + temporal error

- 注意，在这个问题中，我们用的1-p order的integrator，和1-q的local shape function。

因此，我们可以得到下面的meta-theorem： ！！很重要

"Meta-theorem" 9.2.8.5. Convergence of solutions of fully discrete parabolic evolution problems

Assume that

- *the solution of the parabolic IBVP (7.1.1.5)–(7.1.1.8) is "sufficiently smooth" (both in space and time),*
- *its spatial Galerkin finite element discretization relies on degree p Lagrangian finite elements (→ Section 2.6) on uniformly shape-regular families of meshes,*
- *timestepping is based on an $L(\pi)$-stable single step method of order q with uniform timestep $\tau > 0$.*

Then we can expect an asymptotic behavior of the total discretization error according to

$$\left(\tau \sum_{j=1}^{M} |u - u_h(\tau j)|^2_{H^1(\Omega)}\right)^{\frac{1}{2}} \leq C\left(h_{\mathcal{M}}^p + \tau^q\right), \qquad (9.2.8.6)$$

where $C > 0$ must not depend on $h_{\mathcal{M}}, \tau$.

unknown generic constant　spatial error　temporal error

▷ Tells trend of the error for $h_M, \tau \to 0$

Assume: Bound sharp : spatial error $\approx C h_M^p$
temporal error $\approx C\tau^q$

接下来，我们我们像把时空的两个parameter归结为1个。因此，我们定义了下面的factor:

Goal: reduction of total error by a factor of $\varrho > 1$

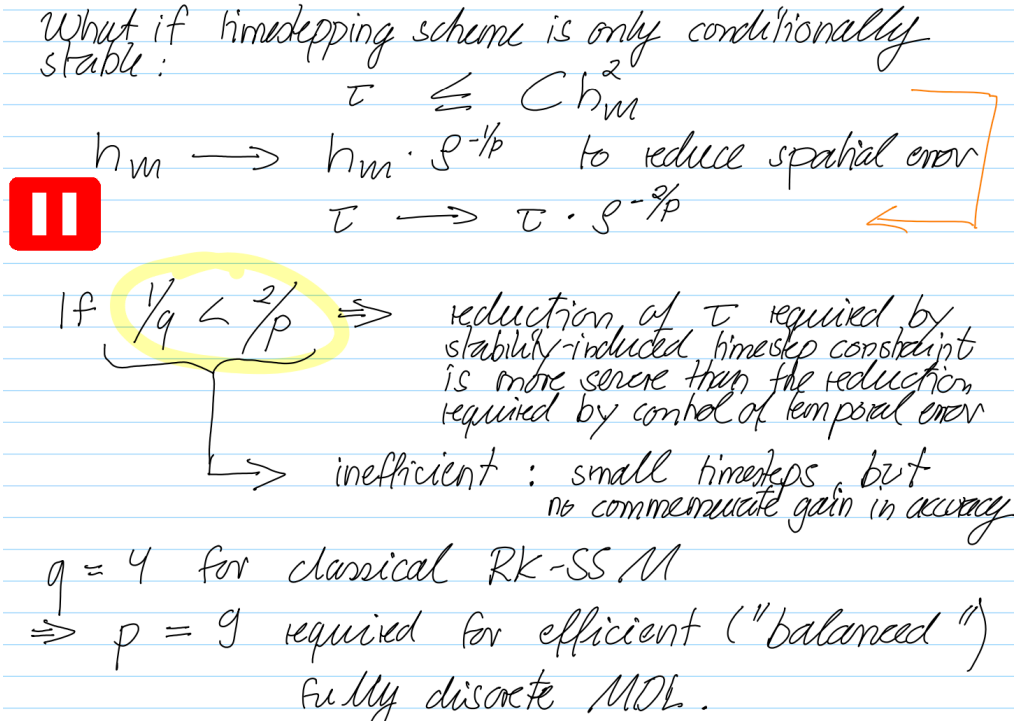
$h_M \to h_M \cdot \varrho^{-1/p}$
$\tau \to \tau \cdot \varrho^{-1/q}$

} : error equilibration principle

- 这个地方是将空间参数和时间参数用一个参数统一起来，并且，这样error反带回去，我们发现error是线性地随着ϱ的减小而变小，并且spatial error和空间的error两个始终保持再一个数量级，也就是不会某一种error占据主导，从而你改变另一个error对结果error几乎造不成变化。

Refinement for fully discrete parabolic evolution problems	
Guideline:	spatial and temporal resolution have to be adjusted in tandem

- 而对于explicit scheme，很有意思：



  - 这里就是如果你的空间阶数p很低的话，即使你有很大的时间阶数的integrator，你也无法获得任何error上的改进。因为这个时候，t满足稳定性的下降速度要远远大于空间。

- 这个地方理解有点困难，之后再仔细研究一下

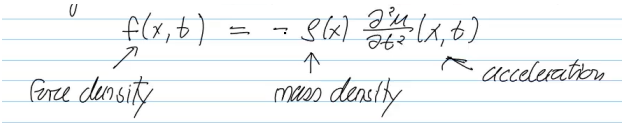
6.3 Linear Wave Equations

This section is dedicated to a class of initial-boundary value problems (IBVP) that have the same structure as (abstract) parabolic IBVP (→ § 6.2.2.7) except for the occurrence of *second derivatives in time*. This will have profound consequences as regards properties of solutions and choice of timestepping schemes.

A conservative evolution In the case of transient heat conduction Lemma 9.2.3.8 teaches that in the absence of time-dependent sources the rate of change of temperature will decay exponentially. (我们讨论stability的地方！)

Now we will encounter a class of evolution problems where temporal and spatial fluctuations will not be damped and will persist for good: *This will be the class of linear conservative wave propagation problems*

6.3.1 Models for vibrating membrane

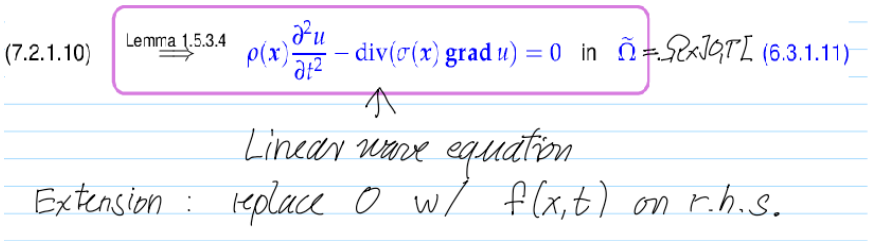


generaly, the physical quantities have unity. In order to simplify it, we rescale it and rescaled nondimensional equation.

- 这里的wave equation我们称hyperbolic PDE

这一章节我们讨论的是wave equation。我们从第一章的elliptic equation出发，将其中的$f(x)$替换成了如下：。从而得到了linear homogeneous linear wave equation.这里需要主要的是

这个地方挺迷惑的，本来这里的f已经是外加载力力，怎么就变成了材料自身的inertia force，此外，后面extension又多了一个外界载力



相比于the case of heat equation，在spatial boundary condtion上没有anything new，但是在initial conditions。我们需要两个initial condition，一个，这里可用动力系统的理论写成向量将其展开：

- Spatial boundary conditions: "nothing new"

On $\partial\Omega\times]0,T[$ we can impose any of the boundary conditions discussed in Section 1.7:
- Dirichlet boundary conditions $u(\boldsymbol{x},t) = g(\boldsymbol{x},t)$ (membrane attached to frame),
- Neumann boundary conditions $\mathbf{j}(\boldsymbol{x},t)\cdot\boldsymbol{n} = 0$ (free boundary, Ex. 1.5.3.11)
- radiation boundary conditions $\mathbf{j}(\boldsymbol{x},t)\cdot\boldsymbol{n} = \Psi(u(\boldsymbol{x},t))$,

and any combination of these as discussed in Ex. 1.7.0.10, yet, *only one* of them at any part of $\partial\Omega\times]0,T[$, see Rem. 1.7.0.9.

- Initial conditions

Wave equation = 2nd-order in time

[2nd-order ODE: $\ddot{u} = g(t,u)$

$[v := \dot{u}]$ ↑ velocity $\Leftrightarrow$ $\frac{d}{dt}\begin{bmatrix} u \\ v \end{bmatrix} = \begin{bmatrix} v \\ g(t,u) \end{bmatrix} = \begin{bmatrix} F(t,y) \end{bmatrix}$

(3)

▷ Initial condition : $u(0), v(0) = \dot{u}(0)$

For LWE : $u(x,0) = u_0(x)$

$\frac{\partial u}{\partial t}(x,0) = v_0(x)$

LWE → 1st-order in time system of PDEs

Additional unknown: velocity $v(\boldsymbol{x},t) = \frac{\partial u}{\partial t}(\boldsymbol{x},t)$

$\rho(x)\frac{\partial^2 u}{\partial t^2} - \operatorname{div}(\sigma(x)\,\mathbf{grad}\,u) = 0$ ⟹ $\begin{cases} \dot{u} = v\,, \\ \rho(x)\dot{v} = \operatorname{div}(\sigma(x)\,\mathbf{grad}\,u) \end{cases}$ in $\tilde{\Omega}$ (6.3.1.18)

with initial conditions

$u(\boldsymbol{x},0) = u_0(\boldsymbol{x})$, $v(\boldsymbol{x},0) = v_0(\boldsymbol{x})$ for $\boldsymbol{x}\in\Omega$. (6.3.1.19)

6.3.2 Wave propagation

Describe the dynamical behavior of elastic membrane.

Second-order equation in time

这里的问题毕竟是，你说这个问题是wave propagation，那么wave在哪呢？怎么体现呢？

这里，我们研究一个1D model cauchy problem（所有没有spatial BC都称cauchy）来看看其解的特点。首先，我们这里用了一下坐标替换的手段，从而化出了解的形式。然后这个问题其实具有统一解的形式，就是d' Alembat solution.我们进一步假设初始加速度为0。那么解可以看成是一个波分成两半以有限速度c（方程参数）向左右两边扩散。

$$\rho(x)\frac{\partial^2 u}{\partial t^2} - \operatorname{div}(\sigma(x)\,\mathbf{grad}\,u) = 0 \quad \text{in} \quad \tilde{\Omega} := \Omega\times]0,T[\tag{6.3.1.11}$$

\+ spatial boundary conditions & in initial conditions $\left\{\begin{array}{rcl} u(\boldsymbol{x},0) &=& u_0(\boldsymbol{x})\,, \\ \frac{\partial u}{\partial t}(\boldsymbol{x},0) &=& v_0(\boldsymbol{x})\,, \end{array}\right. \quad \boldsymbol{x}\in\Omega\,.$

1D model Cauchy problem: $\Omega = \mathbb{R}$ (no spatial b.c.)

$c>0$: $\frac{\partial^2 u}{\partial t^2} - c^2\frac{\partial^2 u}{\partial x^2} = 0$ (no forcing), $u(x,0)=u_0(x)$, $\frac{\partial u}{\partial t}(x,0)=v_0(x)$, $x\in\mathbb{R}$. (6.3.2.2)

(c: wave speed; initial data)

Trick: Use characteristic coordinates

$$\xi = x+ct\,,\quad \tau = x-ct: \qquad \tilde{u}(\xi,\tau) := u\left(\frac{\xi+\tau}{2},\frac{\xi-\tau}{2c}\right). \tag{6.3.2.3}$$

chain rule

$$\frac{\partial^2 u}{\partial t^2} - c^2\frac{\partial^2 u}{\partial x^2} = 0 \quad\Rightarrow\quad \frac{\partial^2 \tilde{u}}{\partial\xi\partial\tau} = 0 \quad\Rightarrow\quad \tilde{u}(\xi,\tau) = F(\xi)+G(\tau)\,,$$

for **any** $F, G\in C^2(\mathbb{R})$!

② $u(x,t) = F(x+ct) + G(x-ct)$

F, G from initial conditions

$u_0(x) = u(x,0) = F(x) + G(x)$

$v_0(x) = \frac{\partial u}{\partial t}(x,0) = c(F'(x) - G'(x))$ $\int_{-\infty}^{x}$

↑ compactly supported

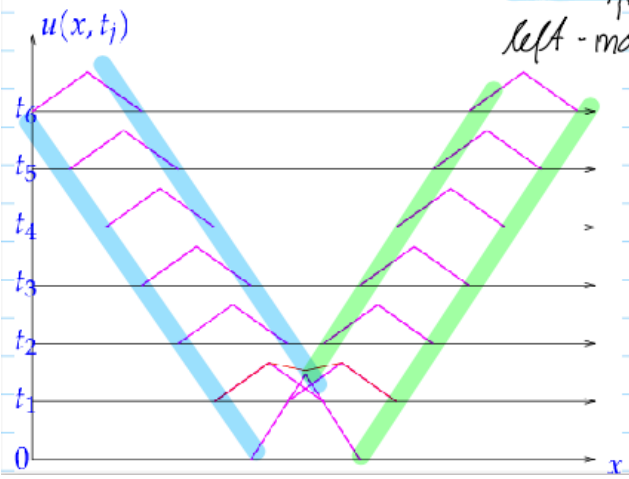
$$u(x,t) = \frac{1}{2}(u_0(x+ct) + u_0(x-ct)) + \frac{1}{2c}\int_{x-ct}^{x+ct} v_0(s)\,ds \,. \qquad (6.3.2.4)$$

[d'Alembert solution]

Special case: $v_0 \equiv 0$, u_0 compactly supported

$u(x,t) = \frac{1}{2}(u_0(x+ct) + u_0(x-ct))$

left-moving right-moving with speed $c > 0$



⇓

finite speed c of propagation

(compact supports preserved)

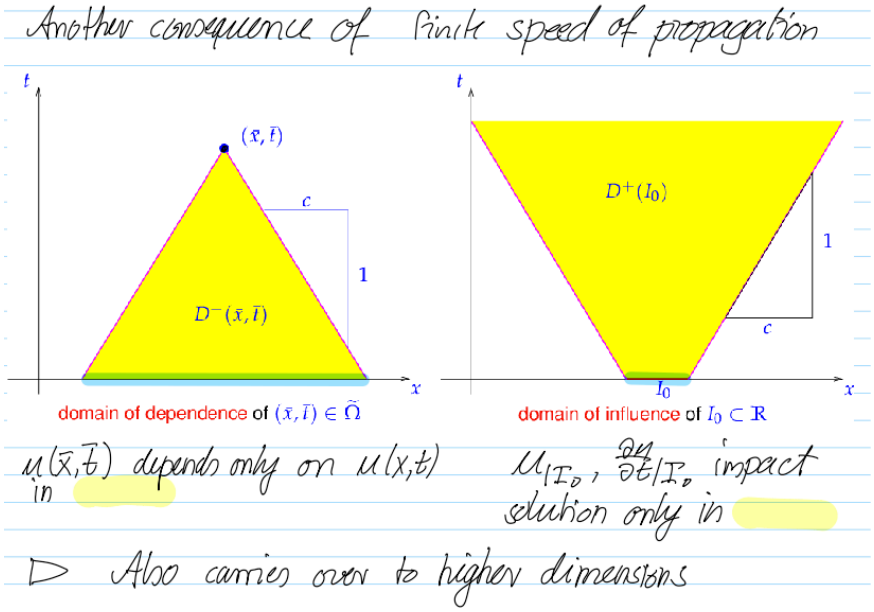
在扩散的过程中，速度是有限的，这和heat equation不同。heat equation扩散的过程中速度是无限的。

NumPDE@ETHZ

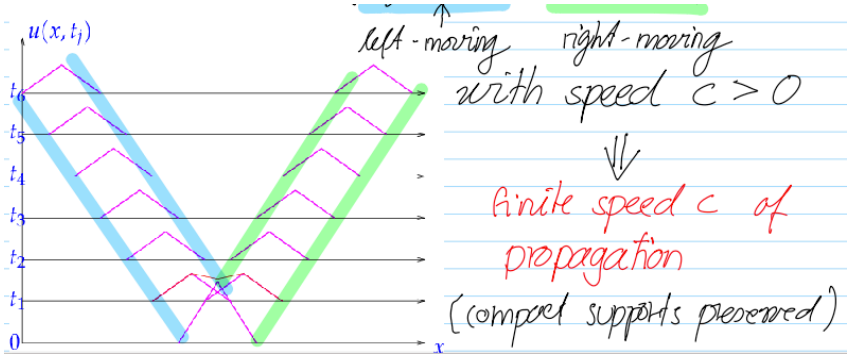
Remark: For 1D heat equation, Cauchy problem

Even if $\operatorname{supp} u_0$ bounded $\Rightarrow$ $\operatorname{supp} u(\cdot,t) = \mathbb{R}$ $\forall t > 0$

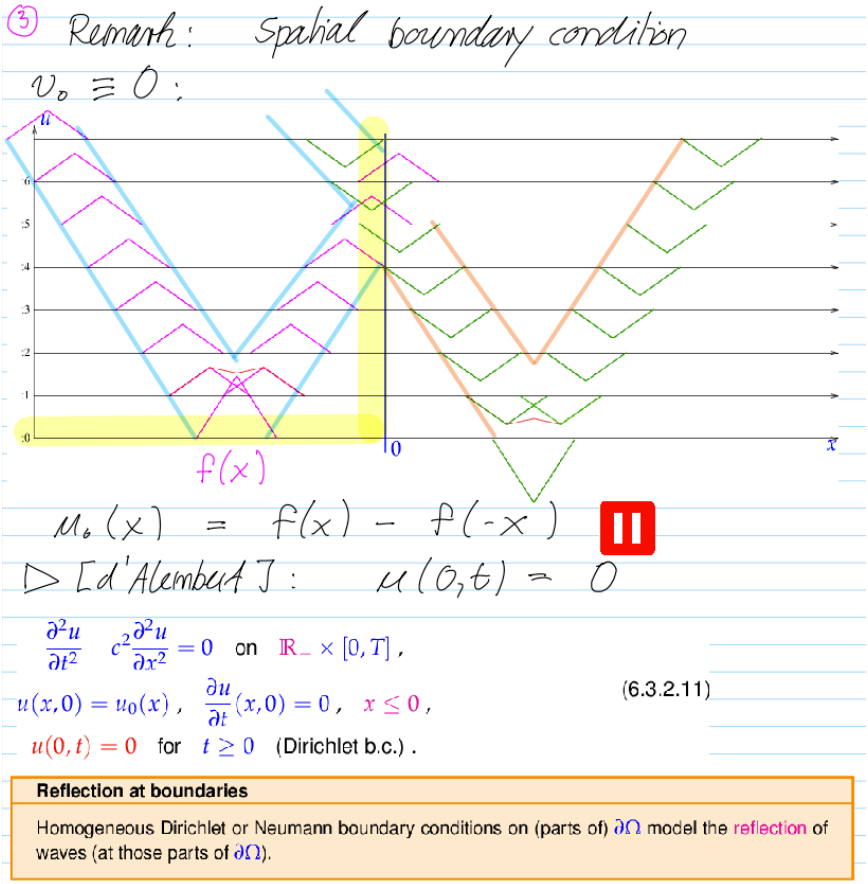
这种有限速度的扩散会造成的一个结果是：



- 时间步上的空间依赖性。该点的值只有前面一个domain里面的值影响
- 该范围上的值只影响后面xxx范围：
- 这两点可以通过这个图直观感受：



第二个能体现wave的特点是，我们可以wave在boundaries上的reflection。看下面这个例子：



最后一个特点是wave equation具备conservation of total energy without forcing。这点和heat equation不同？heat equation不是能量守恒？任何方程都是能量守恒的吧？没搞懂这里想要表达啥？

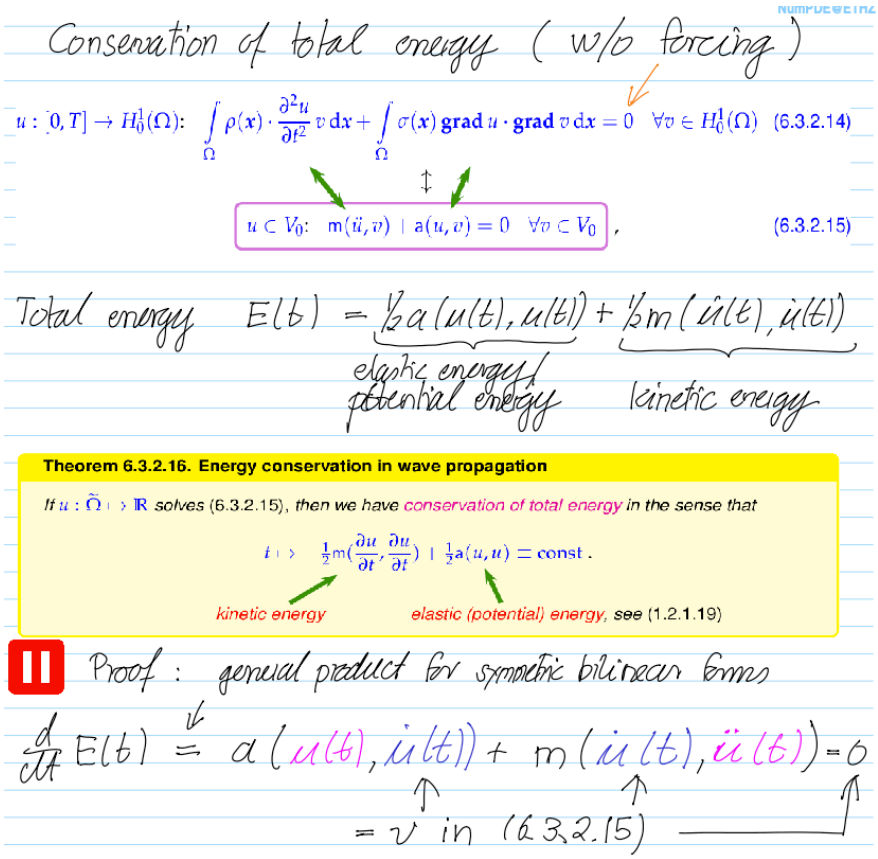
- 这里的意思是对于unbounded domain，解在domain上的L^2，wave equation随着时间marching保持一致。而heat equation则会逐渐降低，L^2 我们称之为energy。
  - Yes, the (L^2) norm can be an indicator of the "energy" or intensity of a function, and it can provide information about the presence of peaks or valleys in the solution profile.

    To clarify:

    1. **Magnitude of the (L^2) Norm:** A high (L^2) norm suggests that the function has regions with significant magnitudes, which could be peaks or valleys.
    2. **Decrease in (L^2) Norm:** A decreasing (L^2) norm, especially in the context of the heat equation, indicates that the solution is getting "smoother" over time. Peaks and valleys are getting flattened out due to diffusion.
    3. **Comparison with (L^1) Norm:** The (L^2) norm emphasizes peaks more than the (L^1) norm. For instance, if you have a function with a sharp spike, the increase in the (L^2) norm will be more pronounced than the increase in the (L^1) norm. This is because in the (L^2) norm, values are squared, so larger function values contribute more heavily.

4. **Limitations:** While the (L^2) norm can provide a general indication of the presence of peaks or valleys, it doesn't give location-specific information. For instance, a function with a single sharp peak and another function with multiple smaller peaks might have the same (L^2) norm.

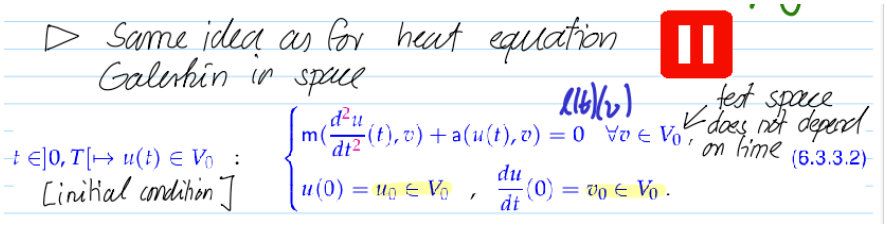
If you specifically want to detect or locate peaks and valleys, other tools and techniques might be more direct, such as gradient information, curvature analysis, or more sophisticated signal processing methods. However, the (L^2) norm can provide a useful measure of the overall intensity or energy of a function and how it evolves over time, especially in the context of PDEs like the heat equation.



6.3.3 Method of Lines for wave propagation

Spatial Galerkin Method

该章节的核心同样是我们得到半离散化的ode(对spatial domain离散)。这种方法称之为method of lines.注意在这里，test space does not depend on time.这个地方其实我不太理解test space depend on time的情况会造成什么样的影响(因为这里的trail space是time-dependent，这里是想说明我们在改时刻上的空间离散化）



然后，我们的正常流程如下：

1. 将test space和trail space用离散化后的space进行表示。这里需要注意，initial condition可能不在我们离散化后的space中，在这种情况下，我们需要将本来的解projection或者interpolant到离散后的space中去。这和我们在elliptic PDE中处理dirichlet BC一样。

   (a)

   I : Replace $V_0 \longrightarrow V_{0,h} \subset V_0$, $\dim V_{0,h} := N$ ↑ independent of time

   $t \in]0,T[\mapsto u(t) \in V_{0,h}$:

   $$\begin{cases} m\left(\frac{d^2 u_h}{dt^2}(t), v_h\right) + a(u_h(t), v_h) = 0 \quad \forall v_h \in V_{0,h}, \\ u_h(0) = \text{projection/interpolant of } u_0 \text{ in } V_{0,h}, \\ \frac{du_h}{dt}(0) = \text{projection/interpolant of } v_0 \text{ in } V_{0,h}. \end{cases} \quad (6.3.3.3)$$

   approximate initial cond.

2. 选择basis 这个地方我有点疑问，我怎么考虑选择不同的space？如果离散的space确定了，那么其basis怎么选都无所谓，反正是针对这个space的。如果我想用RBF basis，我该怎么选择我们的space？这里的$V_{0,h}$当初是怎么决定的？（我觉得应该是basis决定space而不是先有space后再考虑basis，我们之前讨论polynomial space中的basis是假定我们讨论的space就是polynomial组成的 这个地方的解释：如果我们选择用RBF basis，那么你就不具备localization的性质了。你的矩阵也就不是一个大型的sparse matrix，你也不能通过element再组装的方式进行求解Galerkin matrices了。你必须在整个求解域上去做积分从而来建立你的Galerkin matrix。下面是chatgpt的回答。

   > When we talk about the "size" of a function space, we're typically referring to the richness or expressiveness of the functions that can be represented within it. If both the RBF-based method and the traditional FEM method (using, say, polynomial bases) have the same number of nodes, both methods will result in function spaces with the same dimensionality, meaning they can both represent the same number of linearly independent functions.
   >
   > However, the character and properties of these function spaces can be quite different:
   >
   > (a) **Flexibility and Smoothness**: RBFs, especially Gaussian-based or multiquadric RBFs, can represent smooth, infinitely differentiable functions. In contrast, traditional FEM bases (like linear or quadratic polynomials) are piecewise-defined and are only continuously differentiable to a certain degree. This means that for problems where high smoothness is desired, the RBF-based space might be more suitable.
   >
   > (b) **Localization**: Traditional FEM polynomial bases are local, meaning a basis function associated with a node is non-zero only in the vicinity of that node and its neighbors. In contrast, many RBFs are global functions, influencing the entire domain. However, there are also compactly-supported RBFs that have local influence.
   >
   > (c) **Interpolation**: RBFs have an inherent interpolatory property. Given the values at the nodes, RBFs can interpolate the values between the nodes. This is not necessarily true for FEM basis functions, which approximate rather than interpolate.
   >
   > (d) **Higher-Order Representations**: While both RBFs and polynomial bases can represent higher-order functions, achieving this in FEM might require the use of higher-degree polynomials, which complicates the element shape functions and integration procedures. With RBFs, increasing the "order" or flexibility often involves adjusting a parameter rather

than complicating the basis function's structure.

(e) **Generality of Domains**: The RBF method is mesh-free, which means it can be applied to irregular domains and in higher dimensions more naturally than FEM, which requires meshing that gets complex in irregular and higher-dimensional scenarios.

In conclusion, while the dimensionality of the function spaces (the number of linearly independent functions they can represent) might be the same given the same number of nodes, the properties of the spaces can be quite different. Whether one space is "larger" or more expressive than another can depend on the specific problem at hand and the characteristics needed for that problem.

(a)

II. Ordered basis $\mathcal{B} = \{b_h^1, \dots, b_h^N\} \subset V_{0,h}$

$$u_h(t) = \sum_{j=1}^{N} \mu_j(t)\, b_h^j$$

▷ LWE - MOL - ODE : (2nd-order)

(6.3.3.3) ⇒

$$\begin{cases} \mathbf{M}\left\{\frac{d^2}{dt^2}\vec{\mu}(t)\right\} + \mathbf{A}\vec{\mu}(t) = \vec{\varphi}(t) & \text{for } 0 < t < T, \\ \vec{\mu}(0) = \vec{\mu}_0 \,, \quad \frac{d\vec{\mu}}{dt}(0) = \vec{v}_0 \,. \end{cases} \qquad (6.3.3.4)$$

↑ ↑ → state space $\mathbb{R}^N$

▷ s.p.d. stiffness matrix $\mathbf{A} \in \mathbb{R}^{N,N}$, $(\mathbf{A})_{ij} := a(b_h^j, b_h^i)$ (independent of time),
▷ s.p.d. mass matrix $\mathbf{M} \in \mathbb{R}^{N,N}$, $(\mathbf{M})_{ij} := m(b_h^j, b_h^i)$ (independent of time),
▷ source (load) vector $\vec{\varphi}(t) \in \mathbb{R}^N$, $(\vec{\varphi}(t))_i := \ell(t)(b_h^i)$ (time-dependent),
▷ $\vec{\mu}_0 \triangleq$ coefficient vector of a projection of u_0 onto $V_{0,h}$.
▷ $\vec{v}_0 \triangleq$ coefficient vector of a projection of v_0 onto $V_{0,h}$.

接着，我们还要注意，离散后的系统同样需要满足conserved discrete energy（energy conservation is the structural properties?）

▷ $E_h(t) = \frac{1}{2}\vec{\mu}(t)^T A \vec{\mu}(t) + \frac{1}{2}\dot{\vec{\mu}}(t)^T M \dot{\vec{\mu}}(t)$
↳ conserved discrete energy

接着，我们可以选择将我们的LWE-MOL-ODE变化成1-st-order ode，这样我们就可以使用我们提出的一系列iterative method了。

Conversion to 1-st-order ODE : II

Auxiliary function : $\vec{v}(t) = \dot{\vec{\mu}}(t)$

2nd-order ODE : $\mathbf{M}\left\{\frac{d^2}{dt^2}\vec{\mu}(t)\right\} + \mathbf{A}\vec{\mu}(t) = 0$

← auxiliary unknown $\vec{v} = \dot{\vec{\mu}}$

State space $\mathbb{R}^{2N}$

$$\begin{cases} \frac{d}{dt}\vec{\mu}(t) = \vec{v}(t) \,, \\ \mathbf{M}\frac{d}{dt}\vec{v}(t) = -\mathbf{A}\vec{\mu}(t) \,, \end{cases} \quad 0 < t < T \,. \qquad (6.3.3.6)$$

with intial conditions

$$\vec{\mu}(0) = \vec{\mu}_0 \,, \quad \vec{v}(0) = \vec{v}_0 \,. \qquad (6.3.3.7)$$

6.3.4 Timestepping for semi-discrete wave equations

前一章节我们考虑了spatial Galerkin discretization method。为了求解，我们需要full-discrete Numerical method。time-stepping for semi-discrete wave equations

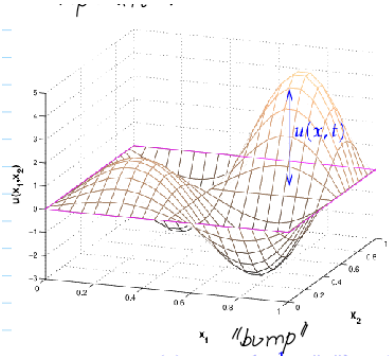
首先，我们先描述这个wave equation所具有的structural properties。而我们再用数值方法求解此问题时，就是注意我们的数值方法要满足这些structural properties才是合理的。

(6.3.3.3) $\Rightarrow$ $\begin{cases} \mathbf{M}\left\{\frac{d^2}{dt^2}\vec{\mu}(t)\right\} + \mathbf{A}\vec{\mu}(t) = \vec{\varphi}(t) & \text{for } 0 < t < T, \\ \vec{\mu}(0) = \vec{\mu}_0 \ , \ \frac{d\vec{\mu}}{dt}(0) = \vec{v}_0 . \end{cases}$ (6.3.3.4)

Structural properties : - Energy conservation
- Reversibility ($t \leftrightarrow -t$)

- autonomous second-order pde才有这个性质：reversibility（同时也得without dissipation）

接着，我们便开始数值试验研究一下不同的数值方法的表现：



Model problem: wave propagation on a square membrane

no Forcing

$$\frac{\partial^2 u}{\partial t^2} - \Delta u = 0 \quad \text{on }]0,1[^2 \times]0,1[,$$
$$u(x,t) = 0 \quad \text{on } \partial\Omega \times]0,T[,$$
$$u(x,0) = u_0(x) \ , \ \frac{\partial u}{\partial t}(x,0) = 0 . \quad v_0 = 0$$

(membrane in flat frame, displaced, but at rest at initial time)

- Initial data $u_0(x) = \max\{0, \frac{1}{5} - \|x\|\}$, $v_0(x) = 0$, $\Rightarrow n = 30$
- $\mathcal{M} \hat{=}$ "structured triangular tensor product mesh", see Fig. 232, n squares in each direction,
- linear finite element space $V_{N,0} = \mathcal{S}^0_{1,0}(\mathcal{M})$, $N := \dim \mathcal{S}^0_{1,0}(\mathcal{M}) = (n-1)^2$,
- All local computations ($\rightarrow$ Section 2.7.5) rely on 3-point vertex based local quadrature formula "2D trapezoidal rule" (2.4.6.10). More explanations will be given in Rem. 6.3.4.18 below.

- Expl./Impl. Euler SSM : $\dot{u} = f(u)$

$\Rightarrow u^{(k+1)} = u^{(k)} + \tau f(u^{(k+1)})$

(6.3.3.3) $\Rightarrow$ $\begin{cases} \mathbf{M}\left\{\frac{d^2}{dt^2}\vec{\mu}(t)\right\} + \mathbf{A}\vec{\mu}(t) = 0 & \text{for } 0 < t < T, \\ \vec{\mu}(0) = \vec{\mu}_0 \ , \ \frac{d\vec{\mu}}{dt}(0) = \vec{v}_0 . \end{cases}$ (6.3.3.4)

Aux. var. $\vec{v} = \dot{\vec{\mu}}(t)$: $\frac{d}{dt}\begin{bmatrix} \vec{\mu} \\ \vec{v} \end{bmatrix} = \begin{bmatrix} \vec{v} \\ -\mathbf{M}^{-1}\mathbf{A}\vec{\mu} \end{bmatrix}$

[$\vec{\varphi} = 0$]

Expl. Euler: $\frac{\mu^{(j)}-\mu^{(j-1)}}{\tau} = \vec{V}^{(j-1)}$

[Impl.] $\frac{\vec{V}^{(j)}-V^{(j-1)}}{\tau} = -M^{-1}A\vec{\mu}^{(j-1)}$

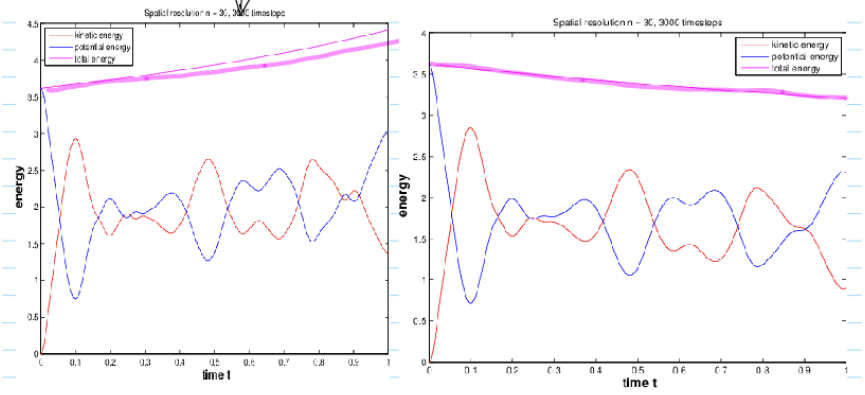
$\vec{\mu}^{(j)} - \vec{\mu}^{(j-1)} = \tau\vec{v}^{(j-1)}$,
$\mathbf{M}(\vec{v}^{(j)} - \vec{v}^{(j-1)}) = -\tau\mathbf{A}\vec{\mu}^{(j-1)}$.

explicit Euler

$\vec{\mu}^{(j)} - \vec{\mu}^{(j-1)} = \tau\vec{v}^{(j)}$,
$\mathbf{M}(\vec{v}^{(j)} - \vec{v}^{(j-1)}) = -\tau\mathbf{A}\vec{\mu}^{(j)}$.

implicit Euler

Observed: blow-up ↓ decay



exp increase of energy　　exp. decay of energy

- 老师这里通过matlab进行呈现的，发现对于explicit euler，我们的解会发散。而对于implicit euler，我们的解最后有会趋于0。通过画出他们的energy随着时间步的变化，我们发现他们一个energy会expontential increase，另一个energy回expontential decay。这太奇怪了，不过是真的有意思🤔

接着，老师这里尝试了implicit midpoint ssm ≡ 1-stage gauss collocation RK-SSM。

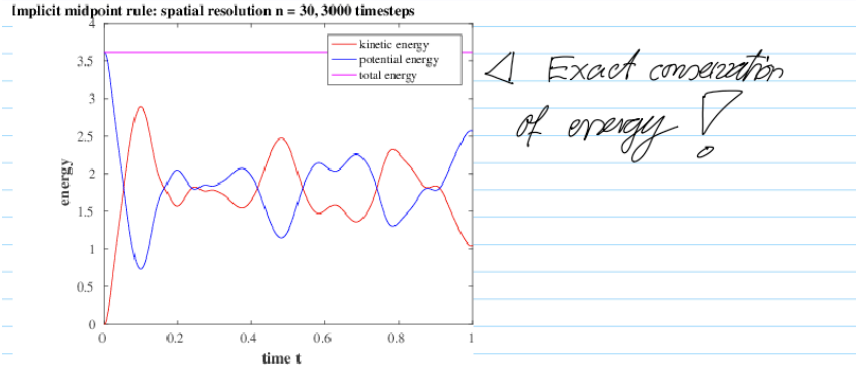
$\Psi^{t,t+\tau}\mathbf{u} := \mathbf{w}$: $\mathbf{w} = \mathbf{u} + \tau\mathbf{f}(t + \frac{1}{2}\tau, \frac{1}{2}(\mathbf{w}+\mathbf{u}))$, (6.2.6.7)

$\frac{w-u}{\tau} = f(\frac{1}{2}(w+u))$

▷ For LWE-MOL-ODE

$\frac{\mu^{(j)}-\mu^{(j-1)}}{\tau} = \frac{1}{2}(\vec{V}^{(j)}+\vec{V}^{(j-1)})$

$\frac{\vec{V}^{(j)}-V^{(j-1)}}{\tau} = -\frac{1}{2}M^{-1}A(\vec{\mu}^{(j)}+\mu^{(j-1)})$



◁ Exact conservation of energy ▽

- 发现这种方法能够exact conservation of energy!

下面我们就要搞清楚为什么，这需要我们进行rogorous analysis:

$$\begin{bmatrix}\vec{\mu}^{(j)}\\ \vec{v}^{(j)}\end{bmatrix}=\begin{bmatrix}\vec{\mu}^{(j-1)}\\ \vec{v}^{(j-1)}\end{bmatrix}+\tau\begin{bmatrix}\frac{1}{2}(\vec{v}^{(j-1)}+\vec{v}^{(j)})\\ -\frac{1}{2}\mathbf{M}^{-1}\mathbf{A}(\vec{\mu}^{(j-1)}+\vec{\mu}^{(j)})\end{bmatrix}+\tau\begin{bmatrix}0\\ \mathbf{M}^{-1}\vec{\varphi}(t_j-\frac{1}{2}\tau)\end{bmatrix}.$$

$$\Updownarrow$$

$$\begin{bmatrix}\mathbf{I} & -\frac{1}{2}\tau\mathbf{I}\\ \frac{1}{2}\tau\mathbf{A} & \mathbf{M}\end{bmatrix}\begin{bmatrix}\vec{\mu}^{(j)}\\ \vec{v}^{(j)}\end{bmatrix}=\begin{bmatrix}\mathbf{I} & \frac{1}{2}\tau\mathbf{I}\\ -\frac{1}{2}\tau\mathbf{A} & \mathbf{M}\end{bmatrix}\begin{bmatrix}\vec{\mu}^{(j-1)}\\ \vec{v}^{(j-1)}\end{bmatrix}+\tau\begin{bmatrix}0\\ \vec{\varphi}(t_j-\frac{1}{2}\tau)\end{bmatrix},$$

$$\Downarrow$$

$$\begin{aligned}&(\vec{\mu}^{(j-1)}+\vec{\mu}^{(j)})^{\top}\mathbf{A}^{\top}\cdot\ \vec{\mu}^{(j)}-\vec{\mu}^{(j-1)}=\frac{1}{2}\tau(\vec{v}^{(j-1)}+\vec{v}^{(j)})\\ &(\vec{v}^{(j-1)}+\vec{v}^{(j)})^{\top}\cdot\ \mathbf{M}(\vec{v}^{(j)}-\vec{v}^{(j-1)})=-\frac{1}{2}\tau\mathbf{A}(\vec{\mu}^{(j-1)}+\vec{\mu}^{(j)})+\tau\vec{\varphi}(t_j-\frac{1}{2}\tau).\end{aligned}\quad(*)$$

Use "$a^2-b^2=(a+b)(a-b)$": $(\vec{\alpha}+\vec{\beta})^{\top}\mathbf{A}(\vec{\alpha}-\vec{\beta})=\vec{\alpha}^{\top}\mathbf{A}\vec{\alpha}-\vec{\beta}^{\top}\mathbf{A}\vec{\beta}$ (from $\mathbf{A}=\mathbf{A}^{\top}$)

- 注意，我们这里用了一个trick，就是左乘这些手写的项。使用这个公式的前提是我们的A是spd的，但是这点特征我们之前在离散格式中已经证明过，所以完全正确。

Add equ.:

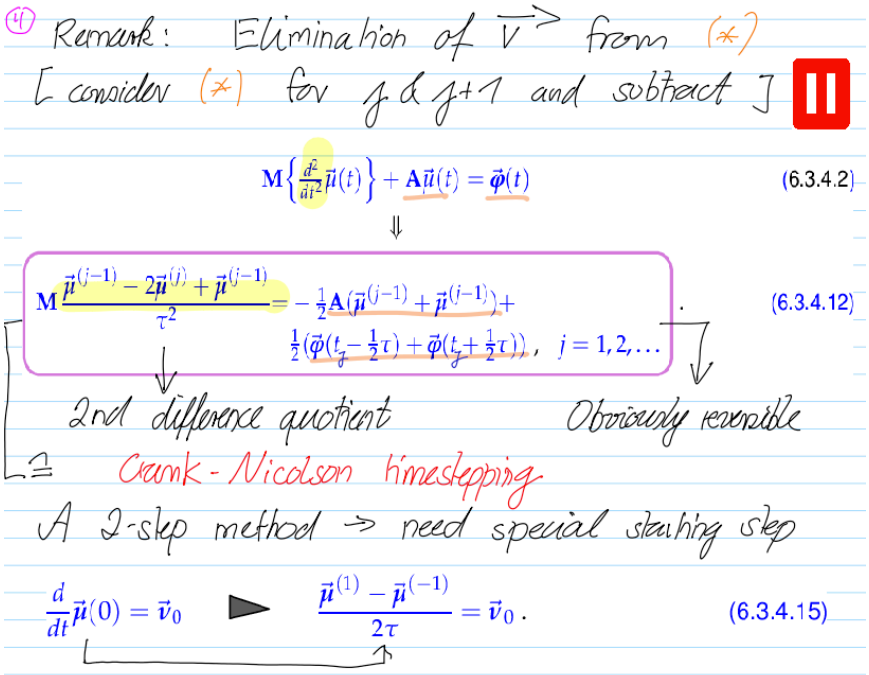
$$\begin{aligned}&(\vec{\mu}^{(j)})^{\top}\mathbf{A}(\vec{\mu}^{(j)})-(\vec{\mu}^{(j-1)})^{\top}\mathbf{A}(\vec{\mu}^{(j-1)})+(\vec{v}^{(j)})^{\top}\mathbf{M}(\vec{v}^{(j)})-(\vec{v}^{(j-1)})^{\top}\mathbf{M}(\vec{v}^{(j-1)})\\ &=\frac{1}{2}\tau\Big((\vec{\mu}^{(j)}+\vec{\mu}^{(j-1)})^{\top}\mathbf{A}(\vec{v}^{(j)}+\vec{v}^{(j-1)})-(\vec{v}^{(j)}+\vec{v}^{(j-1)})^{\top}\mathbf{A}(\vec{\mu}^{(j)}+\vec{\mu}^{(j-1)})\Big)+\\ &\tau(\vec{\mu}^{(j)}+\vec{\mu}^{(j-1)})^{\top}\vec{\varphi}(t_j-\frac{1}{2}\tau)\\ &=\tau(\vec{\mu}^{(j)}+\vec{\mu}^{(j-1)})^{\top}\vec{\varphi}(t_j-\frac{1}{2}\tau),\end{aligned}$$

since $=0$, $\mathbf{A}=\mathbf{A}^{\top}$; $\Leftarrow 0$, if $\vec{\varphi}\equiv 0$

$\hookrightarrow = 2(E_h^{(j)}-E^{(j-1)})$

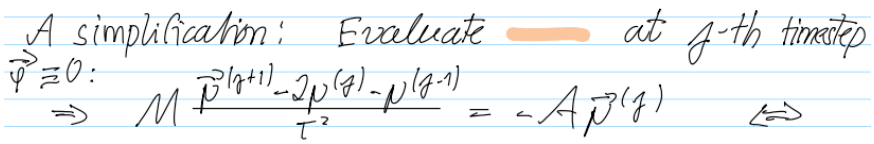
- 接着，我们将两个公式相加并消掉一些项，便可以得到我们在没有外界force的情况下energy是conserved。

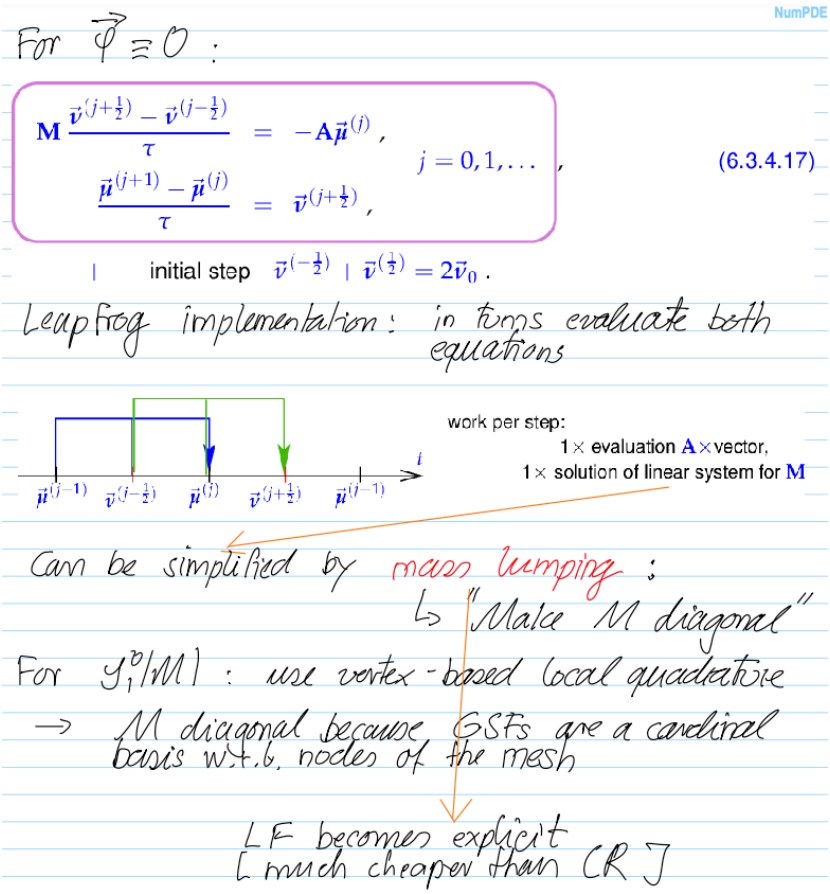
对于这个implicit midpoint ssm，我们通过eliminate auxiliary function v，便可得到如下形式：



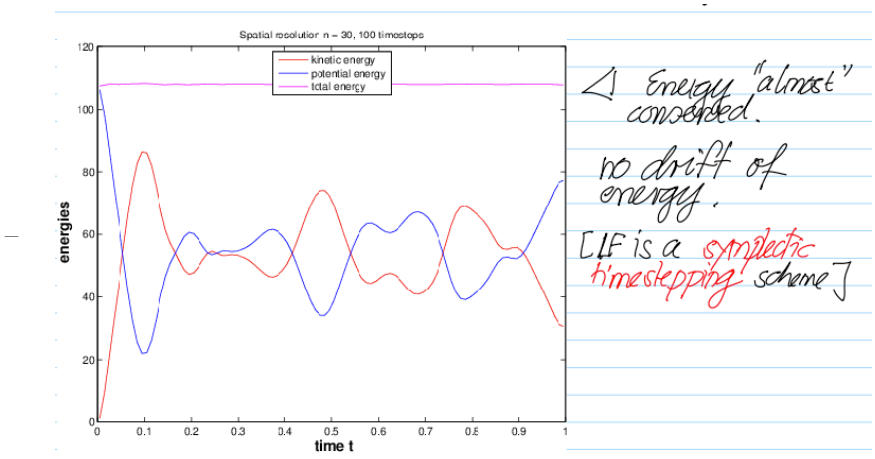
- 公式(6.3.4.12)有点问题，公式里面应该是$A(\vec{\mu}^{(j-1)} + \vec{\mu}^{(j)})$
- 我们将这个形式称为Crank-Nicolson timestepping.这是个需要2-step method。但是，因为我们有加速度（也就是u的导数信息）我们通常可以给出这个2-step method的初始迭代。

下面，我们也可以将这个格式简化（变成半implicit），就是右端项j-th timestep





- 然后，这种方法我们称为：leapfrog implementation。但是，这个方法的需要求LSE，这就使得我们考虑能不能将这个问题进一步simplified。我们用mass lumping将这个问题进一步简化。简化之后的LF就是一个explicit model了，他的速度比CR更快。但是LF does not conserve total energy,但是我们发现：



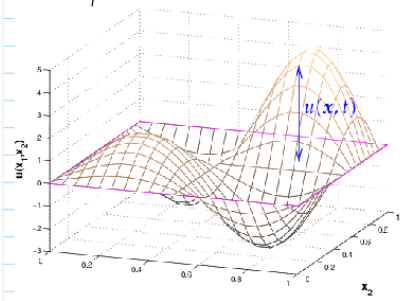
他的energy不会爆炸，只会在一个constant值附近有轻微的涨落！这里面的理论比较复杂，LF is a symplectic timestepping scheme，我这里就不做介绍了。

6.3.5 The Courant-Friedrichs-Levy (CFL) Condition

这里的核心问题就是，当我们用LF去解wave equation，我们是否会遇到算法稳定性的问题？也就是是否有对t的限制从而保证算法的问题？刚才我们只是保证算法在演化的过程中其能量不会爆炸。下面我们想要进一步考虑在time marching的过程中能否error的可控性。

让我们先来看一个数值试验：

② Exp :

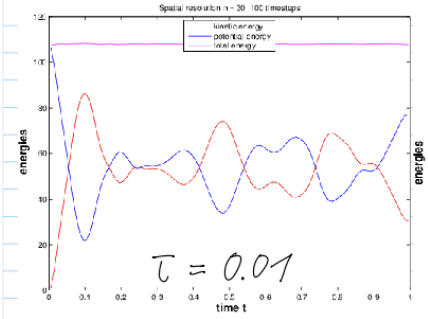


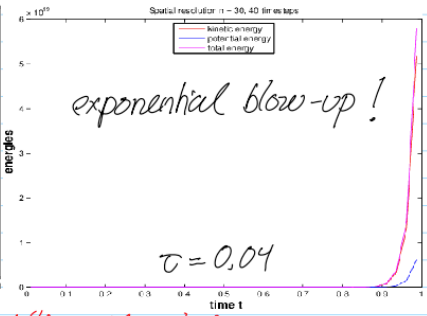
Model problem: wave propagation on a square membrane

$$\frac{\partial^2 u}{\partial t^2} - \Delta u = 0 \quad \text{on} \quad]0,1[^2 \times]0,1[\,,$$
$$u(x,t) = 0 \quad \text{on} \quad \partial\Omega \times]0,T[\,,$$
$$u(x,0) = u_0(x) \quad , \quad \frac{\partial u}{\partial t}(x,0) = 0\,.$$

(membrane in flat frame, displaced, but at rest at initial time)

$S_0^1(M)$ - FEM, $M \subseteq$ t.p. mesh





exponential blow-up !

▷ LF is only conditionally stable
≙ stability-induced timestep constraint

- 通过这个试验我们看出，LF虽然高效，但它不是unconditionally stable的，而是有stability-induced timestep constraint

接下来，我们想研究一下为什么会这样，我们还是和前两章那样的手段，采用diagonalization：

NumPDE@ET

Diagonalization :

$\exists\, T \in \mathbb{R}^{N,N}$: $AT = MTD$, $D = \mathrm{diag}(\lambda_1, \dots, \lambda_N)$
$T^T M T = I$

$$\mathbf{M}\frac{\vec{\mu}^{(j+1)} - 2\vec{\mu}^{(j)} + \vec{\mu}^{(j-1)}}{\tau^2} = \mathbf{A}\vec{\mu}^{(j)} \quad \xrightarrow[\vec{\mu} = T\vec{\eta}]{\vec{\eta} = T^T M \vec{\mu}} \quad \vec{\eta}^{(j+1)} - 2\vec{\eta}^{(j)} + \vec{\eta}^{(j-1)} = \tau^2 \mathbf{D}\vec{\eta}^{(j)}\,.$$

$$\eta_i^{(j+1)} - 2\eta_i^{(j)} + \eta_i^{(j-1)} = -\tau^2\lambda_i\eta_i^{(j)}\,, \quad i = 1,\dots,N\,, \quad j = 1,2,\dots\,. \qquad (6.3.5.4)$$

[Scalar linear 3-term recursion]

Try : $\eta_i^{(j)} = \xi^j$, $\xi \in \mathbb{C}$

▷ ξ is a root of the characteristic polynomial

$$\xi^2 - 2\xi + 1 = -\tau^2\lambda_i\xi \quad \Leftrightarrow \quad \xi^2 - (2 - \tau^2\lambda_i)\xi + 1 = 0\,.$$

$$\Rightarrow \quad \text{two solutions} \quad \xi_\pm = \frac{1}{2}\left(2 - \tau^2\lambda_i \pm \sqrt{(2-\tau^2\lambda_i)^2 - 4}\right)\,.$$

Blow-up $\Leftrightarrow$ $|\xi| > 1$
Vieta's theorem : $\xi_+ \cdot \xi_- = 1$
$|\text{root}| > 1$ $\Leftrightarrow$ two distinct real $\in \mathbb{R}$

discriminant $D := (2-\tau^2\lambda_i)^2 - 4 \leq 0 \Leftrightarrow \boxed{\tau \leq \frac{2}{\sqrt{\lambda_i}}}$. (6.3.5.5)

timestep constraint

blow-up, if $\tau > \frac{2}{\sqrt{\lambda_i}}$ for a single λ_i

For Lagr. FEM, h-refinement

Lemma 6.2.7.32. Behavior of of generalized eigenvalues

Let $\mathcal{M}$ be a simplicial mesh and $\mathbf{A}$, $\mathbf{M}$ denote the Galerkin matrices for the bilinear forms $a(u,v) = \int_\Omega \mathbf{grad}\, u \cdot \mathbf{grad}\, v \,\mathrm{d}x$ and $m(u,v) = \int_\Omega u(x)v(x)\,\mathrm{d}x$, respectively, and $V_{0,h} := \mathcal{S}^0_{p,0}(\mathcal{M})$. Then the smallest and largest generalized eigenvalues of $\mathbf{A}\vec{\mu} = \lambda\mathbf{M}\vec{\mu}$, denoted by $\lambda_{\min}$ and $\lambda_{\max}$, satisfy

$$\frac{1}{\operatorname{diam}(\Omega)^2} \leq \lambda_{\min} \leq C \quad , \quad \lambda_{\max} \geq C h_{\mathcal{M}}^{-2} ,$$

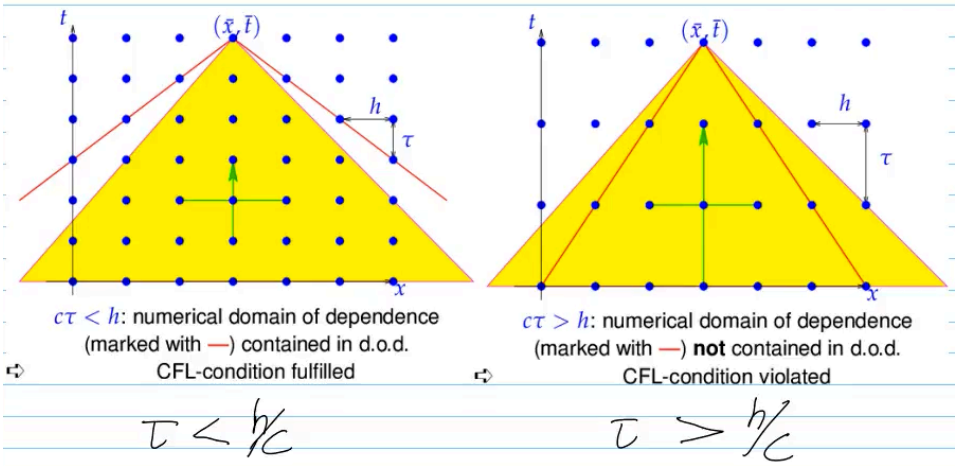
where the "generic constants" (↦ Rem. 3.3.5.10) depend only on the polynomial degree p, the domain Ω, and the shape regularity measure $\rho_{\mathcal{M}}$.

$\triangleright$ $\tau \leq C h_{\mathcal{M}}$: CFL-condition

↑ "generic constant"

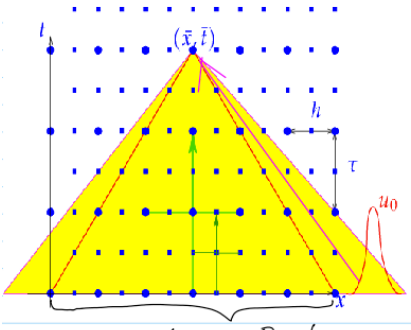
- 这个推导过程中有几个点需要注意：
  - 当我们采用diagonalization decoupling之后，不同于之前的one-step iteration，这里的是3-term recursion。因此我们相当于假设了一个函数，就是解可能随着这个形式变化：Try: $\eta_i^{(j)} = \xi^j$, $\xi \in \mathbb{C}$ 接下来我们就知道，如果值大于1，则会发散。而我们发现该ξ的值刚好由characteristic polynomial来进行决定。而我们知道，只要两个更属于实数，则会爆炸，复数则不会，因此我们给出了我们的constraint标准这里挺迷惑的，为什么可以通过实数和复数就决定他们两个的值是否大于1?
  - 再搞清楚time step和eigen values之间的关联后，我们就可以用上面那个lemma，进而推断出time step和网格width之间的关联，从而就是我们的CFL-condition。

再通过代数的角度推出来CFL-condition之后，我们也可以从几何的角度来理解，核心的概念是要理解flow of information，理解下面这两个图即可。这里稍微还有些疑问，之后再看看



$\tau < h/c$ $\tau > h/c$

- 太神奇了，红线内是数值能看到的，而黄线范围内的是真正影响该点解的。



($\bullet \hat{=}$ coarse grid, $\blacksquare \hat{=}$ fine grid, $\blacksquare \hat{=}$ d.o.d)

< 1D consideration:

sequence of equidistant space-time grids of $\widetilde{\Omega}$ with $\tau = \gamma h$ (τ/h = meshwidth in time/space)

If $\gamma >$ CFL-constraint (here $\gamma > c^{-1}$), then

analytical domain of dependence $\not\subset$ numerical domain of dependence

$u_0 = 0$ here

CFL-condition $\Leftrightarrow$ Exact d.o.d. $\subset$ numerical d.o.d.

接下来，我们想知道CFL所决定的constraint会不会影响我们算法关于error的收敛速度？

Does CFL-condition compromise efficiency [Does it enforce a timestep constraint smaller than required by accuracy ?]

因为我们知道error是等于spatial error+temporal error，当你的temporal被你的constraint所决定从而很小时，即使你的temporal error很小，但是的spatial error基本上还是很大，所以这个时候改变你的time step基本上不会改变你解的error，因此我们想使得解的error下降，就需要协调使得spatial error和temporal error都按照一定比例下降，从而才能使得解的error按照对应的比例下降：

"Meta-theorem" 6.3.5.9. Convergene of fully discrete solutions of the wave equation

Assume that

- *the solution of the IBVP for the wave equation (6.3.2.14) is "sufficiently smooth",*
- *its spatial Galerkin finite element discretization relies on degree p Lagrangian finite elements (→ Section 2.6) on uniformly shape-regular families of meshes.*
- *timestepping is based on the leapfrog method (6.3.4.17) with uniform timestep $\tau > 0$ satisfying (6.3.5.5).*

Then we can expect an asymptotic behavior of the total discretization error according to

$$\left(\tau\sum_{j=1}^{M}\|u-u_h(\tau j)\|^2_{H^1(\Omega)}\right)^{\frac{1}{2}} \leq C\left(h_{\mathcal{M}}^{p}+\tau^2\right), \tag{6.3.5.10}$$

$$\left(\tau\sum_{j=1}^{M}\|u-u_h(\tau j)\|^2_{L^2(\Omega)}\right)^{\frac{1}{2}} \leq C(h_{\mathcal{M}}^{p+1}+\tau^2), \tag{6.3.5.11}$$

where $C > 0$ must not depend on $h_{\mathcal{M}}$, τ. L.F. is 2nd-order !

▷ Spatial & temporal error add up!

Goal: error reduction by factor $\rho > 1$

contribution of spatial (energy) error $\approx Ch_{\mathcal{M}}^{p}$, $h_{\mathcal{M}} \hat{=}$ mesh width (→ Def. 3.2.1.4) ,

contribution of temporal error $\approx C\tau^2$, $\tau \hat{=}$ timestep size . (6.3.5.12)

[Error balancing]

reduce mesh width by factor $\rho^{1/p}$

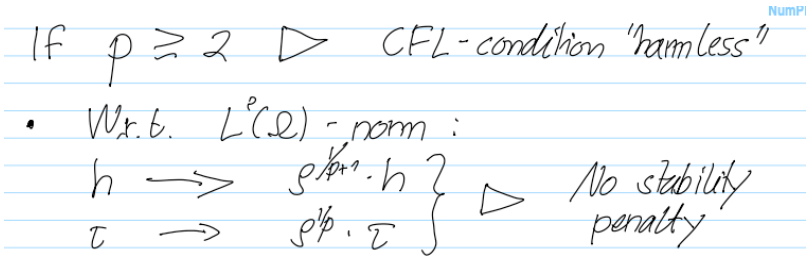
reduce timestep by factor $\rho^{1/2}$

$\overset{(6.2.8.7)}{\Longrightarrow}$ (energy) error reduction by $\rho > 1$. (6.3.5.13)

↳ $H^1(\Omega)$-norm !

CFL : $\tau \leq Ch_{\mathcal{M}}$ ⇒ reduce τ by factor $\rho^{1/p}$

- 就像这里，我们想要error下降factorρ，那么我们的mesh width和timestep需要分别下降$\rho^{1/p}$和$\rho^{1/2}$。



如果我们的p大于2，那么，当mesh width按照比例下降时，所决定的稳定性的time step会大于按照比例下降的time step，从而CFL-condtion无害。

而如果我们看L^2 norm，那就更不用考虑CFL的影响力。

这里挺困惑的，采用不同的norm会有这么大的区别

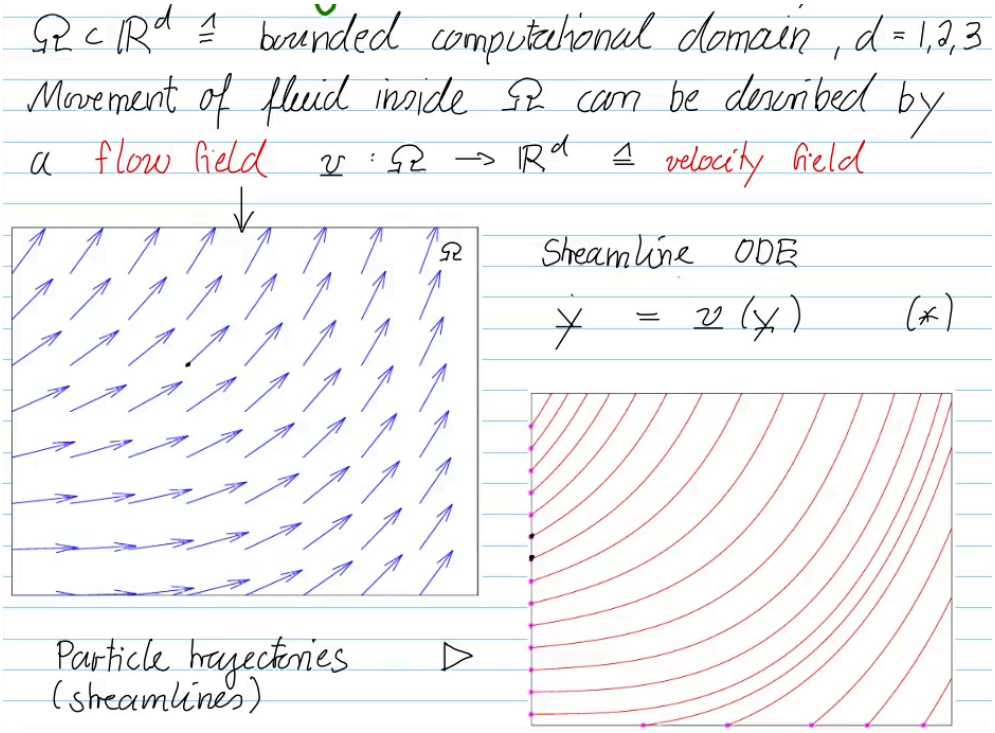
7 Convection-Diffusion Problems

7.1 Heat Conduction in a Fluid

7.1.1 Moldeling Fluid Flow

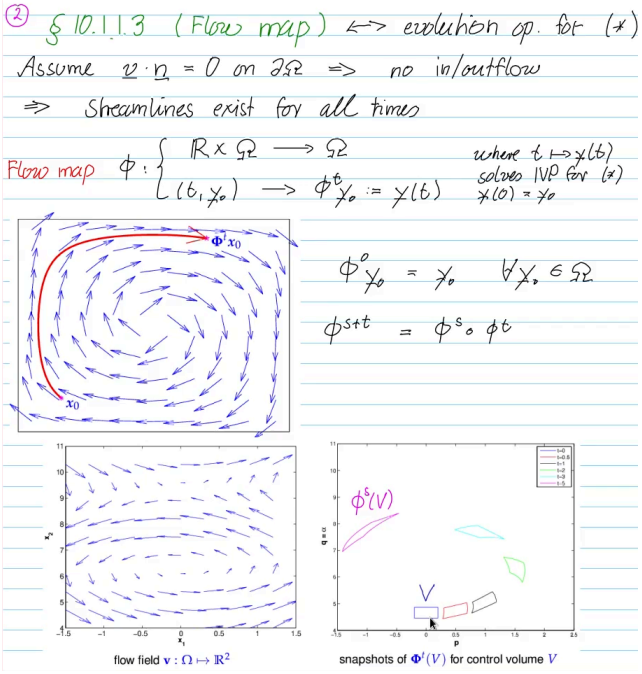
transport given the velocity field. So far, we have seen the transport due to diffusion. This is the typical effect in heat conduction. and wave propogation. Now, it is about transport combined with diffusion. Diffusion in the moving particles. We want to derive the continuous model for the stationary heat conduction in the moving fluid particles.

截至目前，我们已经学习了固体温度的diffusion，wave equation。这一章我们要研究都模型就是有温差的液体，其温度在液体内部的扩散。不同与固体，这里除了温度的扩散还涉及液体particles的迁移。我们需要想要建模这种fluid flow。这里我们用flow field(velocity field)来见面流体的流动。需要注意的是，我们考虑的stationary的情况，也就是flow field不随时间变化。那么，粒子的轨迹就是妥妥的stramlines，我们这里提出sreamline ODE如果ODE的话，那么particels是个有限维度的描述？streamline ODE其实就是描述有限维度流体例子的trajectories?



- 可以看出用stramline是一种描述流体运动的方式 Lagrange Description
- v is supposed to be lipshitz continuous.

另一种是用Flow map，其实就是evolution operator，这个我见过很多了。这里举的例子中有假设，也就是粒子被confined在这个domain中，不会流出去。并且，也介绍了flow map妈祖的一些性质，像Group Properties.



7.1.2 Heat Convection and Diffusion

下面，我们就要来研究我们上一章提到的那个问题，heat在fluid particles中的运动。我们要建模我们的BVP。这里，velocity field是给你了的（其中，velocity field我们是假设v is supposed to be lipshitz continuous.）。并且container也给了（这里应该又是一个fixed container）。我们的目标是build我们的BVP governing temperature field.我们采用了和heat diffusion一样的逻辑。

Given : "Container" $\Omega \subset \mathbb{R}^d$, $d = 1,2,3$

Stationary fluid velocity field $v : \Omega \to \mathbb{R}^d$

Sought : BVP governing temperature field $u : \Omega \to \mathbb{R}$

Same reasoning as for heat conduction in a solid :

Conservation of energy

$$\operatorname{div} \mathbf{j} = f \Leftrightarrow \int_{\partial V} \mathbf{j} \cdot \boldsymbol{n} \, dS = \int_V f \, dx \quad \text{for all "control volumes" } V. \qquad (1.6.0.3)$$

power flux through surface of V — heat production inside V

New : Heat flux due to transport of stored thermal energy

↓

Simple model : $= \beta u$

$\beta > 0$: heat capacity

我们有balance law，唯一不同的点就是对balance law中flux的刻画包含了两项。

Fourier's law in moving fluid

$$\mathbf{j}(x) = -\kappa \operatorname{grad} u(x) + \mathbf{v}(x)\rho u(x), \quad x \in \Omega. \qquad (10.1.2.4)$$

diffusive heat flux (due to spatial variation of temperature) — convective heat flux (due to fluid flow)

由此我们便可以得到我们的Convection-diffusion equation for temperature.

$$\operatorname{div} \mathbf{j} = f \quad + \quad \mathbf{j}(x) = -\kappa \operatorname{grad} u(x) + \mathbf{v}(x)\rho u(x)$$

$$-\operatorname{div}(\kappa \operatorname{grad} u) + \operatorname{div}(\rho \mathbf{v}(x) u) = f \quad \text{in } \Omega. \qquad (10.1.2.8)$$

Linear scalar **convection-diffusion equation** (CDE)

$$-\operatorname{div}(\kappa \operatorname{grad} u) + \operatorname{div}(\rho \mathbf{v}(x) u) = f.$$

$-\operatorname{div}(\kappa \operatorname{grad} u)$: diffusive term (2nd-order); $\operatorname{div}(\rho \mathbf{v}(x) u)$: convective term (1st-order)

接下来，一个完整的BVP还要求Boundary conditions的施加，这里，和二阶椭圆PDE一样，没有什么新的东西，这主要是基于这个理论，边界条件主要用高阶项来主导。但是需要⚠️的一点是，当boundary conditions中出现flux这些时，我们需要用上面提到的包含convective term的flux

"Nothing new" : ↳ same as for std. 2nd-order scalar elliptic PDEs

Suitable boundary conditions fitting a PDE are determined by the highest-order term.

↓

Here : diffusion term

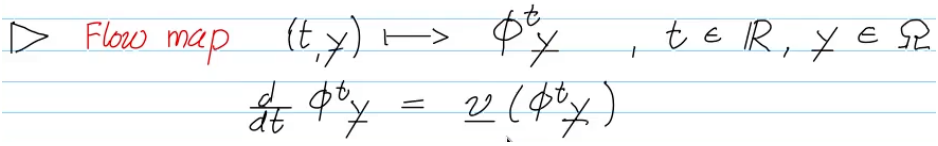
7.1.3 Incompressible Fluids

这一章我们想看看不可压流在公式上是怎么体现的，其又有哪些性质：

Given : Velocity field $v : \Omega \to \mathbb{R}^d$, $\Omega \subset \mathbb{R}^d$, describing motion of fluid

首先我们先回顾flow map和velocity field的关系，这对我们后面推导公式非常有用：

Flow map is the evolution operator for the streamline ODE.



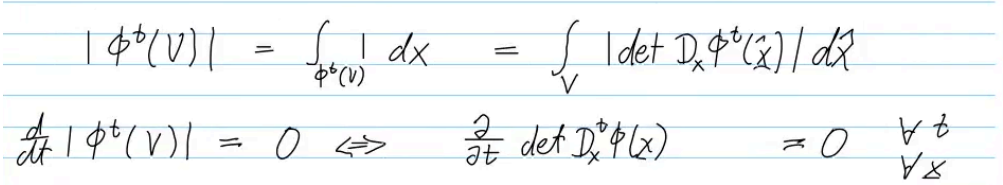
- Flow map在domain中某点处的时间导数就是改点的速度场(velocity field)

incompressible：就是由一堆particles组成的volume，在flow map的作用下不发生变化。在flow map上的数学表述就是这样

Definition 10.1.3.1. Incompressible flow field

A fluid flow is called **incompressible**, if the associated flow map Φ^t is *volume preserving*,

$$|\Phi^t(V)| = |\Phi^0(V)| = |V| \quad \text{for all sufficiently small } t > 0, \text{ and for all control volumes } V .$$



- 这里的变化用了transformation formula for integrals。这个推导挺自然的。

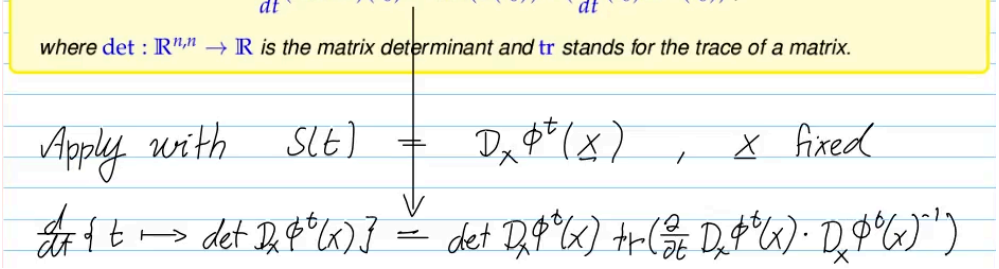
接着，这个determinants比较难处理，我们就用了下面这个公式去吧determinats提出来。

Theorem 10.1.3.7. Differentiation formula for determinants

Let $\mathbf{S} : I \subset \mathbb{R} \mapsto \mathbb{R}^{n,n}$ be a smooth matrix-valued function. If $\mathbf{S}(t_0)$ is regular for some $t_0 \in I$, then

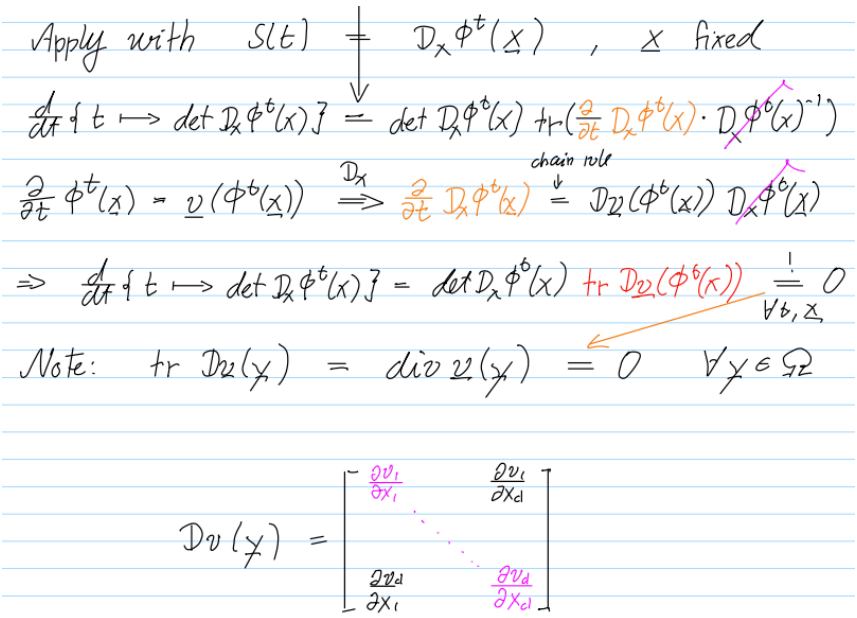
$$\frac{d}{dt}(\det \circ \mathbf{S})(t_0) = \det(\mathbf{S}(t_0)) \operatorname{tr}\left(\frac{d\mathbf{S}}{dt}(t_0)\mathbf{S}^{-1}(t_0)\right) ,$$

where $\det : \mathbb{R}^{n,n} \to \mathbb{R}$ is the matrix determinant and tr stands for the trace of a matrix.



- 这个地方假设了Jacobian是invertible的。当然这个也很显然成立，在t很小的时候。

接着，我们再用了变量的一些替代，最后就得到其实就是速度场Jacobian的trace在场中任何一个点都为零。



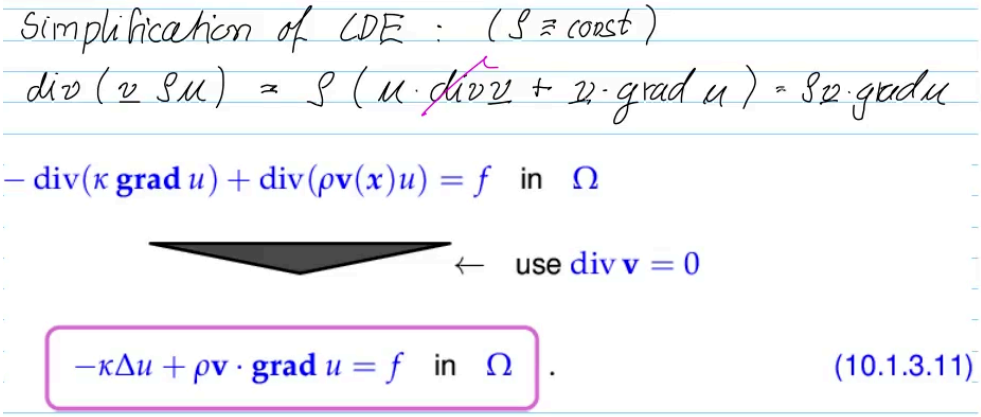
- 而这就是速度场在场中任意一点的散度都为零。

Theorem 10.1.3.8. Divergence-free velocity fields for incompressible flows

A stationary fluid flow in $\Omega \subset \mathbb{R}^d$ is incompressible (→ Def. 10.1.3.1), if and only if its associated velocity field $\mathbf{v} = [v_1, \ldots, v_d]^\top : \Omega \to \mathbb{R}^d$ satisfies $\operatorname{div} \mathbf{v} = \sum_{j=1}^d \frac{\partial v_j}{\partial x_j} = 0$ everywhere in Ω.

- 这个地方我们讨论的是stationary fluid flow，这点一定要注意。
- 不可压缩最基本的假设是flow map不改变volume，接着，在经过一系列的推导后，我们得到不可压缩就是速度场的散度是零。

之前推出的那个Convection-Diffusion Equation是不可压的，当我们把不可压条件表述进来后，我们得到：



- 注意，这里是假设我们的k和ρ都是constant

接下来，我们看看consequence of the incompressible convection-diffusion equaitons。其后果就是我们可以推出来maximum principle for convection-diffusion equations。不是incompressible我们就不能general到这个结论。

Remember :

Theorem 3.7.1.2. Maximum principle for 2nd-order elliptic BVP

For $u \in C^0(\overline{\Omega}) \cap H^1(\Omega)$ holds the ***maximum principle***

$$-\operatorname{div}(\kappa(x)\,\mathbf{grad}\,u) \geq 0 \implies \min_{x\in\partial\Omega} u(x) = \min_{x\in\Omega} u(x)\,,$$
$$-\operatorname{div}(\kappa(x)\,\mathbf{grad}\,u) \leq 0 \implies \max_{x\in\partial\Omega} u(x) = \max_{x\in\Omega} u(x)\,.$$

Can be generalized :

Theorem 10.1.3.13. Maximum principle for scalar 2nd-order convection diffusion equations
→

Let $\mathbf{v} : \Omega \mapsto \mathbb{R}^d$ be a continuously differentiable vector field and $u \in C^0(\overline{\Omega}) \cap C^2(\Omega)$. Then there holds the ***maximum principle***

$$-\Delta u + \mathbf{v}\cdot\mathbf{grad}\,u \geq 0 \implies \min_{x\in\partial\Omega} u(x) = \min_{x\in\Omega} u(x)\,,$$
$$-\Delta u + \mathbf{v}\cdot\mathbf{grad}\,u \leq 0 \implies \max_{x\in\partial\Omega} u(x) = \max_{x\in\Omega} u(x)\,.$$

heat sources only

- 这个就是就是告诉我们当有heat source在内部时，minimum值在边界上达到
- not source则是maximum在边界上达到
- 这个properties非常有用，将会指导我们的discretization

elliptic PDE的一个非常重要的structural properties。非常的有趣！

7.1.4 Transient Heat Flow in a Fluid

这里我们假设temperature field和velocity field都是time-dependent。

$\Omega \subset \mathbb{R}^d \triangleq$ bounded computational domain ("container")

Given : Time-dependent velocity field $\mathbf{v} = \mathbf{v}(x,t)$, $x \in \Omega$

Same reasoning as for solids (→ Sect. 9.2.1)

- 接着我们采用和solid一样的reasoning。一样的推导过程。我们就得到了下面的balance law。

这里的energy stored物理内涵之后再理解一下

Conservation of energy:

$$\frac{d}{dt}\int_V \rho u \,\mathrm{d}x + \int_{\partial V} \mathbf{j}\cdot\boldsymbol{n}\,\mathrm{d}S = \int_V f\,\mathrm{d}x \quad \text{for all "control volumes" } V\,. \qquad (9.2.1.3)$$

energy stored in V | power flux through ∂V | heat generation in V

- 这里的是intergral form，通过使用Gaussian theorem，我们可以将其转换成local form

$$\frac{\partial}{\partial t}(\rho u)(x,t) + (\operatorname{div}_x \mathbf{j})(x,t) = f(x,t) \quad \text{in } \Omega\,. \qquad (9.2.1.5)$$

- 这里挺有意思的，transient实在conservation of energy上做了改变，而moving particles则是在flux上做了改变。diffusion问题的核心就是conservation of energy和flux的表达形式。还有diffusion-reaction problem，其重要又是在source term上做了改动。

通过再generalized fourier law，我们得到下面的形式：

+ Generalized Fourier law :

$$\mathbf{j}(\boldsymbol{x},t) = -\kappa\,\mathbf{grad}\,u(\boldsymbol{x},t) + \mathbf{v}(\boldsymbol{x},t)\rho u(\boldsymbol{x},t)\,, \quad \boldsymbol{x}\in\Omega\,. \tag{10.1.2.4}$$

▷ Transient CDE :

$$\frac{\partial}{\partial t}(\rho u) - \operatorname{div}(\kappa\,\mathbf{grad}\,u) + \operatorname{div}(\rho\mathbf{v}(\boldsymbol{x},t)u) = f(\boldsymbol{x},t) \quad \text{in } \widetilde{\Omega} := \Omega\times]0,T[\,. \tag{10.1.4.1}$$

+ "elliptic" boundary conditions
+ initial condition : $u(x,0) = u_0(x)$

- 其中boundary conditon还是和elliptic的形式一样。而initial condtion在需要补上。
- 注意，这里我们都是驾驶vecotr field是知道的（在这个情况中vector field是个关于x，t的函数），转而求解的是temperature field，在后面我们会见到vector field不知道的情况，而同时求解vector field和temperature field。这就需要我们进一步用到动量守恒这个公式。目前我们只用到了能量守恒。

如果研究的incompressible flow，则公式进一步化简：

Equivalent for $\operatorname{div}_{\boldsymbol{x}} \underline{v}(\underline{x},t) = 0$:

$$\frac{\partial}{\partial t}(\rho u) - \kappa\Delta u + \rho\mathbf{v}(\boldsymbol{x},t)\cdot\mathbf{grad}\,u = f(\boldsymbol{x},t) \quad \text{in } \widetilde{\Omega} := \Omega\times]0,T[\,. \tag{10.1.4.2}$$

7.2 Stationary Convection-Diffusion BVPS: Numerics

在这一章节中，我们将会讨论CD BVPS的数值解法，在讨论这些数值解法之前，我们先来聊聊这类问题的一个特征，而这个特征将是我们处理CD的main problem - Singular Perturbation

首先，我们先来recap一下convection-diffusion BVPs

$\Omega \subset \mathbb{R}^d \triangleq$ bounded computational domain

Model problem :

$$-\epsilon\Delta u + \mathbf{v}(\boldsymbol{x})\cdot\mathbf{grad}\,u = f \quad \text{in } \Omega\,, \quad u = 0 \quad \text{on } \partial\Omega\,. \tag{10.2.0.1}$$

diffusive term (2nd-order term)

convective term (1st-order term)

Assume : $\underline{v} \in C^0(\bar{\Omega})$, $\operatorname{div}\underline{v} = 0$, $\|\underline{v}\|_\infty = 1$

↳ incompressible flow

- 我们这里写出PDE，其中，我们对velocity field做了一些假设，这将在后面保证我们解空间的存在性。

接着，我们乘test function，得到variational formulation形式的方程：

§ 10.2.0.2 (Variational formulation)

$$u \in H_0^1(\Omega): \quad \underbrace{\epsilon \int_\Omega \mathbf{grad}\, u \cdot \mathbf{grad}\, w \,\mathrm{d}x + \int_\Omega (\mathbf{v} \cdot \mathbf{grad}\, u)\, w \,\mathrm{d}x}_{\text{bilinear form } a(u,w)} = \underbrace{\int_\Omega f(x)\, w(x) \,\mathrm{d}x}_{\text{linear form } \ell(w)} \quad \forall w \in H_0^1(\Omega)\,. \tag{10.2.0.3}$$

$\hat{=}$ linear variational problem

- $a(\cdot,\cdot)$ not symmetric $\Rightarrow$ no energy norm
- $a(\cdot,\cdot)$ continuous on $H^1(\Omega)$, since v bounded

- 注意，这里有个非常关键的问题，就是bilinear form is not symmetric, which gives rise to that we cannot write down the energy norm.没有energy norm，我们就无法给出合理的function space从而有解的存在（也就是第一章中，我们讨论过，如果要使得solution存在，bilinear form不能无限大，也就是这里说的continuous）。在这里，由于我们刚才的假设，velocity field is bounded. therefore, we use H^1 space来替当我们的energy norm决定的space。

接着，我们讨论一下bilinear form的一个性质，我们发现：

- $w \in H_0^1(\Omega)$: $a(w,w) = \varepsilon \int_\Omega \|\mathrm{grad}\, w\|^2 \mathrm{d}x > 0 \quad w \neq 0$

$$\int_\Omega (v \cdot \mathrm{grad}\, w) w \,\mathrm{d}x = \int_\Omega (vw) \cdot \mathrm{grad}\, w \,\mathrm{d}x$$

i.b.p. $\downarrow$

$$= -\int_\Omega \mathrm{div}(v \cdot w)\, w \,\mathrm{d}x + \int_{\partial\Omega} w^2\, n \cdot v \,\mathrm{d}S \quad (\int_{\partial\Omega} = 0)$$

$= 0$

$$= -\int_\Omega (w \,\mathrm{div}\, v + v \cdot \mathrm{grad}\, w)\, w \,\mathrm{d}x \quad (\mathrm{div}\, v = 0)$$

$\Rightarrow$ $a(\cdot,\cdot)$ is positive definite

($\varepsilon > 0$)

$\triangleright$ Existence and uniqueness of solutions of (10.2.0.3)

- 这里的bilinear form其实是positive definite的，通过第一章的假设，这证明了我们解的existence and uniqueness。

7.2.1 Singular Perturbation

但是，上面讨论的那都不是问题，现在我们来看一个方程中隐藏的问题，就是我们讨论一下fast-moving fluid，速度场很大，intutively, transport of energy induced by the flow will be larger than that induced by the diffusion.在我们将方程rescale之后，其主要呈现下面这种形式：

$$-\varepsilon \Delta u + v(x) \cdot \mathrm{grad}\, u = f \quad \text{in } \Omega \qquad (10.2.0.1)$$

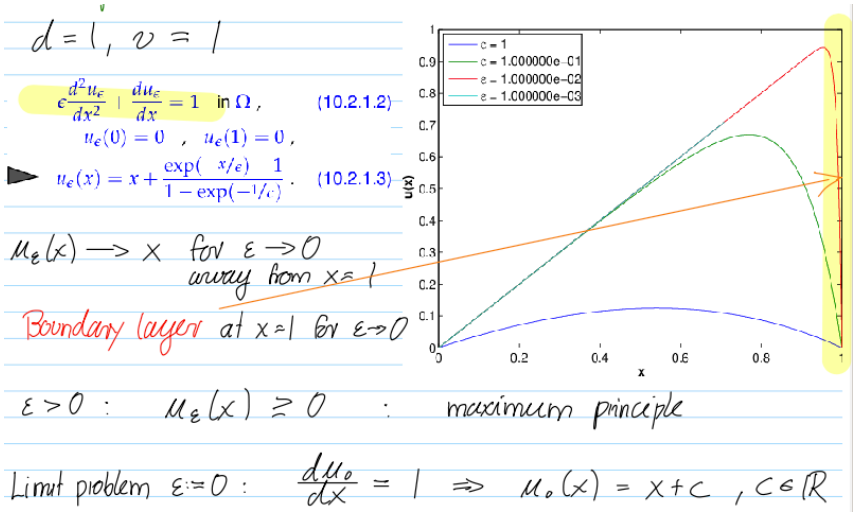
$$u = 0 \quad \text{on } \partial\Omega$$

Fast moving fluids: $\varepsilon \ll \|v\|_\infty = 1$

$\hat{=}$ dominant convection

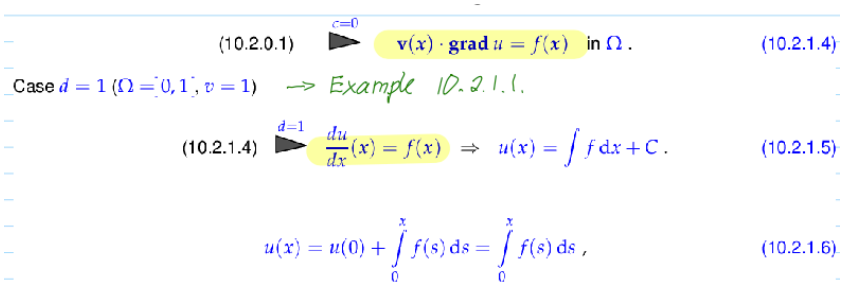
- 我们称之为dominant convection。

下面我们先来考虑一个1D problem从而来说明我们的问题：

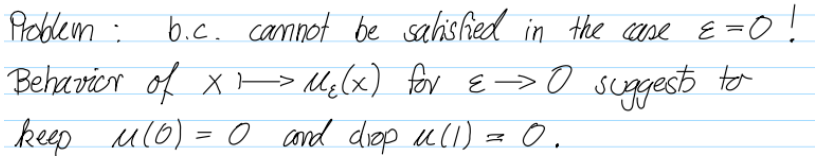


- 在这个例子中，我们发现，当ε越来越小的时候，我们的解会靠近x。此外，在右端会发生类似于boundary layer的情况。此外，只要当我们的$\varepsilon > 0$，根据我们在第一章中讨论的maximum principle，我们就知道解都是为正的。
- 当我们考虑limit problem的时候，也就是$\varepsilon = 0$，会发现boundary无法被满足。这非常奇怪。

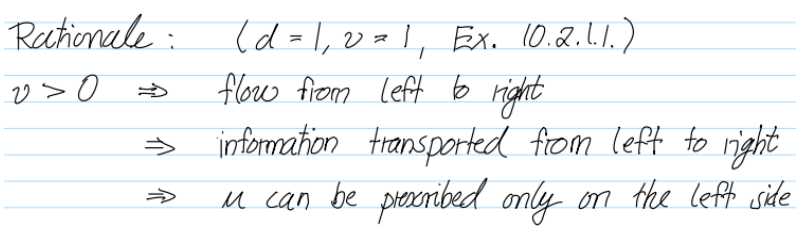
接下来我们进一步把limit problem拿出来单独考虑，我们考虑general dimension的limit problem，在这一类problem中，pure transport占据主导：



我们发现，问题其实退化成了一个常微分方程。而对于一个常微分方程，我们只需要initial condition，也就是这里$t = 0$时的情况，然后信息是从左往右进行流动。这就回引起下面的问题：



然后我们就得出了下面的rationale：



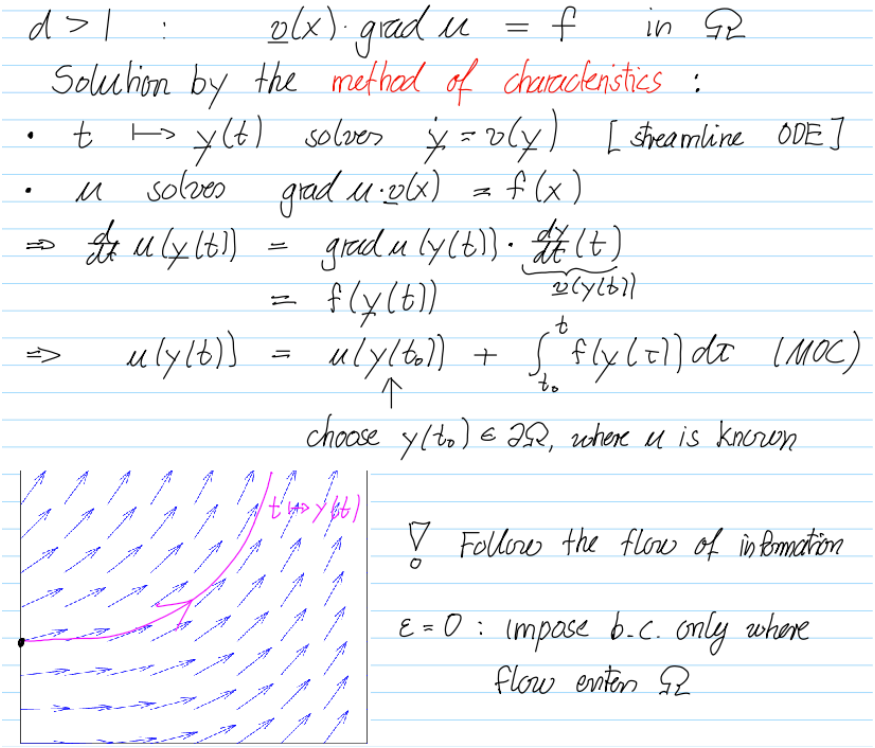
将这个问题整理一下，就是我们这里的singularly perturbed problem

Notion 10.2.1.9. Singularly perturbed problem

A boundary value problem depending on parameter $\epsilon \approx \epsilon_0$ is called singularly perturbed, if the limit problem for $\epsilon \to \epsilon_0$ is not compatible with the boundary conditions.

- 也就是BVP问题的解受到方程中某个参数的影响。而当这个参数无限靠近某个值的时候（可能是0，也可能不为0），我们的解将无法与我们的BC兼容。

下面我们进一步研究pure convection，只不过我们不控制维度为1



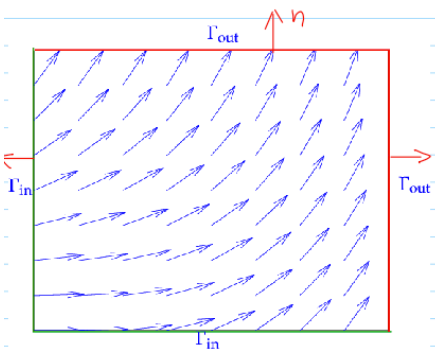
- 然后我们通过method of characteristics得出我们方程的解。
- 然后用streamline来表示方程的解，其中sreamline满足这样的常微分方程这里需要注意的是streamline在求解这个方程前就已经被指定了。其满足这个方程，由这个速度场来控制
- 下面，我们想看看温度沿这个streamline是怎么变化的。由此我们便得到了沿着stramline的解的形式。这个形式和我们刚才的一维形式很像，就是我们只在flow enter这个求解域的地方施加边界条件。因此，对于pure transport的问题，我们得到了如下的结论：

For the pure transport problem $\mathbf{v} \cdot \mathbf{grad}\, u = f$ Dirichlet boundary conditions can be imposed only on the inflow boundary part

$$\Gamma_{\text{in}} := \{x \subset \partial\Omega : \mathbf{v}(x) \cdot n(x) < 0\} . \tag{10.2.1.11}$$

but not on its complement in $\partial\Omega$, the outflow boundary part

$$\Gamma_{\text{out}} := \{x \in \partial\Omega : \mathbf{v}(x) \cdot n(x) > 0\} . \tag{10.2.1.12}$$



$\varepsilon > 0$: Dirichlet b.c. can be imposed everywhere on $\partial\Omega$

$\varepsilon = 0$: Dirichlet b.c. can be imposed only on Γ_{in}

⇓

singular perturbation

Notion 10.2.1.9. Singularly perturbed problem

A boundary value problem depending on parameter $\epsilon \approx \epsilon_0$ is called singularly perturbed, if the limit problem for $\epsilon \to \epsilon_0$ is not compatible with the boundary conditions.

并且，最后我们考虑一种特殊的情况，就是strealine中有closed loop，我们发现这种有closed loop的情况对于pure transport problem没有unique solution。这个需要尤其注意，其也会影响我们关于$\varepsilon \neq 0$的情况。

7.2.2 Upwinding

Recap: Singularly perturbed for $\varepsilon \to 0$

$\Omega \subset \mathbb{R}^d \triangleq$ bounded computational domain

Model problem :

$$-\epsilon\Delta u + \mathbf{v}(x) \cdot \mathbf{grad}\, u = f \quad \text{in } \Omega , \quad u = 0 \quad \text{on } \partial\Omega . \tag{10.2.0.1}$$

diffusive term (2nd-order term)

convective term (1st-order term)

singularly perturbed for $\varepsilon \to 0$

[$\varepsilon = 0$: Dirichlet b.c. can only be imposed on inflow boundary]

Therefore, the goal of this chapter is to find numerical methods that yield accurate solution for all $\varepsilon > 0$ ε-robust methods.

Goal (Dream): Numerical methods that yield accurate solutions for all $\varepsilon > 0$!

ε-robust methods

Robust discretization in the singular perturbation limit

We desire a **robust discretization** of (10.2.2.1)
= discretization that produces qualitatively correct $(*)$ solutions for **any** $\epsilon > 0$

$(*)$: "qualitatively correct", e.g., satisfaction of maximum principle, Thm. 10.1.3.13]

- 这里我们不需要完全定量的正确，定性的争取即可，即得满足structural properties: satisfaction of maximum principle.

接下来，我们考虑一个1D的问题，尝试用我们前面讨论的有限元方法去求解，会发生什么问题：

$$-\epsilon \frac{d^2u}{dx^2} + \frac{du}{dx} = f(x) \quad \text{in } \Omega\,, \quad u(0) = 0 \quad , \quad u(1) = 0\,. \tag{10.2.2.1}$$

$$u \in H_0^1(]0,1[): \quad \underbrace{\epsilon \int_0^1 \frac{du}{dx}(x)\,\frac{dv}{dx}(x)\,\mathrm{d}x + \int_0^1 \frac{du}{dx}(x)\,v(x)\,\mathrm{d}x}_{=:a(u,v)} = \underbrace{\int_0^1 f(x)v(x)\,\mathrm{d}x}_{=:\ell(v)} \quad \forall v \in H_0^1(]0,1[)\,.$$

Galerkin FE discretization based on $V_{0,h} = S^0_{1,0}(\mathcal{M})$,
$\mathcal{M} \hat{=}$ equidistant mesh of $]0,1[$, mesh width $h > 0$

$$\left(-\frac{\epsilon}{h} - \frac{1}{2}\right)\mu_{i-1} + \frac{2\epsilon}{h}\mu_i + \left(-\frac{\epsilon}{h} + \frac{1}{2}\right)\mu_{i+1} = hf(ih)\,, \quad i = 1,\ldots,M-1\,, \tag{10.2.2.2}$$

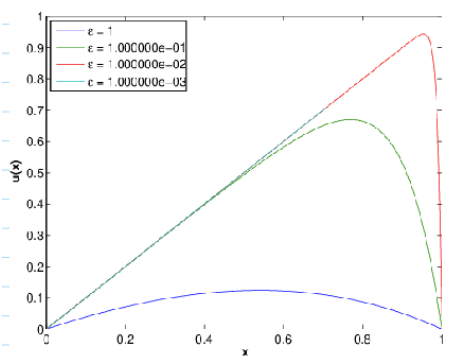
↑ ↑ ↑

tent-function basis expansion coefficients: $\mu_i = u_h(ih)$

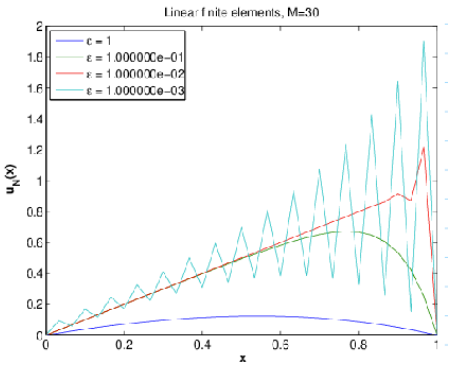
$$\begin{array}{ccccc} -\epsilon\frac{d^2u}{dx^2} & + & \frac{du}{dx} & = & f(x) \\ \updownarrow & & \updownarrow & & \updownarrow \\ \epsilon\underbrace{\frac{-\mu_{i+1}+2\mu_i-\mu_{i-1}}{h^2}}_{\text{difference quotient for } \frac{d^2u}{dx^2}} & + & \underbrace{\frac{\mu_{i+1}-\mu_{i-1}}{2h}}_{\text{symmetric d.q. for } \frac{du}{dx}} & = & f(ih)\,. \end{array} \tag{10.2.2.2}$$

- 这个地方表达的意思是如果我们用linear Lagrange basis，并且采用tensor-product mesh，有限元最后化简的形式就是有限差分。

通过一些数值试验，我们来看看这个方法的一些局限性：



exact solutions



FE solutions

我们可以看到，当$\varepsilon << 1 \rightarrow$ spurious oscillations，下面给出了关于这个现象的一些解释 建议再看看视频6:30仔细理解一下这一章节

(10.2.2.2) for $\varepsilon = 0$:

$$\mu_{i+1} - \mu_{i-1} = 2h f(ih), \quad i = 1, \ldots, M-1$$

$\rightarrow$ even-odd decoupling (singular LSE for even M)

- 这种算法求解时，当$\varepsilon \rightarrow 0$，会发生even-odd decoupling! (也就是奇数的节点和偶数的节点之间不发生信息的传递)并且M为even时，所有节点不依赖于end point，因此在end point上施加信息并不会改变节点上的解。

下面，我们如何解决这个问题?

§ 10.2.27 (Discretization of 1D limit problem)

Guideline: Robust numerical methods for singularly perturbed problems must cope with the limit problem

- 这里我们首先需要明白的一个道理是如果你想搞一种robust algorithm处理singularly perturbed problems，那么你的method同时也需要处理（cope with）limit problem。

那么我们看1D的limit problem情况。在1D的limit problem情况下，整个问提变成了一个ODE。对于这个ODE,我们有两种Euler method

Limit problem, $\varepsilon = 0$: $\frac{du}{dx}(x) = f(x) \triangleq$ ODE

Explicit Euler method: $\mu_{i+1} - \mu_i = hf(\xi_i) \quad i = 0, \ldots, M-1,$

Implicit Euler method: $\mu_{i+1} - \mu_i = hf(\xi_{i+1}) \quad i = 0, \ldots, M-1.$

Explicit Euler: $\frac{du}{dx}(x_i) \approx \frac{u(x_{i+1}) - u(x_i)}{h}$, Implicit Euler: $\frac{du}{dx}(x_i) \approx \frac{u(x_i) - u(x_{i-1})}{h}$.

[forward d.q.] [backward d.q.]

$\rightarrow$ one-sided d.q.

- 通过将两种方法化成the spirit of finite difference的格式，我们得到one-sided difference quotient，这区分于我们通过有限元得到的在(10.2.2,2)中的symmetric d.q.

下面，我们直接考虑在有限元得到的离散格式中，把其中的convection term的symmetric d.p.替换成两种one-sided d.q.，就得到如下LSE：

1. Linear system arising from *use of backward difference quotient* $\frac{du}{dx}|_{x=x_i} = \frac{\mu_i - \mu_{i-1}}{h}$:

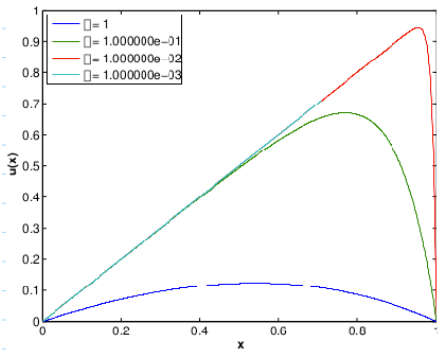
$$\left(-\frac{\epsilon}{h}-1\right)\mu_{i-1} + \left(\frac{2\epsilon}{h}+1\right)\mu_i + -\frac{\epsilon}{h}\mu_{i+1} = hf(ih)\,, \quad i = 1, \ldots, M-1\,, \qquad (10.2.2.8)$$

2. Linear system arising from *use of forward difference quotient* $\frac{du}{dx}|_{x=x_i} = \frac{\mu_{i+1} - \mu_i}{h}$:

$$-\frac{\epsilon}{h}\mu_{i-1} + \left(\frac{2\epsilon}{h}-1\right)\mu_i + \left(-\frac{\epsilon}{h}+1\right)\mu_{i+1} = hf(ih)\,, \quad i = 1, \ldots, M-1\,, \qquad (10.2.2.9)$$

下面我们做数值试验和真实的解比较一下：

Experiment 10.2.2.10 (One-sided difference approximation of convective terms)

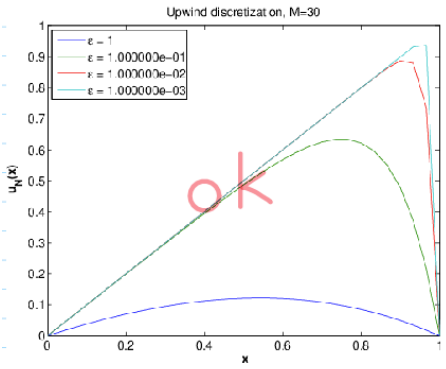


$f \equiv 1$:

We revisit the model problem of Ex. 10.2.2.4.

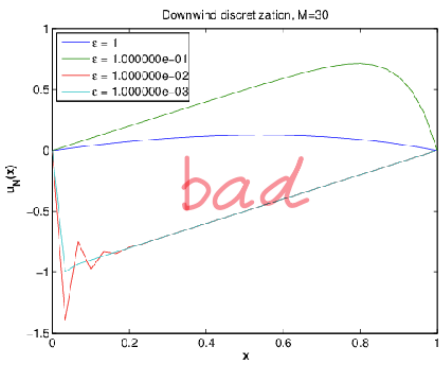
◁ exact solutions for different values of ϵ

We study the solutions obtained by the discretizations (10.2.2.8) and (10.2.2.9).



backward difference quotient

respects maximum principle



forward difference quotient

violated for small ϵ

- 可以明显看到backward d.q.效果更好，并且满足maximum principle。下面，我们想从数学上的角度说明这种数值解法理论上是满足maximum principle的：

我们回忆3.7章节

§ 10.2.2.11 (Selection criterion : Discrete maximum principle)

(Linearly interpolated) discrete solution satisfies maximum principle (3.7.2.1).
⇕
System matrix complies with sign-conditions (3.7.2.8)–(3.7.2.10).

- (3.7.2.8): positive diagonal entries. $(\mathbf{A})_{ii} > 0$,
- (3.7.2.9): non-positive off-diagonal entries, $(\mathbf{A})_{ij} \leq 0$, if $i \neq j$,
- "(3.7.2.10)": diagonal dominance, $\sum_j (\mathbf{A})_{ij} \geq 0$.

- 我们再分析一下两种d.p.的system matrix:

$\leq 0 \ \forall \varepsilon, h$ $\quad > 0 \ \forall \varepsilon, h$

1. Linear system arising from *use of backward difference quotient* $\left.\frac{du}{dx}\right|_{x=x_i} = \frac{\mu_i - \mu_{i-1}}{h}$:

$$\left(-\frac{\epsilon}{h} - 1\right)\mu_{i-1} + \left(\frac{2\epsilon}{h} + 1\right)\mu_i + -\frac{\epsilon}{h}\mu_{i+1} = hf(ih) \ , \quad i = 1, \ldots, M-1 \ , \qquad (10.2.2.8)$$

2. Linear system arising from *use of forward difference quotient* $\left.\frac{du}{dx}\right|_{x=x_i} = \frac{\mu_{i+1} - \mu_i}{h}$:

$$-\frac{\epsilon}{h}\mu_{i-1} + \left(\frac{2\epsilon}{h} - 1\right)\mu_i + \left(-\frac{\epsilon}{h} + 1\right)\mu_{i+1} = hf(ih) \ , \quad i = 1, \ldots, M-1 \ , \qquad (10.2.2.9)$$

> 0 only if $h < 2\varepsilon$

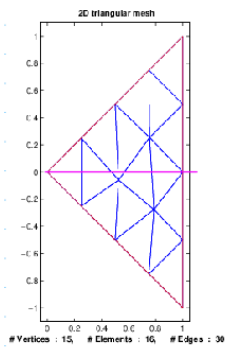
Upwinding

- 我们把这种满足maximum principle的one-sided d.q称之为upwind

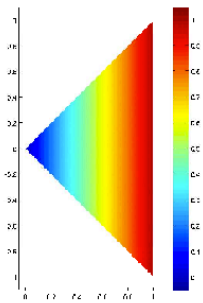
下面，我们有考虑一个2D problem:

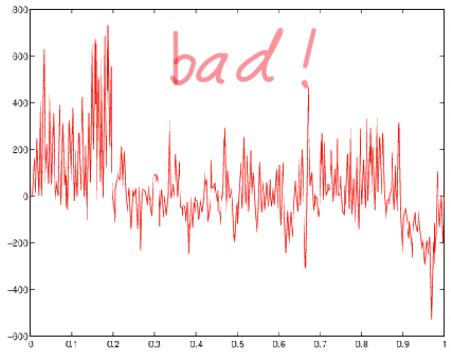
Exp. 10.2.2.12 (Galerkin FEM for CD-BVP in 2D)

- Triangle domain $\Omega = \{(x,y) : 0 \leq x \leq 1, -x \leq y \leq x\}$.
- Velocity $\mathbf{v}(x) = \binom{1}{0}$ ➢ (10.2.0.1) becomes $\epsilon \Delta u \ | \ u_x = 1$.
- Exact solution: $u_\epsilon(x_1, x_2) = x - \frac{1}{1-e^{-1/\epsilon}}(e^{-(1-x_1)/\epsilon} - e^{-1/\epsilon})$, Dirichlet boundary conditions set accordingly
- Standard Galerkin discretization by means of linear finite elements on sequence of triangular mesh created by regular refinement.



Coarse initial mesh


Exact solution ($\epsilon = 10^{-10}$)



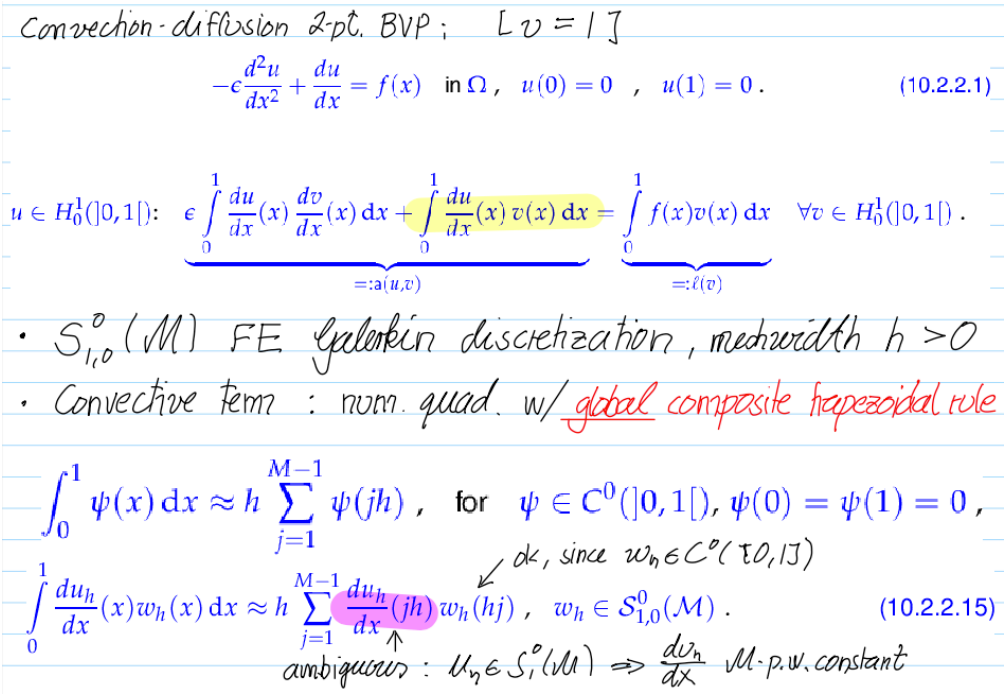
Standard Galerkin solution on $x_2 = 0$-line

- 对于convection term，我们还是采用和1D的upwind，但是效果很差(这里应该想表达的是我们用FEM的效果很差，而怎么在high dimension的问题中使用upwind是一个需要考虑的问题），怎么解决这个问题是我们接下来两章节要讨论的问题。

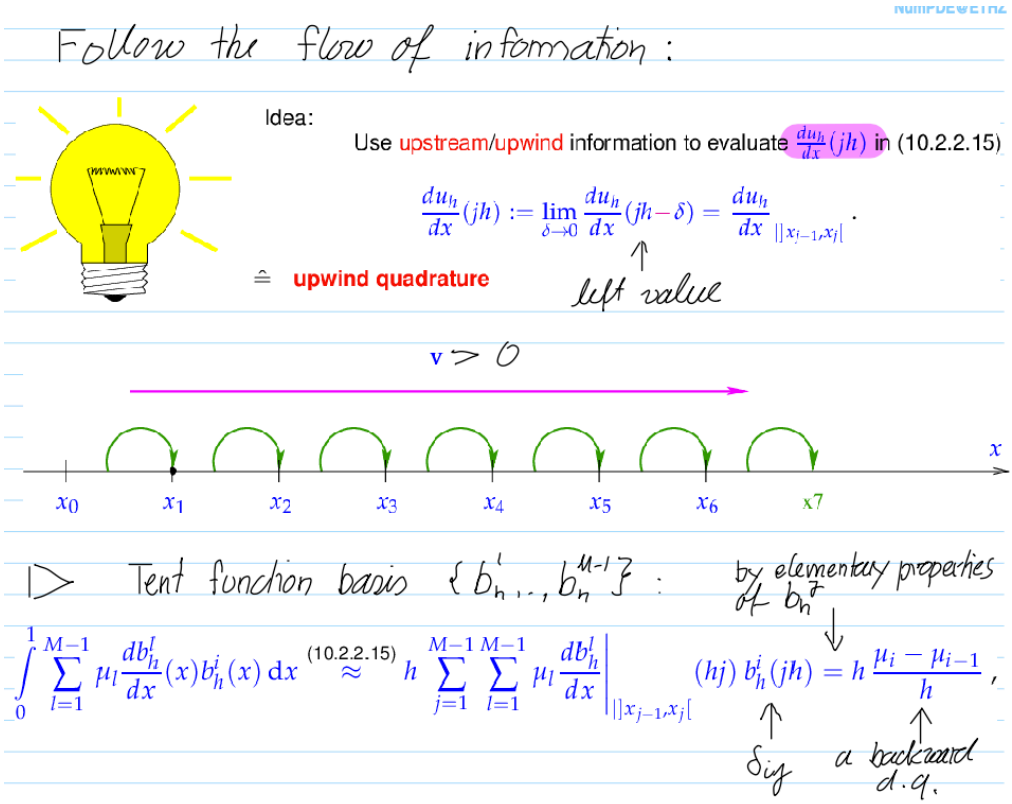
7.2.2.1 Upwind Quadrature

general upwind idea to higher dimension.

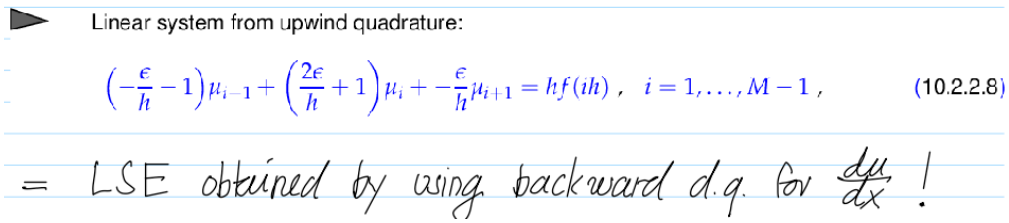
我们还是拿一维的例子来举例和推导，不过这一次，我们从另一个角度推导出upwind scheme。



- 也就是不同于在得到weak form之后直接将方程离散化，我们这里对于convective term，采用numerical quadrature去表达，而不是直接离散。
- 这里，我们采用的numerical quadrature不同于之前我们有限元在单元里面做local quadrature，在这个问题中，我们是用numerical quadratuer rule在整个domian上做计算。因此我们会得到10.2.2.15的形式。但是，我们需要注意的是这个numerical quadrature通常适用于continuous function，而对于10.2.2.15，里面一个关键的问题是u_h是piece-wise continuous，这就导致导数在不同网格边界处有jump，那么我们到底去左边的值还是取右边的值呢？**这里给出的一个策略就是，找到information的流向，选择上流的点作为我们的值**



- 最后，我们发现，如果采用上流的信息提取，convective term刚好就化成了我们之前讨论的backward d.q.:



下面我们将这个思路general到2D problem上面：

$\Omega \subset \mathbb{R}^2$, LVP for CD-BVP:

$$u \in H_0^1(\Omega): \quad \underbrace{\epsilon \int_\Omega \mathbf{grad}\, u \cdot \mathbf{grad}\, w \,\mathrm{d}x + \int_\Omega (\mathbf{v} \cdot \mathbf{grad}\, u)\, w \,\mathrm{d}x}_{\text{bilinear form } a(u,w)} = \underbrace{\int_\Omega f(x)\, w(x) \,\mathrm{d}x}_{\text{linear form } \ell(w)} \quad \forall w \in H_0^1(\Omega)\,. \tag{10.2.0.3}$$

+ $S^0_{1,0}(\mathcal{M})$ Galerkin FEM, $\mathcal{M} \hat{=}$ triangular mesh

I. Use global composite trapezoidal rule

$$\int_\Omega \psi(x)\,\mathrm{d}x = \sum_{K\in\mathcal{M}} \int_K \psi(x)\,\mathrm{d}x \approx \sum_{K\in\mathcal{M}} \frac{|K|}{3}(\psi(a_K^1)+\psi(a_K^2)+\psi(a_K^3)) \tag{10.2.2.17}$$

$$\approx \sum_{p\in\mathcal{V}(\mathcal{M})} \Big(\frac{1}{3}\sum_{K\in\mathcal{U}_p} |K|\Big)\,\psi(p)\,,$$

↑ area of node patch

for convective part of $a(\cdot,\cdot)$:

$$\int_\Omega (\mathbf{v}\cdot\mathbf{grad}\, u_h) w_h \,\mathrm{d}x \approx \sum_{p\in\mathcal{V}(\mathcal{M})} \Big(\frac{1}{3}\sum_{K\in\mathcal{U}_p} |K|\Big) \cdot \mathbf{v}(p)\cdot\mathbf{grad}\, u_h(p)\, w_h(p)\,. \tag{10.2.2.18}$$

ambiguous for $u_h \in S_1^0(\mathcal{M})$!

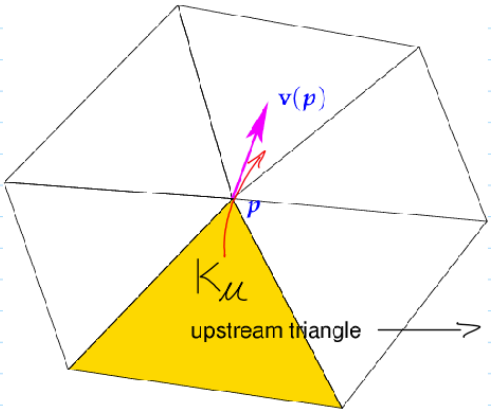
✎ notation: $\mathcal{U}_p := \{K \in \mathcal{M}: p \in \overline{K}\}$, $p \in \mathcal{V}(\mathcal{M})$

II. Follow flow of information:

Idea: Use upstream/upwind information to evaluate $\mathbf{grad}\, u_h(p)$ in (10.2.2.18)

$$\mathbf{v}(p)\cdot\mathbf{grad}\, u_h(p) := \lim_{\delta\to 0} \mathbf{v}(p)\cdot\mathbf{grad}\, u_h(p - \delta\mathbf{v}(p))\,. \tag{10.2.2.19}$$

$\hat{=}$ general **upwind quadrature**. (UWQ)



$\mathrm{grad}\, u_h(p) \leftarrow \mathrm{grad}\, u_{h|K_\mu}(p)$

K_μ → triangle where information comes from

- 这里有两个点需要注意，一是对于二维的问题，global composite trapezoidal rule的形式是啥样的
- 而就是上游的点怎么取。
- 这里的推导稍微有点不清楚，之后再研究一下

(i) Detailed examination :

Contribution of UWQ convective term to i-th row of FE Galerkin matrix

$$\int_\Omega (\mathbf{v}\cdot\mathbf{grad}\, u_h) w_h \,\mathrm{d}\mathbf{x} \approx \sum_{p\in\mathcal{V}(\mathcal{M})} \left(\frac{1}{3}\sum_{K\in\mathcal{U}_p}|K|\right) \cdot \mathbf{v}(p)\cdot\mathbf{grad}\, u_h(p)\, w_h(p)\,. \qquad (10.2.2.18)$$

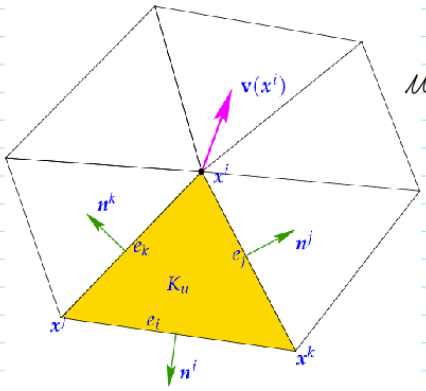
ambiguous for $u_h \in \mathcal{S}_1^0(\mathcal{M})$!

notation: $\mathcal{U}_p := \{K\in\mathcal{M}: p\in\overline{K}\}$, $p\in\mathcal{V}(\mathcal{M})$

$\mathbf{grad}\, u_h|_{K_u}$

$b_h^i \to w_h$

$$\underbrace{\left(\frac{1}{3}\sum_{K\in\mathcal{U}_i}|K|\right)}_{=:U_i} \mathbf{v}(x^i)\cdot\mathbf{grad}\, u_h|_{K_u}$$



$u_h|_{K_u} = \mu_j \lambda_j^k + \mu_k \lambda_k^k + \mu_i \lambda_i^k$

$\mathbf{grad}\,\lambda_*^k = \frac{|e_i|}{2|K|} n^*$

$* = i, j, k$

然后，我们在用3.7章节的system matrix进行分析，得到其是满足discrete maximum principle的：

▷ i-th row, convective contribution :

$$\frac{U_i}{2\,K_u}\left(\underbrace{-\left|\mathbf{x}^j-\mathbf{x}^k\right| n^i\cdot\mathbf{v}(x^i)}_{\leftrightarrow \text{diagonal entry}} \mu_i - \left|\mathbf{x}^i-\mathbf{x}^j\right| n^k\cdot\mathbf{v}(x^i)\mu_k - \left|\mathbf{x}^i-\mathbf{x}^k\right| n^j\cdot\mathbf{v}(x^i)\mu_j \right) = \ldots$$

< 0 $\quad \geq 0$ $\quad \geq 0$

⇔ Sign conditions for discrete maximum principle :

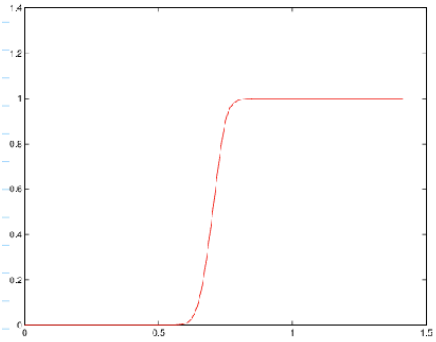
- (3.7.2.8): positive diagonal entries, $(\mathbf{A})_{ii} > 0$,
- (3.7.2.9): non-positive off-diagonal entries, $(\mathbf{A})_{ij} \leq 0$, if $i \neq j$,
- "(3.7.2.10)": diagonal dominance, $\sum_j (\mathbf{A})_{ij} \geq 0$.

我们通过数值试验，确实证明了算法的稳定性：

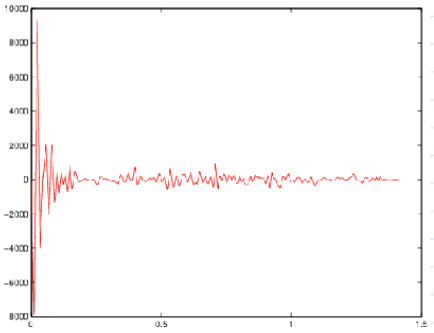
Exp. 10.2.2.20 (Upwind quadrature discretization)

- Computational domain: unit square $\Omega = [0,1]^2$
- $-\epsilon\Delta u + \binom{1}{1}\cdot\mathbf{grad}\, u = 0$
- Dirichlet boundary conditions: $u(x,y) = 1$ for $x > y$ and $u(x,y) = 0$ for $x \leq y$
- Limiting case ($\epsilon \to 0$): $u(x,y) = 1$ for $x > y$ and $u(x,y) = 0$ for $x \leq y$
- layer along the diagonal from $\binom{0}{1}$ to $\binom{1}{0}$ in the limit $\epsilon \to 0$
- 2D triangular Delaunay triangulation, see Rem. 4.2.2.3
- linear finite element upwind quadrature discretization

⑤ $\varepsilon = 10^{-10}$:



upwind quadrature solution



standard Galerkin solution

no spurious oscillations

7.2.2.2 Streamline Diffusion

我们仍旧处理的是stationary的CD。

10.2.2. Upwinding

10.2.2.2. Streamline Diffusion

$\Omega \subset \mathbb{R}^d \triangleq$ bounded computational domain

Model problem :

$$-\varepsilon\Delta u + \mathbf{v}(x)\cdot\mathbf{grad}\, u = f \quad \text{in } \Omega, \quad u = 0 \quad \text{on } \partial\Omega. \qquad (10.2.0.1)$$

diffusive term (2nd-order term)

convective term (1st-order term)

singularly perturbed for $\varepsilon \to 0$

Desirable : ε-robust discretization

这里，我们仍旧循着上一章的思路，先分析1D的问题，进行一些推导后再generalize到高维。这里引出了另一种预防singularly perturbed的方法，不同于上一章的upwind quadrature。这里，我们做的就是artificial viscosity，就是人工给diffusion term前面加一个很小的量来避免diffusion term趋近于零：

§ 10.2.2.22 (Artificial viscosity in 1D)

1D convection-diffusion model problem, $\Omega =]0,1[$

$$-\epsilon\frac{d^2u}{dx^2} + \frac{du}{dx} = f(x) \quad \text{in } \Omega, \quad u(0) = 0 \quad , \quad u(1) = 0 .$$

Another view of 1D upwinding :

$$\left(-\frac{\epsilon}{h} - 1\right)u_{i-1} + \left(\frac{2\epsilon}{h} + 1\right)u_i + -\frac{\epsilon}{h}u_{i+1} = hf(ih) , \quad i = 1,\ldots,M-1 . \qquad (10.2.2.8)$$

$$\Updownarrow$$

$$(\epsilon + h/2)\underbrace{\frac{u_{i-1} - 2u_i + u_{i+1}}{h^2}}_{\doteq \text{difference quotient for } \frac{d^2u}{dx^2}} + \underbrace{\frac{u_{i-1} - u_{i+1}}{2h}}_{\doteq \text{difference quotient for } \frac{du}{dx}} = f(ih) ,$$

Upwinding = h-dependent enhancement of diffusive term
(This is widely known as **artificial diffusion/viscosity**)

→ CFD

Extension to $d > 1$?

Simply add $h \cdot (-\Delta)$?

- 对于1D的问题，我们也确实这样做了。但是对于高维的问题，我们还能这样简单操作吗?
- 答案是否定的。

对于高维的问题，我们先来看看有diffusion的作用和影响是什么（因为我们要给diffusion term前面再加一个人工的diffusion term，所以我们自然就想分析多的term会造成什么后果）

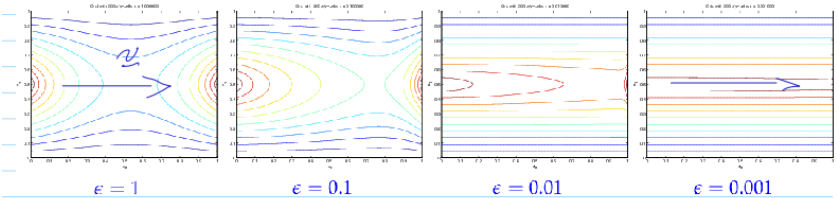
Ex. 10.2.2.23 (Effect of added diffusion)

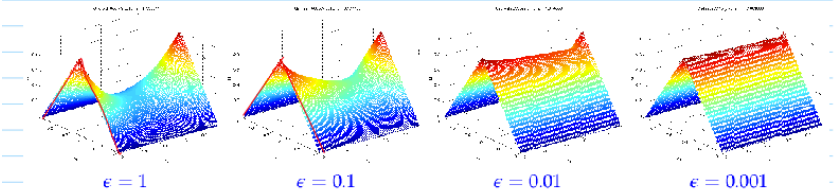
2D CD-BVP :

$\underline{v} = [1]$:

$$-\epsilon\Delta u + \frac{\partial u}{\partial x_1} = 0 \quad \text{in } \Omega =]0,1[^2 , \quad u = g \quad \text{on } \partial\Omega .$$

Here, the Dirichlet data are $g(\boldsymbol{x}) = 1 - 2|x_2 - \frac{1}{2}|$ ("roof function").



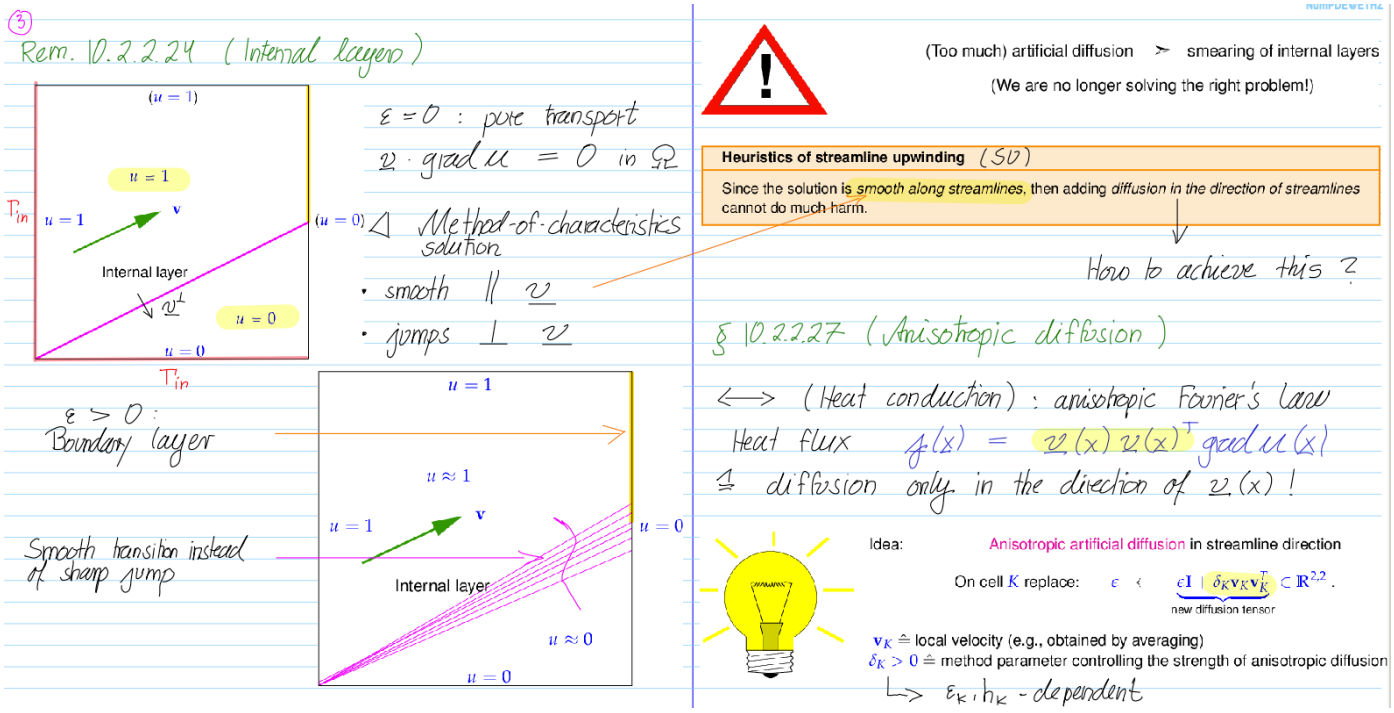


[Large diffusion] → smooth solution

[Tiny diffusion] → sharp internal ridge

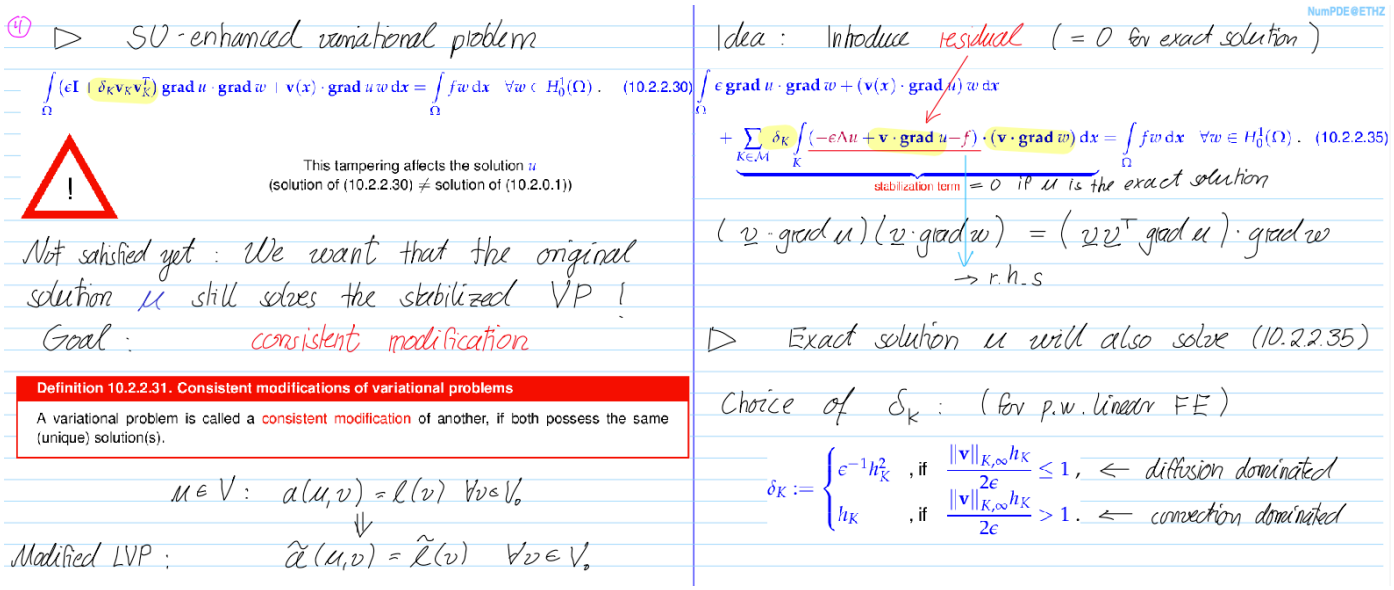
Enhancing diffusion ⇒ smearing of internal layers and ridges

- 通过这个例子，我们可以看出enhancing diffusion会smear（模糊）求解域内的ridges。
- 下面那个例子我们会看到smear internal layers

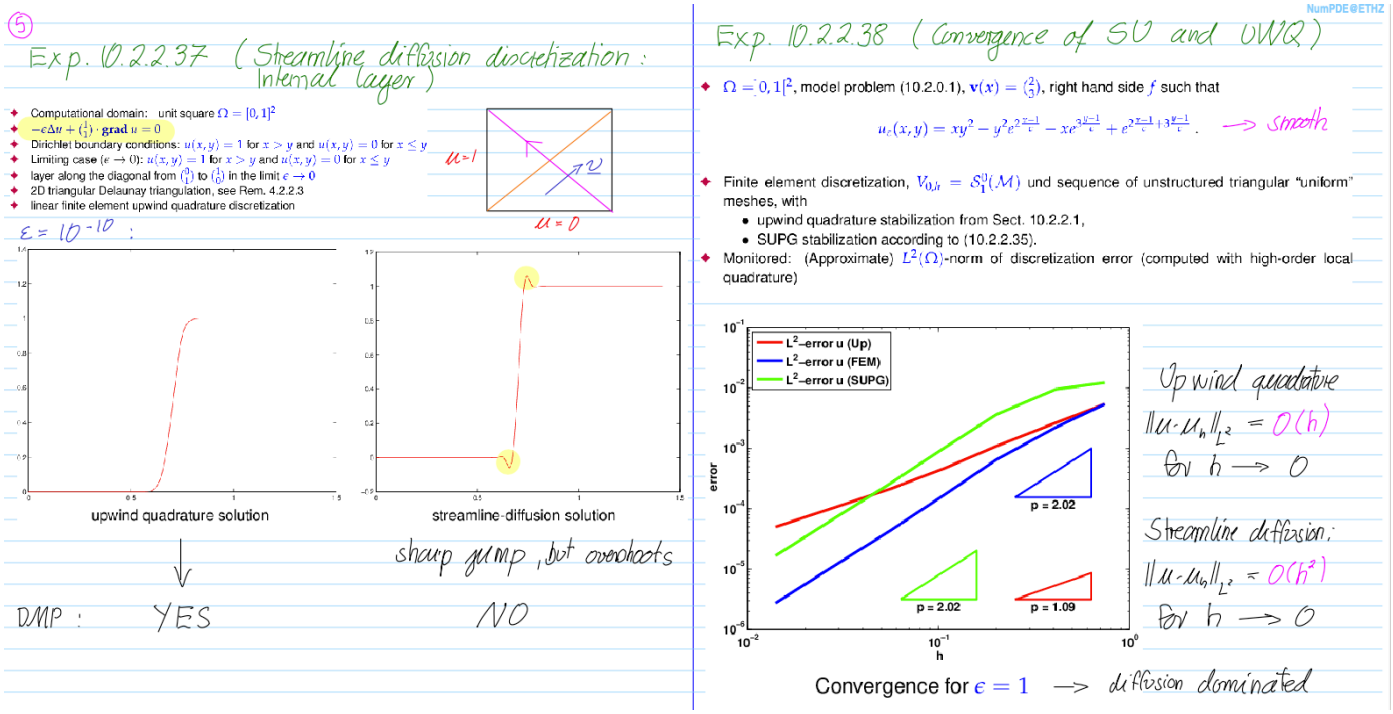


- 对于这个有internal layer的例子，我们不想让额外加的artificial diffusion影响我们的解，因此我们提出了各向异性地条件diffusion，也就是旨在stramline的方向上添加diffusion这个地方不是很懂，但大概知道这个意思，之后再看看

然后又会有人反驳，你加了diffusion term，你的问题就不是原来的问题了。这里，我们又提出了consistent modification的概念，就是修改后的变分问题的解仍是原来问题的解。这里我们给出的思路是引入residual（就是真实解时residual为0）并且给出了δ_K的选取标准：



至此，我们的方法部分结束，下面我们来看看数值试验的结果：



从左边那张图，我们可以看到，steamline-diffusion 的solution不满足discrete maximum principle，那为什么我们还要选择用这个呢？为什么我们不用upwind quadrature呢？原因是，当你在convection term引入upwind后，会破坏原本有限元的收敛速度，本来有限元的收敛速度时(h^2)。而streamline则不会破坏。

这里，我们还要告诉你的一个道理是：

"Meta theorem":
No ϵ-robust linear method that respects the maximum principle strictly can be more than 1st-order convergent in L^2!

- 也就是至少对于linear 的method（也就是最后求解的是LSE），你不得不面临这个tradeoff。

7.3 Discretization of Transient Convection-Diffusion IBVPs

introduce a class of numerical methods for the discretization of time-dependent CD problems. 这里面的核心问题同样是处理singular perturbed case

首先先回归一下我们的问题：

IBVP describing non-stationary heat conduction in a fluid flowing with time-dependent velocity

$$\underline{v} : \widetilde{\Omega} := \Omega \times]0,T[\longrightarrow \mathbb{R}^d :$$

Temperature $u : \widetilde{\Omega} \longrightarrow \mathbb{R}$, $u = u(x,t)$ satisfies

$$\frac{\partial}{\partial t}(\rho u) + \operatorname{div}(-\kappa \operatorname{grad} u + \rho \underline{v} u) = f \text{ in } \widetilde{\Omega}$$

↑ heat source, $f = f(x,t)$

② + boundary conditions, e.g. $u(x,t) = g(x,t)$, $x \in \partial\Omega$

+ initial conditions: $u(x,0) = u_0(x)$, $x \in \Omega$

$\Downarrow$ $\operatorname{div}_x \underline{v} = 0$, rescaling

$$\frac{\partial u}{\partial t} - \varepsilon \Delta u + \underline{v}(x,t) \cdot \operatorname{grad} u = f \text{ in } \widetilde{\Omega} \quad (*)$$

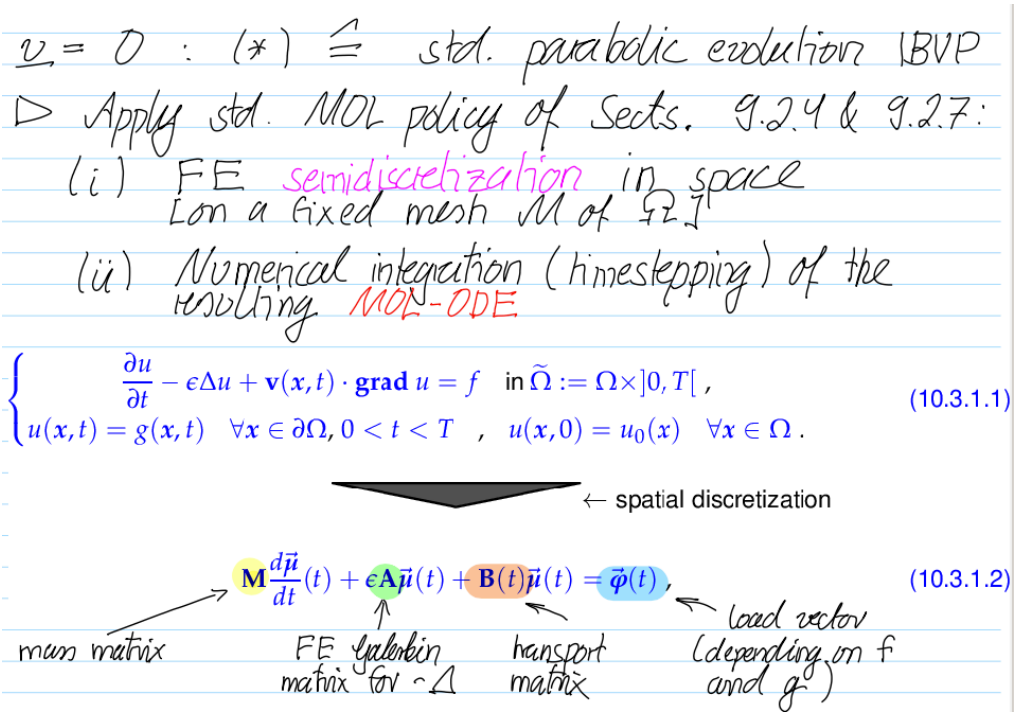
↑ can large on some parts of $\widetilde{\Omega}$: singular perturbation

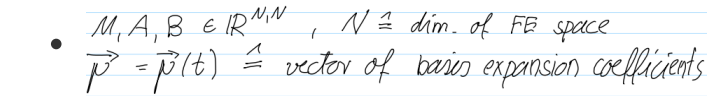
Challenge: $\varepsilon / \|\underline{v}\|$ - robust discretization

- 当我们考虑incompressible flow，同时将方程无量纲化后，我们得到我们上一章讨论的方程形式在这个方程中，我们需要考虑的是singular pertubation，因为速度是时空相关的，不像我们之前讨论的常数形式。因此，他可能会在local部分远远大于diffusion term，同样会引起pertubation，所以我们需要引入robust discretization。

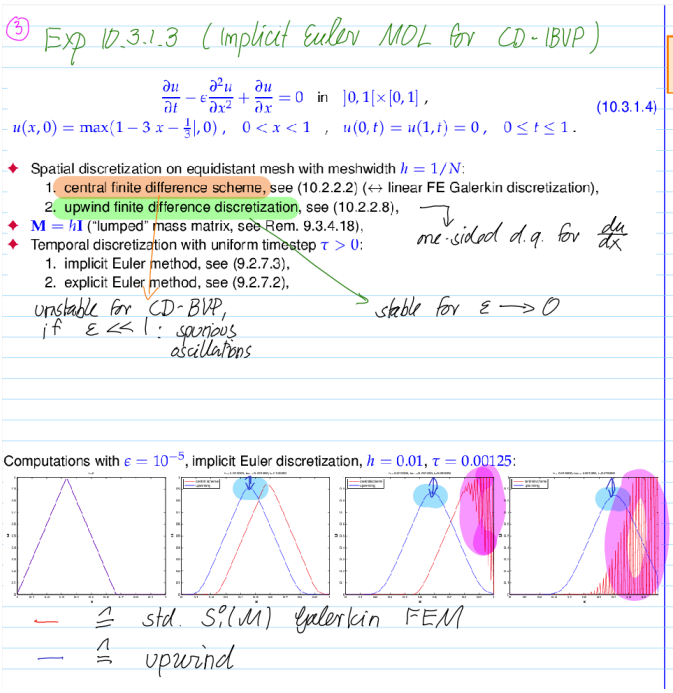
7.3.1 Method of Lines (MOL)

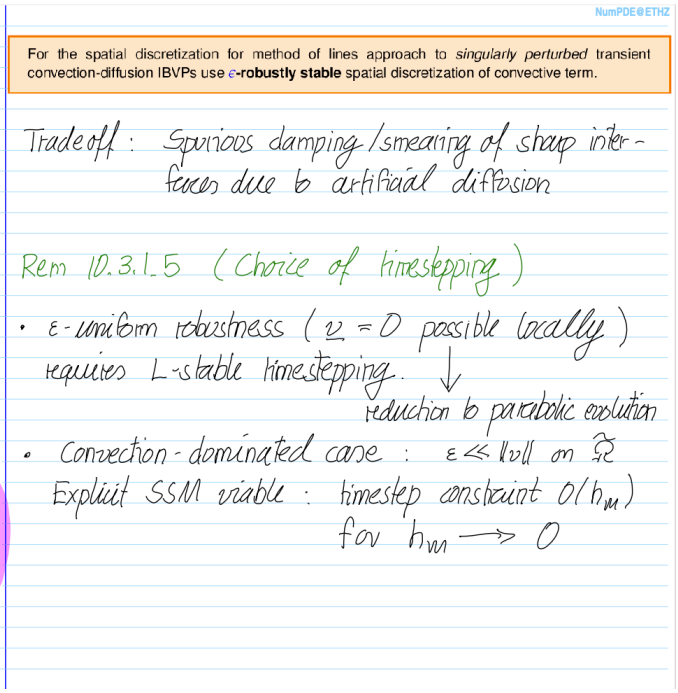
这里使用第九章我们解决parabolic PDEs的方法。其实相对于parabolic PDEs，我们就只多了一项convective term。



- 多的这一项我们称之为transport matrix。
- 

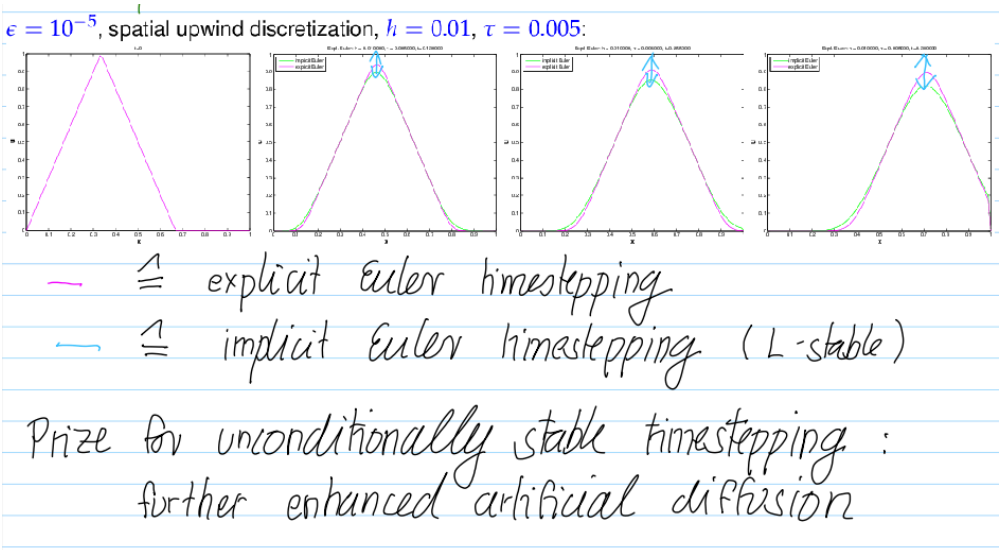
接下来我们做做数值试验看看这个方法的效果：





- 对于convective term，我们采样了两种方法，并做了比较：
  - 可以看到central FD会导致spurious oscillations。而upwind FD虽然不会有oscillations，但是，会due to the artificial diffusion introduced by upwinding. artificial diffusion不会另一种方法吗？怎么也和upwind扯上了关系。而这里的artificial diffusion又会引入spurious damping（也就是解会下降，这在这个问题中不合理，因为这个问题中速度场很大，所以基本不会衰减从左传到右）第二个是smearing，也就是尖端逐渐被磨平。**但是，这种tradeoff是值得的，总比oscillation要好**

- 另一个是关于time-stepping method的，在第九章，我们知道parabolic problem我们应该使用L-stable timestepping。为了避免CD出现局部的parabolic problem，我们使用L-stable。但是，如果我们知道在整个求解域上都是convection-domainated case的话，我们也可以用explicit SSM，然后timestep有constraint?

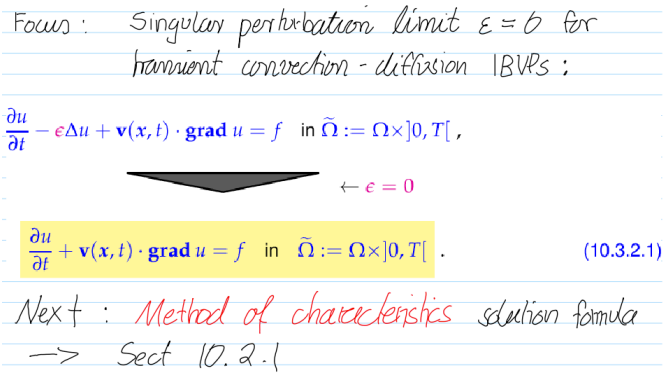


- 这个图告诉我们了$L-stable$也不是完全好的，也会引入artificial diffusion，这也会影响我们模型的smearing。但是，这些是可以接受的，因为一旦有oscillation，影响的是我们整个求解。
- a way to mitigate the artificial diffusion is to use higher-order methods. We won't discuss here.

7.3.2 Transport Equation

这一章是为special discretization method for transient CD problems做一个预先的铺垫。这个地方考虑的还是针对singular perturbed problem，在这里，我们假设$\varepsilon = 0$,pure convection equation (also known as pure transport equation.)

在前面的章节中，我们讨论过stationary的情况，当时用了method of characteristics从而得到了explicit solution。Now, we focused on the transient situation.



method of characteristics的关键是follow the solution along the streamline.在这我们流体力学中称之为Lagrangian view，也就是track behavior of u (concerned physical variables) along trajectories of fluid particles.

② Lagrangian view : track behavior of u along trajectories of fluid particles

↳ $t \to \gamma(t)$: $\dot{\gamma}(t) = v(\gamma(t), t)$ (streamline ODE)

$$\frac{d}{dt} u(\gamma(t), t) \overset{\text{chain rule}}{=} \operatorname{grad}_x u(\gamma(t), t) \cdot \dot{\gamma}(t) + \frac{\partial u}{\partial t}(\gamma(t), t)$$
$$= \operatorname{grad}_x u(\gamma(t), t)\, v(\gamma(t), t) + \frac{\partial u}{\partial t}(\gamma(t), t)$$
$$(10.3.2.1) \Rightarrow = f(\gamma(t), t)$$

- 可以看到，x空间维度不能让他取任意，而是对其做约束，让其为一个函数，也就是我们这里的trajectory。然后求解的是物理量沿着这个trajectory的解。

下面我们先来考虑一种简单的情况，看看解的特点： Solution formula for sourceless transport

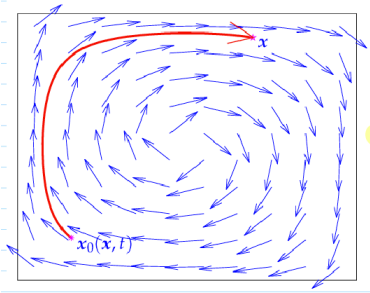
Now : $f = 0 \Rightarrow \frac{d}{dt} u(\gamma(t), t) = 0$

If $f \equiv 0$, then a fluid particle "sees" a constant temperature!

Also assume : $\vec{v} \cdot \vec{n} = 0$ on $\partial\Omega$

[No in/outflow]

- u is a constant along the trajectory.



u constant on particle trajectories

$\Downarrow$

$u(x,t) = u_0(x_0(x,t))$

$x_0(x,t) \hat{=}$ starting point of fluid particle in $x \in \Omega$ at t.

$\triangleright$ u by backtracking along streamlines

- 也就是整个问题的解如上面高亮那种形式。
- 注意，在stationary的情况下，我们只有一个边界条件，也就是inflow的BC。而现在，对于transietn情况，我们需要有两个，initial 和Boundary。而上面那个解其实是initial condition决定的?

下面我们考虑general 的解:

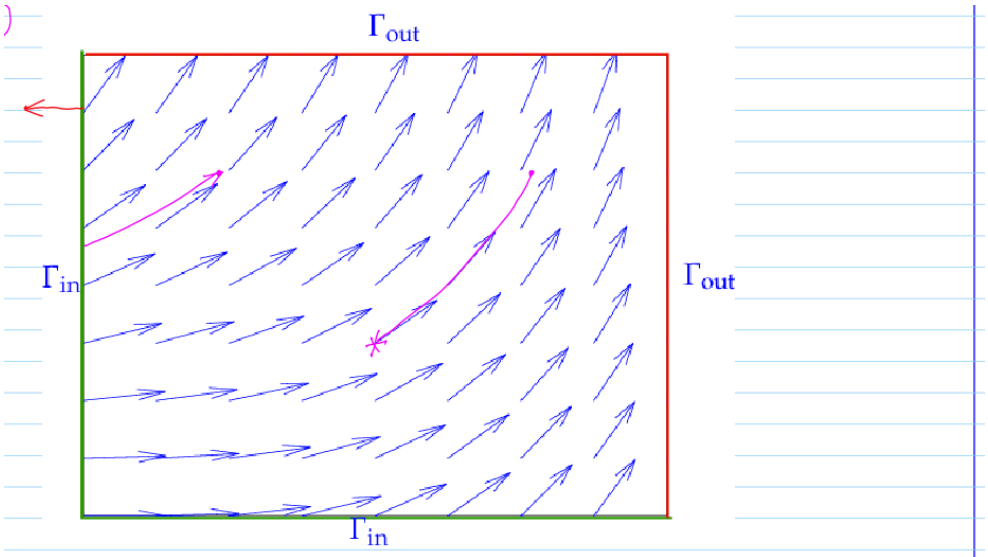
§ 10.3.2.6 (Method-of-characteristics solution formula)

Admit general f, v :

Dirichlet b.c. on inflow boundary :

$u(x,t) = g(x,t)$ for $x \in \Gamma_{\text{in}} := \{x \in \partial\Omega : \mathbf{v}(x) \cdot n(x) < 0\}$, *cf.* (10.2.1.11).

↳ where streamlines enter Ω



$$\frac{d}{dt}u(\mathbf{y}(t)) = f(\mathbf{y}(t),t) \quad \text{where} \quad \dot{\mathbf{y}}(t) = \mathbf{v}(\mathbf{y}(t),t)\,.$$

$$u(x,t) = \begin{cases} u_0(x_0) + \int_0^t f(\mathbf{y}(s),s)\,\mathrm{d}s & \text{, if } \mathbf{y}(s) \in \Omega \quad \forall 0 < s < t\,, \\ g(\mathbf{y}(s_0),s_0) + \int_{s_0}^t f(\mathbf{y}(s),s)\,\mathrm{d}s & \text{, if } \mathbf{y}(s_0) \in \partial\Omega,\ \mathbf{y}(s) \in \Omega \quad \forall s_0 < s < t\,, \end{cases} \tag{10.3.2.7}$$

streamline starts inside Ω

streamline starts at boundary in $\mathbf{y}(s_0)$

- 这里，u应该还要和时间有关。也就是无论是否是stationary situation，控制方程就被简化成u沿着tranjectory关于时间的导数为右端项关于tranjectory。
- 然后解又分为两部分，起源于BC的和起源于internal。话说不太理解stramline如何起源于internel? streamline可以不闭合?

7.3.3 Lagrangian Split-Step Method

In this section we develop an alternative to the method-of-lines approach, which belongs to the class of Lagrangian discretization schemes. Lagrangian discretization schemes for the IBVP (7.3.1.1) are inspired by the method-of-characteristics solution formulas (7.3.2.5) and (7.3.2.7).

The variant that we are going to study separates the transient convection-diffusion problem into a pure diffusion problem (heat equation → Section 6.2.1) and a pure transport problem (7.3.2.1). This is achieved by means of a particular approach to timestepping that will be introduced next.

For the title, the lagrangian means: based on method of characteristics ("particle tracking")

Split-step: separate treatment of convection and diffusion.

7.3.3.1 Split-Step Timestepping

First, we learn how to achive the split-step timestepping. we have seen this before in the numerical integration.

r.h.s = sum of two functions

$$\dot{\mathbf{y}} = \mathbf{g}(t,\mathbf{y}) + \mathbf{r}(t,\mathbf{y}) \,, \quad \mathbf{g},\mathbf{r} : [0,T] \times \mathbb{R}^m \mapsto \mathbb{R}^m \,. \tag{10.3.3.1}$$

Idea: In turns solve (1) $\dot{\mathbf{y}}(t) = \mathbf{g}(t,\mathbf{y})$, (2) $\dot{\mathbf{y}}(t) = \mathbf{r}(t,\mathbf{y})$, over small timesteps.

- 在数值integration，我们选择做split的原因是，右端两项，如果不拆开，则会遇到stiffness等问题，但是拆开后，我们甚至可以用两个解析的解来进行求解。

Let us introduce the evolution operators (→ Def. 11.1.4.3) for both summands:

(Continuous) evolution maps: $\boldsymbol{\Phi}_f^t \leftrightarrow$ ODE $\dot{\mathbf{y}} = \mathbf{f}(\mathbf{y})$, $\boldsymbol{\Phi}_g^t \leftrightarrow$ ODE $\dot{\mathbf{y}} = \mathbf{g}(\mathbf{y})$.

Temporarily we assume that both $\boldsymbol{\Phi}_f^t$, $\boldsymbol{\Phi}_g^t$ are available in the form of analytic formulas or highly accurate

- or highly accurate approximations

Idea: Build single step methods (→ Def. 11.3.1.5) based on the following discrete evolutions

Lie-Trotter splitting: $\boldsymbol{\Psi}^h = \boldsymbol{\Phi}_g^h \circ \boldsymbol{\Phi}_f^h$, (12.5.0.3)

Strang splitting: $\boldsymbol{\Psi}^h = \boldsymbol{\Phi}_f^{h/2} \circ \boldsymbol{\Phi}_g^h \circ \boldsymbol{\Phi}_f^{h/2}$. (12.5.0.4)

These splittings are easily remembered in graphical form:

(12.5.0.3) ↔

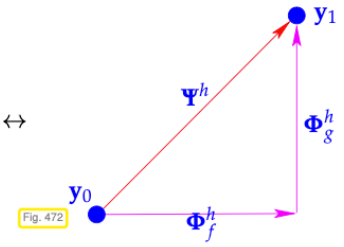


Fig. 472

(12.5.0.4) ↔

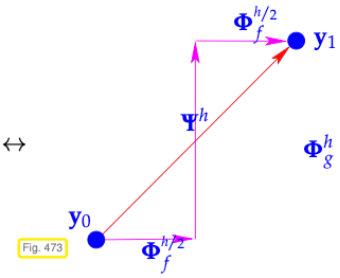


Fig. 473

在复习了strange splitting，我们来看我们这里的问题：

② More details : [temporal mesh $0 = t_0 < t_1 < t_2 < \dots < t_M = T$]

Strang splitting single step method for (10.3.3.1), timestep $\tau := t_j - t_{j-1} > 0$:
compute $\mathbf{y}^{(j)} \approx \mathbf{y}(t_j)$ from $\mathbf{y}^{(j-1)} \approx \mathbf{y}(t_{j-1})$ according to

① $\tilde{\mathbf{y}} := \mathbf{z}(t_{j-1} + \frac{1}{2}\tau)$, where $\mathbf{z}(t)$ solves $\dot{\mathbf{z}} = \mathbf{g}(t,\mathbf{z})$, $\mathbf{z}(t_{j-1}) = \mathbf{y}^{(j-1)}$, (10.3.3.2)

② $\hat{\mathbf{y}} := \mathbf{w}(t_j)$ where $\mathbf{w}(t)$ solves $\dot{\mathbf{w}} = \mathbf{r}(t,\mathbf{w})$, $\mathbf{w}(t_{j-1}) = \tilde{\mathbf{y}}$, (10.3.3.3)

③ $\mathbf{y}^{(j)} := \mathbf{z}(t_j)$, where $\mathbf{z}(t)$ solves $\dot{\mathbf{z}} = \mathbf{g}(t,\mathbf{z})$, $\mathbf{z}(t_{j-1} + \frac{1}{2}\tau) = \hat{\mathbf{y}}$. (10.3.3.4)

↳ practise : by another SSM

Theorem 10.3.3.5. Order of Strang splitting single step method

Assuming exact solution of the initial value problems of the sub-steps, the Strang splitting single step method for (10.3.3.1) is of second order.

- 这里和我们在Numerical Integration里面不同的一点是这里的sub-steps IVP不能够通过analytic form等求解，我们需要用到another single step methods。并且如果我们的sub-steps能够准确求解的话，我们的方法其实是一个second order的求解精度。

接下来，我们将这种方法apply在CDE:

Apply to CDE :

$$\begin{array}{ccccc} \frac{\partial u}{\partial t} & = & \epsilon\Delta u & + & f - \mathbf{v}\cdot\mathbf{grad}\, u \\ \updownarrow & & \updownarrow & & \updownarrow \\ \dot{\mathbf{y}} & = & \mathbf{g}(\mathbf{y}) & + & \mathbf{r}(\mathbf{y}) \end{array}$$

▷ Strang split-step method for convection-diffusion evolution eq.:
(w/ Dirichlet b.c. $u(x,t) = g(x,t)$, $(x,t) \in \partial\Omega \times]0,T[$)

- 注意这里只处理的是时间积分，空间积分还是得用finite element mesh来进行处理。不过PDE本来也可以看成是function的integration。

我们分三步走可得:

① (10.3.3.2) $\leftrightarrow$

$$\begin{aligned} &\frac{\partial w}{\partial t} - \epsilon\Delta w = 0 \quad \text{in } \Omega\times]t_{j-1}, t_{j-1}+\tfrac{1}{2}\tau[\,, \\ &w(\boldsymbol{x},t) = g(\boldsymbol{x},t) \quad \forall \boldsymbol{x}\in\partial\Omega,\ t_{j-1} < t < t_{j-1}+\tfrac{1}{2}\tau\,, \\ &w(\boldsymbol{x},t_{j-1}) = u^{(j-1)}(\boldsymbol{x}) \quad \forall \boldsymbol{x}\in\Omega\,. \end{aligned} \tag{10.3.3.6}$$

- 妥妥的 parabolic IBVP

② (10.3.3.3) $\leftrightarrow$

$$\begin{aligned} &\frac{\partial z}{\partial t} + \mathbf{v}(\boldsymbol{x},t)\cdot\mathbf{grad}\, z = f(\boldsymbol{x},t) \quad \text{in } \Omega\times]t_{j-1}, t_j[\,, \\ &z(\boldsymbol{x},t) = g(\boldsymbol{x},t) \quad \text{on inflow boundary } \Gamma_{\text{in}}\,,\ t_{j-1} < t < t_j\,, \\ &z(\boldsymbol{x},t_{j-1}) = w(\boldsymbol{x},t_{j-1}+\tfrac{1}{2}\tau) \quad \forall \boldsymbol{x}\in\Omega\,. \end{aligned} \tag{10.3.3.7}$$

- Pure transport problem

③

(10.3.3.4) $\leftrightarrow$

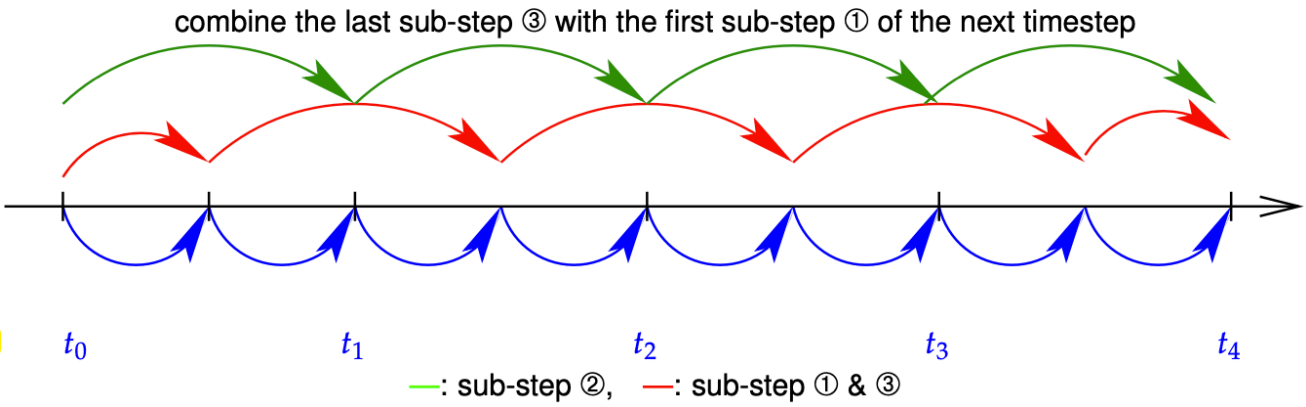
$$\begin{aligned} &\frac{\partial w}{\partial t} - \epsilon \Delta w = 0 \quad \text{in } \Omega \times]t_{j-1} + \tfrac{1}{2}\tau, t_j[\,, \\ &w(\boldsymbol{x}, t) = g(\boldsymbol{x}, t) \quad \forall \boldsymbol{x} \in \partial\Omega\,, \; t_{j-1} + \tfrac{1}{2}\tau < t < t_j\,, \\ &w(\boldsymbol{x}, t_{j-1} + \tfrac{1}{2}\tau) = z(\boldsymbol{x}, t_j) \quad \forall \boldsymbol{x} \in \Omega\,. \end{aligned} \tag{10.3.3.8}$$

Then set $u^{(i)}(\boldsymbol{x}) := w(\boldsymbol{x}, t_j)$, $\boldsymbol{x} \in \Omega$.

- Parabolic IBVP

Leap-frog implementation of Strange splitting

An efficient "implementation" of Strang splitting timestepping can merge sub-steps belonging to different timesteps. If $\mathbf{g} = \mathbf{g}(\mathbf{y})$ we can



446

- 也就是1和3是相同的，可以被用一个single step method over τ 来处理
- 这个地方没搞懂怎么将两个合并成一个的，你的initial condition不同，怎么就能合并成一个求解了呢？老师也没仔细讲，而是粗略地提了一下

7.3.3.2 Particle Method for Pure Transport (Advection)

接下来，在将问题split之后，对于每个方程，我们分别分开来进行求解，对于convection，我们采用method of characteristics.

下面我们简单回顾一下method of characteristics

Recall MOC solution (→ Sect 10.3.2)

$$\frac{\partial u}{\partial t} + \mathbf{v}(\boldsymbol{x},t)\cdot\mathbf{grad}\, u = f \quad \text{in } \widetilde{\Omega} := \Omega\times]0,T[\,,$$
$$u(\boldsymbol{x},t) = g(\boldsymbol{x},t) \quad \text{on} \quad \Gamma_{\text{in}}\times]0,T[\quad,\quad u(\boldsymbol{x},0) = u_0(\boldsymbol{x}) \quad \text{in } \Omega\,, \tag{10.3.3.11}$$

$$\Gamma_{\text{in}} := \{\boldsymbol{x}\in\partial\Omega\colon \mathbf{v}(\boldsymbol{x})\cdot\boldsymbol{n}(\boldsymbol{x}) < 0\}\,. \tag{10.2.1.11}$$

Case $f \equiv 0$:

$$\blacktriangleright\quad u(\boldsymbol{x},t) = \begin{cases} u_0(\boldsymbol{x}_0) & \text{, if } \ \boldsymbol{y}(s)\in\Omega \quad \forall 0 < s < t\,,\\ g(\mathbf{y}(s_0),s_0) & \text{, if } \ \boldsymbol{y}(s_0)\in\partial\Omega, \boldsymbol{y}(s)\in\Omega \quad \forall s_0 < s < t\,,\end{cases} \tag{10.3.3.12}$$

for $(\boldsymbol{x},t)\in\widetilde{\Omega}$, where $s\mapsto\mathbf{y}(s)$ solves the initial value problem for the streamline ODE

$$\frac{d\mathbf{y}}{ds}(s) = \mathbf{v}(\mathbf{y}(s),s) \quad,\quad \mathbf{y}(t) = \boldsymbol{x}\,, \tag{10.3.3.13}$$

fluid particle trajectory

General f:

$$\blacktriangleright\quad u(\boldsymbol{x},t) = \begin{cases} u_0(\boldsymbol{x}_0) + \int_0^t f(\mathbf{y}(s),s)\,\mathrm{d}s & \text{, if } \ \boldsymbol{y}(s)\in\Omega \quad \forall 0 < s < t\,,\\ g(\mathbf{y}(s_0),s_0) + \int_{s_0}^t f(\mathbf{y}(s),s)\,\mathrm{d}s & \text{, if } \ \boldsymbol{y}(s_0)\in\partial\Omega, \boldsymbol{y}(s)\in\Omega \quad \forall s_0 < s < t\,.\end{cases}$$

(10.3.2.7)

- 注意，只能在inflow处施加boundary condition
- 如果源项为0，则我们只需要知道该处的particle最开始是从哪里过来的即可
- 如果源项不为0，则我们需要trajectory做积分。

接下来我们讨论particle method particle method和method of characteristics有啥区别？

particle method = forward tracking of particles

Particle method = forward tracking of particles

I. Pick initial particle locations $p_i \in \Omega$, $i = 1, \dots, N$

II. "Particle pushing": Apply SSM to IVPs

$$\dot{y}(t) = v(y(t), t), \quad y(0) = p_i,$$

timestep $\tau := T/M$

▷ sequence $\left(p_i^{(j)} := y^{(j)}\right)_{j=0}^{M}$, $i = 1, \dots, N$

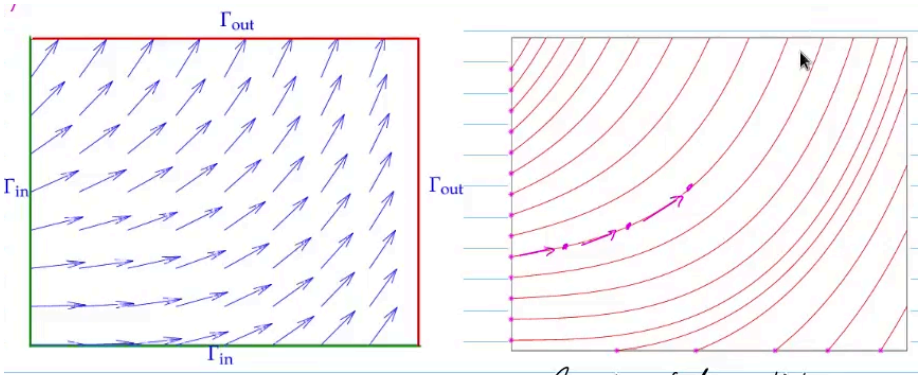
III. Reconstruct approximation $u_h^{(j)} \approx u(\cdot, t_j := j\tau)$ by interpolation:

↙ nix formula

$$u_h^{(j)}(p_i^{(j)}) := u_0(p_i) + \tau \sum_{l=1}^{j-1} f\left(\tfrac{1}{2}(p_i^{(l)} + p_i^{(l-1)}), \tfrac{1}{2}(t_l + t_{l-1})\right), \quad i = 1, \dots, N.$$

↑ for no in/outflow: $v \cdot n = 0$ on $\partial\Omega$

- 核心思想很简单，就是现在初始时刻采点，然后求解那个streamline方程，从而得到这些点随时间演化的一系列点的坐标。然后再用10.3.2.7这个公式（将点的坐标和时刻代入右端项）结合数值积分来进行求解。
- 不太明白为什么考虑的no in/outflow的情况？确实，如果在particle pushing的过程中，会遇到particle离开domain的情况，这时怎么去处理呢？



• ≙ injected particles

- The balance of the number of particles.

Remark 10.3.3.15 (Particle method adapted to inflow/outflow) General velocity fields $\mathbf{v} : \Omega \mapsto \mathbb{R}^d$ that penetrate $\partial\Omega$ can be dealt with by the following modifications of the particle method:

- Stop tracking i-th trajectory as soon as an interpolation nodes $\boldsymbol{p}_i^{(j)}$ lies outside spatial domain Ω.
- In each timestep start new trajectories from fixed locations on inflow boundary Γ_{in} ("particle injection"). These interpolation nodes will carry the boundary value.

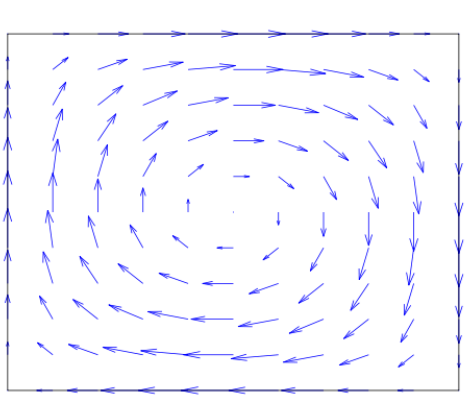
- 不是特别清楚

下面我们用point particle method for pure advection来做数值试验看看

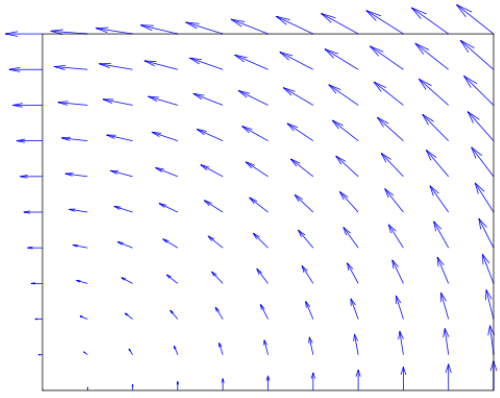
EXPERIMENT 10.3.3.16 (Point particle method for pure advection) In this experiment we take a qualitative look at the ability of particle methods to simulate a rotating flow.

- IBVP (10.3.3.11) on $\Omega =]0,1[^2$, $T = 2$, with $f \equiv 0$, $g \equiv 0$.
- Initial locally supported bump $u_0(\boldsymbol{x}) = \max\{0, 1 - 4\left\|\boldsymbol{x} - \begin{bmatrix} 1/2 \\ 1/4 \end{bmatrix}\right\|\}$.
- Two stationary divergence-free velocity fields
  - $\mathbf{v}_1(\boldsymbol{x}) = \begin{bmatrix} -\sin(\pi x_1)\cos(\pi x_2) \\ \cos(\pi x_1)\sin(\pi x_2) \end{bmatrix}$ satisfying (10.1.1.5),
  - $\mathbf{v}_2(\boldsymbol{x}) = \begin{bmatrix} -x_2 \\ x_1 \end{bmatrix}$.
- Initial positions of interpolation points on regular tensor product grid with meshwidth $h = \frac{1}{40}$.
- Approximation of trajectories by means of explicit trapezoidal rule [Hip22, Eq. (11.4.0.8)] (method of Heun).

- 这里老师用matlab画了左边那张图的解，用了两个图来表示，左边那张图记录particles的位置，右边那张图记录u（也就是温度场）随时间的演化，可以看到，演化的过程中no artificial diffusion: no smearing/damping.



velocity field $\mathbf{v}_1$ (`circvel`)



velocity field $\mathbf{v}_2$ (`rotvel`)

Fig. 448

7.3.3.3 Particle Mesh Method (PMM)

上面我们讨论了advection problem的求解，但是，我们的动机：How to combine Lagrangian advection with a method for the pure diffusion problem (10.3.3.6) faced in the other sub-steps of the Strang splitting timestepping?

Strang split-step method :

In turns solve → Pure transport IBVP : Particle method

→ Parabolic IBVP : MOL

How to combine both ?

我们的思路是：particle method中，particle的temperatures可以看成finite element中的nodal values。**也就是particels来控制我们finite element中网格的mesh划分**太巧妙了。因此，我们的求解过程见下图：

$$\begin{cases} \dfrac{\partial u}{\partial t} - \epsilon \Delta u + \mathbf{v}(x) \cdot \mathbf{grad}\, u = f \quad \text{in } \tilde{\Omega} := \Omega \times]0, T[\,, \\ u(x,t) = 0 \quad \forall x \in \partial\Omega, 0 < t < T \quad , \quad u(x,0) = u_0(x) \quad \forall x \in \Omega\,. \end{cases} \tag{10.3.1.1}$$

→ Single timestep of PPM of size $\tau > 0$: $t_{j-1} \longrightarrow t_j$

given : • mesh $\mathcal{M}^{(j-1)}$

• $u_h^{(j-1)} \in S^0_{1,0}(\mathcal{M}^{(j-1)})$

我们的问题描述是我们有$j-1$时刻的mesh以及其上的nodal values。

I. Diffusion (half) step :

{ $S^0_{1,0}(\mathcal{M}_{j-1})$ - FE + implicit RK-SSM } - MOL for

$$\begin{aligned} \frac{\partial w}{\partial t} - \epsilon \Delta w = 0 \quad &\text{in } \Omega \times]t_{j-1}, t_{j-1} + \tfrac{1}{2}\tau[\,, \\ w(x,t) = 0 \quad &\forall x \in \partial\Omega, t_{j-1} < t < t_{j-1} + \tfrac{1}{2}\tau\,, \\ w(x, t_{j-1}) = u_h^{(j-1)}(x) \quad &\forall x \in \Omega\,. \end{aligned}$$

- 首先我们先进行diffusion step，做一次时间上的演化

II. Advection step :

Particle method for pure transport LBVP

$$\begin{aligned} \frac{\partial z}{\partial t} + \mathbf{v}(x,t) \cdot \mathbf{grad}\, z = f(x,t) \quad &\text{in } \Omega \times]t_{j-1}, t_j[\,, \\ z(x,t) = 0 \quad &\text{on inflow boundary } \Gamma_{\text{in}}\,, t_{j-1} < t < t_j\,, \\ z(x, t_{j-1}) = w_h(x, t_{j-1} + \tfrac{1}{2}\tau) \quad &\forall x \in \Omega\,. \end{aligned}$$

• Initial particle locations $p_i \in \Omega$ = nodes of $\mathcal{M}^{(j-1)}$

• $z(p_i) = w_h(p_i, t_{j-1} + \frac{1}{2}\tau)$

- 接着，我们做particles的演化，我们却nodes作为我们的initial particles locations从而做时间上的演化。

III. Remeshing :

$\mathcal{M}^{(j)}$ = mesh with nodes a final particle locations

IV. Diffusion (half) step on $\mathcal{M}^{(j)}$

- 接着，我们先做remeshing，之后再在remeshing后的particles上做finite element。

接着，我们用做一下数值试验看看：

1D problem

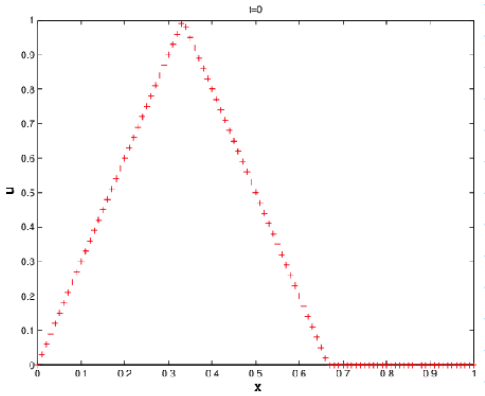
Exp. 10.3.3.20 (PPM for 1D CD-IBVP)

Same IBVP (10.3.1.4) as in Exp. 10.3.1.3:

$$\frac{\partial u}{\partial t} - \epsilon \frac{\partial^2 u}{\partial x^2} + \frac{\partial u}{\partial x} = 0\,,$$
$$u(x,0) = \max(1 - 3|x - \tfrac{1}{3}|, 0)\,,$$
$$u(0) = u(1) = 0\,.$$

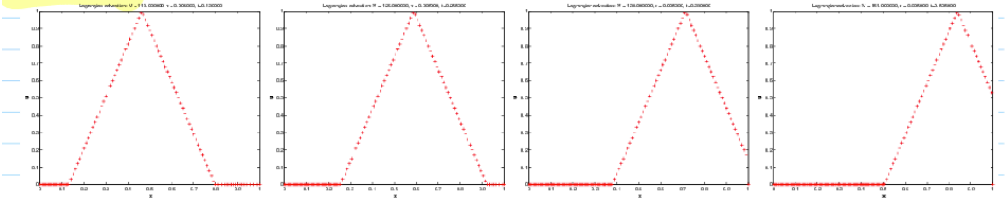
- Linear finite element Galerkin discretization with mass lumping in space
- Strang splitting applied to diffusive and convective terms
- (Sub-optimal) implicit Euler timestepping for diffusive partial timestep

$x \mapsto u_0(x)$ ▷



Convection-dominated:

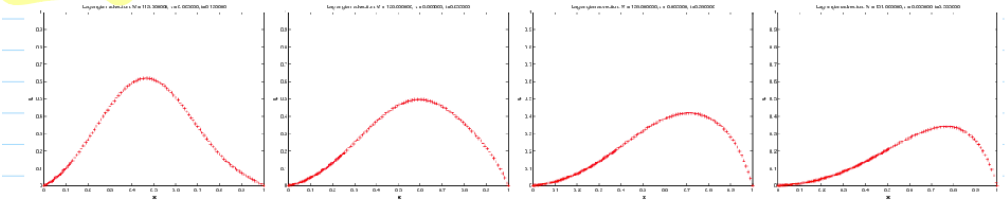
$\epsilon = 10^{-5}$:



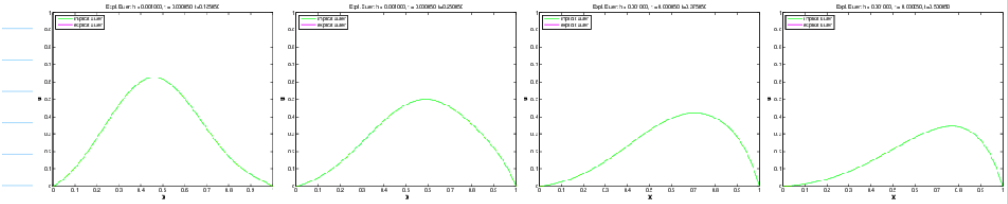
→ No spurious diffusion

Significant diffusion

$\epsilon = 0.1$:



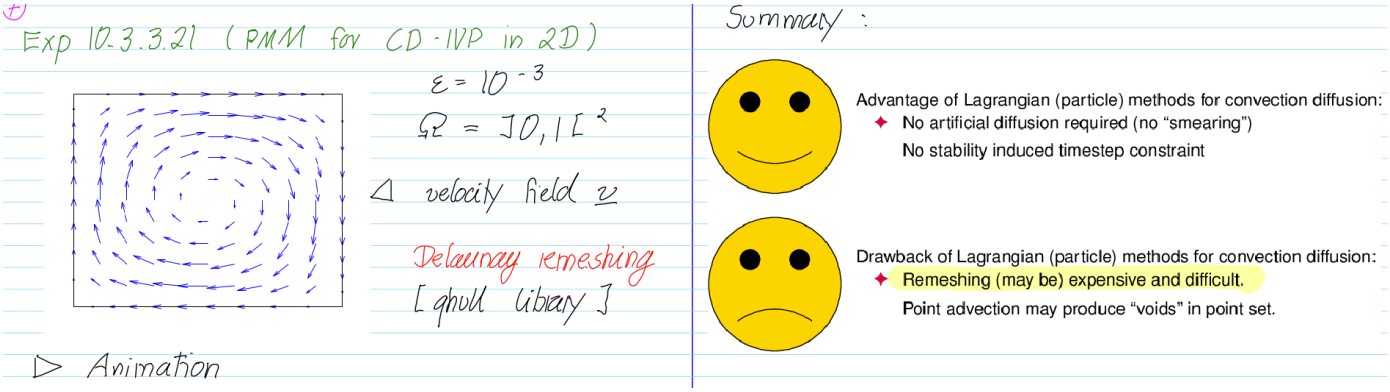
"Reference solution" computed by method of lines, see Exp. 10.3.1.3, with $h = 10^{-3}$, $\tau = 5 \cdot 10^{-5}$ ("overkill resolution"):



→ Good agreement

- 我们发现无论是convection-dominated还是diffusion-dominated的情况，结果都拟合的非常好

2D



- 通过老师课上那个animation，我们发现remeshing后的网格质量很差。

7.3.4 Semi-Lagrangian Method

Now we study a family of methods for transient convection-diffusion that takes into account transport along streamlines, but, in constrast to genuine Lagrangian methods, relies on a fixed mesh.

First, let's recap the CD-IBVP and particle trajectory:

CD-IBVP, Dirichlet b.c.:

$$\frac{\partial u}{\partial t} - \varepsilon \Delta u + \mathbf{v}(\boldsymbol{x},t) \cdot \mathbf{grad}\, u = f \quad \text{in} \quad \tilde{\Omega} := \Omega \times]0,T[\,, \tag{10.3.0.1}$$

- Dirichlet boundary conditions: $u(\boldsymbol{x},t) = g(\boldsymbol{x},t) \quad \forall \boldsymbol{x} \in \partial\Omega\,, \quad 0 < t < T$
- initial conditions: $u(\boldsymbol{x},0) = u_0(\boldsymbol{x}) \quad \forall \boldsymbol{x} \in \Omega.$

Particle trajectory $t \mapsto \boldsymbol{y}(t)$ through $(\boldsymbol{x},t) \in \tilde{\Omega}$:

$$\dot{\boldsymbol{y}}(t) = \boldsymbol{v}(\boldsymbol{y}(t),t)\,, \quad \boldsymbol{y}(t) = \boldsymbol{x}$$

- particle trajectory的关键是写出streamline的ODE，求解这个ODE即可得到从某点出发的stramline。然后我们的CD-IBVP解沿着stramline的物理量的变化。

下面我们引入material derivative的概念，并用这个概念将我们的方程简化，而我们的数值算法也是基于简化后的CD来进行讨论的。

The material derivative the Lagrangian concept of the "rate of change of a function seen by an observer moving with the flow"

Rate of change of a function $f : \Omega \times]0,T[\to \mathbb{R}$ "sensed" by particle moving with the flow :

Material derivative

$$\frac{Df}{D\boldsymbol{v}}(\boldsymbol{x},t) := \frac{d}{d\tau}\{\tau \to f(\boldsymbol{y}(\tau),\tau)\}_{|\tau=t}$$

[Chain rule]

$$= \mathbf{grad}_{\boldsymbol{x}} f(\boldsymbol{y}(t),t) \cdot \dot{\boldsymbol{y}}(t) + \frac{\partial f}{\partial t}(\boldsymbol{y}(t),t)$$

$$= \mathbf{grad}_{\boldsymbol{x}} f(\boldsymbol{y}(t),t)\, \boldsymbol{v}(\boldsymbol{y}(t)) + \frac{\partial f}{\partial t}(\boldsymbol{y}(t),t)$$

$$\frac{Df}{D\mathbf{v}}(\boldsymbol{x},t) = \mathbf{grad}_{\boldsymbol{x}} f(\boldsymbol{x},t) \cdot \mathbf{v}(\boldsymbol{x},t) + \frac{\partial f}{\partial t}(\boldsymbol{x},t)\,. \tag{10.3.4.5}$$

- the value of $f(y(\tau),\tau)$ seen by the moving observer
- 这里的理解就是有一个particle，这个particle是我们的观测变量，我们用其来观测某个场的变化，在这里，我们讨论的是$f(y(\tau),\tau)$这个场随着时间变化的速率，然后这个速率用particle地运动来观测。

至此，我们的问题得到了简化：Convection-diffusion equation with material derivative

$$\frac{\partial u}{\partial t} - \epsilon\Delta u + \mathbf{v}(\boldsymbol{x},t)\cdot\mathbf{grad}\, u = f \quad \text{in} \quad \tilde{\Omega} := \Omega\times]0,T[\,,$$

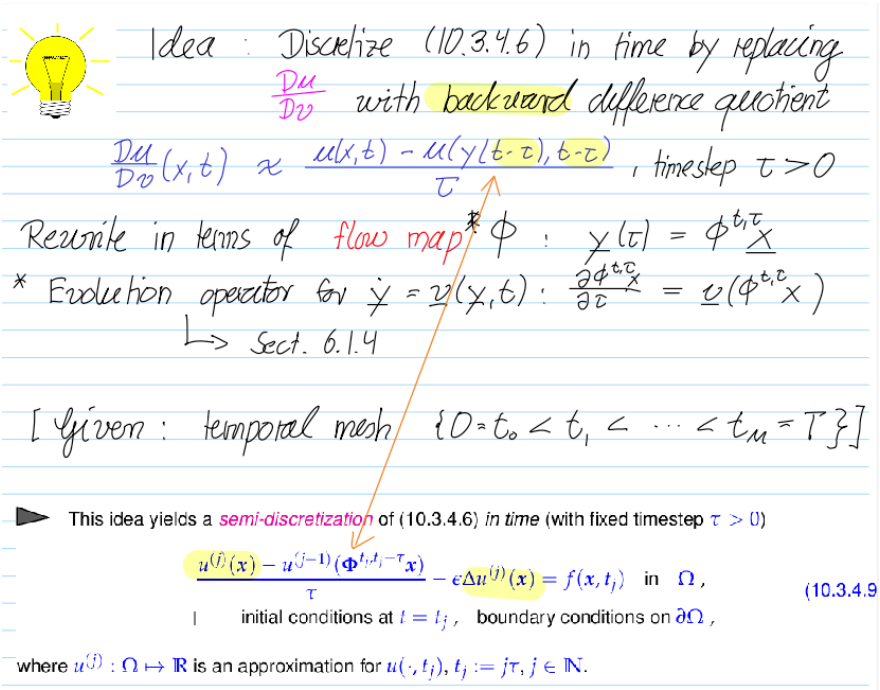
$$\Updownarrow \quad \leftarrow (10.3.4.5)$$

$$\frac{Du}{D\mathbf{v}} - \epsilon\Delta u = f \quad \text{in} \quad \tilde{\Omega} := \Omega\times]0,T[\,. \qquad (10.3.4.6)$$

- 把temporal term和convective term全移到一个term里面去了，这个term就是material derivative.

推导出这个equation是我们的start point for Semi-Lagrangian method。Semi-Lagrangian method不同于之前的method of lines是它semi-discrete先离散的时间，然后才是空间。

对于material derivative的离散，我们采用的是backward difference quotient。然后我们采用flow map替换到material derivative中的$y(t)$,这个flow map其实就是输出的particle的position。evolution operator for $\dot{y} = v(y,t)$. 其满足的对evolution operator求时间导数就相当于对position求时间导数，其输出的是当前点的速度。接下来，我们用离散方法来进行求解：



- 注意此时的u是snapshot，是我们在temporal mesh上每一刻的值，所以和时间无关了。而这个方程中的未知量就是j时刻的流场值。整个问题也可以看成scalar 2nd-order elliptic BVP for u^j.

接下来，我们做空间上的离散（这里假设zero BC），我们就得到了**semi-Lagrangian method**.

§10.3.4.10 (Finite element spatial discretization) Cast (10.3.4.9) into variational form according to the recipe of Section 1.8

$$u^{(j)} \in H^1_0(\Omega): \quad \int_\Omega (u^{(j)}(\boldsymbol{x}) - u^{(j-1)}(\boldsymbol{\Phi}^{t_j,t_j-\tau}\boldsymbol{x}))v(\boldsymbol{x}) + \tau\epsilon\,\mathbf{grad}\,u^{(j)}\cdot\mathbf{grad}\,v\,\mathrm{d}\boldsymbol{x}$$
$$= \tau\int_\Omega f(\boldsymbol{x},t_j)v(\boldsymbol{x})\,\mathrm{d}\boldsymbol{x} \quad \forall v \in H^1_0(\Omega)\,, \quad (10.3.4.11)$$

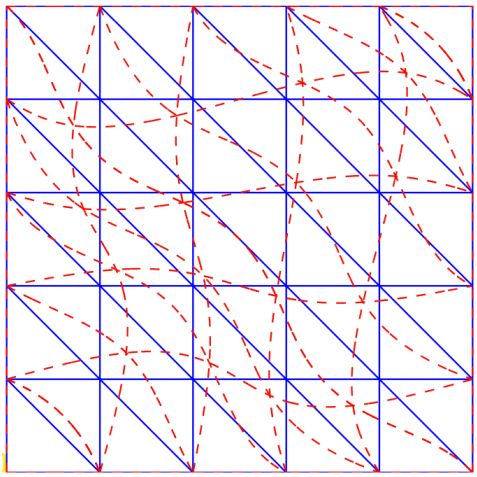
and apply *Galerkin discretization* (here discussed for piecewise linear finite elements, homogeneous Dirichlet boundary conditions $u = 0$ on $\partial\Omega$).

This yields one timestep (size τ) for the semi-Lagrangian method: the approximation $u_h^{(j)}$ for $u(j\tau)$ (equidistant timesteps) is computed from the previous timestep according to

$$u_h^{(j)} \in \mathcal{S}^0_{1,0}(\mathcal{M}): \quad \int_\Omega \frac{u_h^{(j)}(\boldsymbol{x}) - u_h^{(j-1)}(\boldsymbol{\Phi}^{t_j,t_j-\tau}\boldsymbol{x})}{\tau} v_h(\boldsymbol{x})\,\mathrm{d}\boldsymbol{x} + \epsilon\int_\Omega \mathbf{grad}\,u_h^{(j)}\cdot\mathbf{grad}\,v_h\,\mathrm{d}\boldsymbol{x}$$
$$= \int_\Omega f(\boldsymbol{x},t_j)v_h(\boldsymbol{x})\,\mathrm{d}\boldsymbol{x} \quad \forall v_h \in \mathcal{S}^0_{1,0}(\mathcal{M})\,. \quad (10.3.4.12)$$

Here, $\mathcal{M}$ is supposed to be a *fixed* triangular mesh of Ω. ⌟

然后，我们这里就遇到一个问题，我们发现$\boldsymbol{x} \mapsto u_h^{(j-1)}(\boldsymbol{\Phi}^{t_j,t_j-\tau}\boldsymbol{x}) \in H^1(\Omega) \not\in S^0_1(\mathcal{M})$ not $\mathcal{M}$-p.w.-smooth (even if $x \to \Phi^{t_j,t_j-\tau}x$ is smooth)



◁ - - - ≙ image of $\mathcal{M}$ (—) under the mapping $\boldsymbol{\Phi}^{t_j,t_j-\tau}$

The pullback $\boldsymbol{x} \mapsto v_h(\boldsymbol{\Phi}^{t_j,t_j-\tau}\boldsymbol{x})$ of $v_h \in \mathcal{S}^0_{1,0}(\mathcal{M})$ is piecewise smooth w.r.t. the mapped mesh drawn with - - -. Hence, it is not smooth inside the cells of $\mathcal{M}$.

► In general the transported function $\boldsymbol{x} \mapsto v_h(\boldsymbol{\Phi}^{t_j,t_j-\tau}\boldsymbol{x})$ will **not** be a finite element function on $\mathcal{M}$,

- 上面那张图，dash line是blue line (edges)在$t_j - \tau$的mapping
- Therefore, no analytic formula for integral
- $\mathcal{M}$-local quadrature problematic
- 这个地方不懂，之后再反复研究一下

因此，我们又需要进一步做两个近似：

Two more approximations:

1) Replace $x \mapsto u_h^{(j-1)}(\Phi^{t_j,t_j-\tau}x)$ with its $\mathcal{M}$-p.w. linear interpolant

2) Explicit Euler approximation of $\Phi^{t_j,t_j-\tau}$:

$$\Phi^{t_j,t_j-\tau}x = x - v(x,t_j)\tau$$

▷ Practical semi-Lagrangian method:

$u_h^{(j)} \in \mathcal{S}^0_{1,0}(\mathcal{M})$:

$$\int_\Omega \frac{u_h^{(j)}(x) - I_1(u_h^{(j-1)}(\cdot - \tau v(\cdot,t_j)))(x)}{\tau} w_h(x)\,dx + \varepsilon \int_\Omega \mathbf{grad}\, u_h^{(j)} \cdot \mathbf{grad}\, w_h\,dx$$

$$= \int_\Omega f(x,t_j) w_h(x)\,dx \quad \forall w_h \in \mathcal{S}^0_{1,0}(\mathcal{M}).$$

▷ LSE for $\vec{\mu}^{(j)} \longleftrightarrow u_h^{(j)} \propto u(\cdot,t_j)$

$$M(\mu^{(j)} - \vec{\eta}(u_h^{j})) + \tau\varepsilon A \mu^{(j)} = \tau \varphi^{(j)}$$

M: mass matrix; A: FE-Galerkin matrix for $-\Delta$

- 首先就是evolution operator用离散化的方法提到，discrete evolution operator.
- 另一个就是对于那个u_h^{j-1}，采用interpolation进行插值，而插值函数刚好就是我们的tent function，因此我们可以得到全离散的形式：

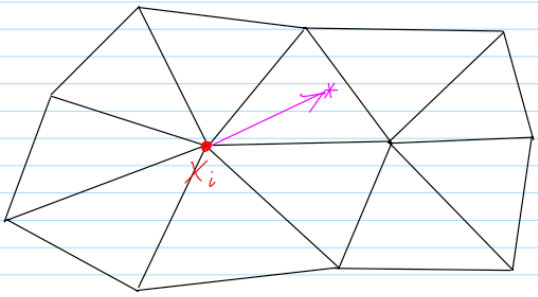
$$\vec{\eta}(u_h^{(j-1)}) \longleftrightarrow I_1(u_h^{(j-1)} - \tau v(\cdot,t_j))$$

$$\eta_i(u_h^{(j-1)}) = u_h^{(j-1)}(x_i - v(x_i,t_j)\tau), \quad i = 1,\dots,N$$

$\{x_i\} \triangleq$ interior mesh nodes

$$\Rightarrow \quad (M + \tau\varepsilon A)\vec{\mu}^{(j)} = M\vec{\eta}(u_h^{(j-1)}) + \tau\vec{\varphi}^{(j)}$$

$$\eta_i(u_h^{(j-1)}) = u_h^{(j-1)}(x_i - v(x_i,t_j)\tau) : *$$



For small τ : $* \in$ cell adjacent to x_i

$\Rightarrow$ efficient local search

接下来我们又做了数值试验，可以看到虽然会出现smearing，但总体上是resonable agreement。而这种方法的好处不用进行remeshing从而可以使得我们的问题可以处理的规模增大，甚至是三维的问题。

8 Numerial Methods for Conservation Laws

Conservation laws describe physical phenomena governed by

- *conservation* laws for certain physical quantities (e.g., mass momentum, energy, etc.),
- *transport* of conserved physical quantities.

We have already examined problems of this type in connection with transient heat conduction in Section 10.1.4. There thermal energy was the conserved quantity and a prescribed external velocity field v determined the transport. Familiarity with Chapter 10 is advantageous, but not essential for understanding this chapter.

A new aspect emerging for general conservation laws is that the transport velocity itself may depend on the conserved quantities themselves, which gives rise to non-linear models.

- 在上一章节，我们讨论了conserved quantity是energy，然后transport是用velocity field来定义的，在这个问题中，velocity field和conserved quantity是不依赖的，下面我们要看一下例子，也就是transport velocity会依赖于conserved quantiteis.

8.1 Conservation Laws: Examples

In this section we will study a few important examples of linear and non-linear conservation laws. In the process we will discover key properties of their solutions. The focus here and throughout this chapter is on **Cauchy problems**, evolution problems posed on the unbounded spatial domain $\Omega = \mathbb{R}^d$.

- Cauchy problems are pure initial value problems (no boundary values). Cauchy problem → no spatial boundary conditions

Why do we restrict ourselves to Cauchy Problems? Mainly for two reasons:

❶ *Finite speed of propagation* typical of conservation laws → Thm. 11.2.7.3

(This means that potential spatial boundaries will not affect the solution for some time in the case of compactly supported initial data, *cf.* situation for wave equation, where we also examined the Cauchy problem, see (9.3.2.2).)

❷ No spatial boundary ➢ need not worry about (spatial) boundary conditions!

(The issues arising for spatial boundary conditions can be very intricate for conservation laws, *cf.* Rem. 11.2.1.8. We would like to avoid these complications.)

8.1.1 Linear Advection

The phenomena modeled by conservation laws are usually **transport-dominated**. The simplest case are models where the transport is due to a *given* velocity field $\mathbf{v} : \Omega \times [0, T] \to \mathbb{R}^d$, *cf.* Section 10.1.1, which leads to *linear* advection/convection evolution problems as discussed in detail in Section 10.3.

8.1.2 Traffic Modeling

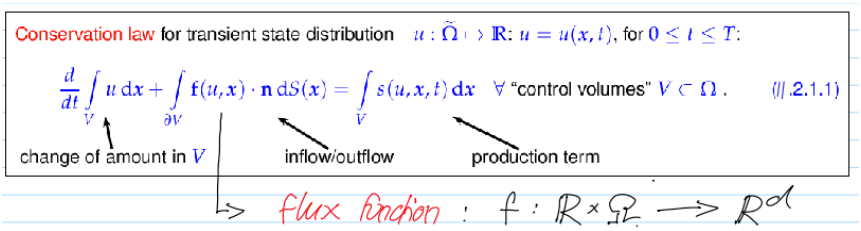
8.2 Scalar Conservation Laws in 1D

This section will give a brief introduction to the mathematics of an important class of mathematical models leading to scalar conservation laws in one spatial dimension. We will first present the general form of the underlying partial differential equation, and then define a suitable notion of "solution", the weak solutions, which can even be discontinuous. Their properties will be discussed in the remaining sub-sections, in order to understand, what features approximate numerical solutions should possess.

8.2.1 Integral and Differential Form

Conservation Laws其实描述的是一组quantity的方程，只不过这个quantity满足一些conservation特性，因此我们可以用conservation law将他们写出来。

我们先来看看integral form of scalar conservation law, general form的形式：



- 对于balance for transient heat equation，$u \equiv$ temperature distribution就是我们这里的quantity，而flux function就是Fourier law：Fourier law : "$f(u,x) = -\kappa(x)\,\mathrm{grad}\,u$" [diffusive flux]
  - 但是这种情况其实并不在我们上述讨论的框架中，因为上述讨论的框架不依赖于quantity(state)的梯度，这里这种称之为diffusive flux（其实，heat diffusion也算conservation laws？只不过不是我们这里研究的范畴，无法放到这里的大类中，用这个大类的方法去求解？）
- Assumptions on flux
  - Depends only on x and the local state $u(x,t)$
  - May be non-linearly depend on u

Next, we can recast (11.2.1.1) in two ways:

► Space-time integral form of (11.2.1.1), *cf.* (11.1.3.2),

$$\int_V u(x,t_1)\,dx - \int_V u(x,t_0)\,dx + \int_{t_0}^{t_1}\int_{\partial V} \mathbf{f}(u,x)\cdot n\,dS(x)dt = \int_{t_0}^{t_1}\int_V s(u,x,t)\,dxdt \qquad (11.2.1.4)$$

for all $V \subset \Omega$, $0 < t_0 < t_1 < T$, $n \hat{=}$ exterior unit normal at ∂V

On the other hand we can apply Gauss' theorem Thm. 1.5.2.4 in space.

► (local) differential form of (11.2.1.1):

$$\frac{\partial}{\partial t}u + \mathrm{div}_x \mathbf{f}(u,x) = s(u,x,t) \quad \text{in } \tilde{\Omega}\,. \qquad (11.2.1.5)$$

div acting on spatial variable x only

下面我们考虑scalar cases：

In the special case $d = 1$ the differential form (11.2.1.5) of a one-dimensional scalar conservation law for the "density" $u : \widetilde{\Omega} \mapsto \mathbb{R}$ is

$$\frac{\partial u}{\partial t}(x,t) + \frac{\partial}{\partial x}(f(u(x,t),x)) = s(u(x,t),x,t) \quad \text{in }]\alpha,\beta[\times]0,T[, \quad \alpha,\beta \in \mathbb{R} \cup \{\pm\infty\}\,. \qquad (11.2.1.6)$$

In any case, these evolution equations have to be supplemented with initial conditions $u(\boldsymbol{x},0) = u_0(\boldsymbol{x})$, $\boldsymbol{x} \in \Omega$

- 并且我们考虑Cauchy Problem $\Omega = \mathbb{R}$: complete statement only requires initial condition $u(x,0) = u_0(x), x \in \mathbb{R}$
- Followings are some examples:

$$\text{linear advection:} \quad \frac{\partial}{\partial t}(\rho u) + \operatorname{div}(\mathbf{v}(\boldsymbol{x},t)(\rho u)) = f(\boldsymbol{x},t) \quad \text{in} \quad \Omega\times]0,T[\,, \quad (\Omega \subset \mathbb{R}^d)\,, \quad (11.1.1.3)$$

$$\text{Burgers equation:} \quad \frac{\partial u}{\partial t} + \frac{\partial}{\partial x}\left(\tfrac{1}{2}u^2\right) = 0 \quad \text{in } \Omega\times]0,T[\,, \quad (\Omega \subset \mathbb{R})\,, \qquad (11.1.3.4)$$

$$\text{traffic flow equation:} \quad \frac{\partial u}{\partial t} + \frac{\partial}{\partial x}(u(1-u)) = 0 \quad \text{in } \Omega\times]0,T[\,, \quad (\Omega \subset \mathbb{R})\,, \qquad (11.1.2.23)$$

are all written in the form (11.2.1.5) and, thus, the corresponding flux functions are easily found:

- For Burgers equation (11.1.3.4): $f(u,x) = f(u) = \frac{1}{2}u^2, \quad s = 0,$
- For traffic flow equation (11.1.2.23): $f(u,x) = f(u) = u(1-u), \quad s = 0,$
- For linear advection (11.1.1.3): $\mathbf{f}(u,\boldsymbol{x}) = \mathbf{v}(\boldsymbol{x},t)u, \quad s = f(\boldsymbol{x},t)$
  (Note: in this case the conserved quantity is actually ρu, which was again denoted by u)

8.2.2 Characteristics

In this section we will come across a surprising ostensible solution formula for non-linear scalar conservation laws in one spatial dimension. Yet, at second glance, we will see that this formula has problem. Its breakdown will teach us that discontinuous solutions are meaningful and very common in the case of conservation laws.

首先，我们研究的对象还是Cauchy problem:

Cauchy problem for 1D scalar conservation law

(CL) (11.2.2.1)
$$\partial_t u + \partial_x f(u) = 0 \quad \text{on } \mathbb{R}\times]0,T[$$
$$u(x,0) = u_0(x) \quad x \in \mathbb{R}$$

with C^1 flux function $f : \mathbb{R} \to \mathbb{R}$

$u \triangleq$ smooth solution of (CL)

Next, we define the characteristic curve for CL:

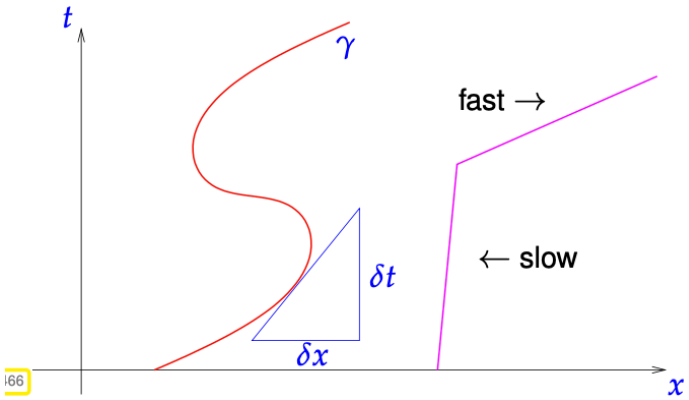
Definition 11.2.2.3. Characteristic curve for one-dimensional scalar conservation law

A curve $\Gamma := (\gamma(\tau), \tau) : [0, T] \mapsto \mathbb{R} \times]0, T[$ in the (x, t)-plane is a characteristic curve for the conservation law (11.2.2.1), if

$$\frac{d}{d\tau}\gamma(\tau) = f'(u(\gamma(\tau), \tau)) , \quad 0 \le \tau \le T , \tag{11.2.2.4}$$

where u is a continuously differentiable solution of (11.2.2.1).

- 与flux function相关，这的flux function与时间t无关。
- 这是一个常微分方程，在我们不给出initial condtion的情况下，其表达的是family of curves



◁ curves in an $x - t$-diagram, described by a function $x = \gamma(t)$
↔ movement of a point on the real axis.
◁ $x - t$-diagram

$\frac{d}{d\tau}\gamma(\tau) =$ speed of interface γ.

接下来，我们看一个例子: Linear advection with constant velocity $f(u) = vu, v \in \mathbb{R}$

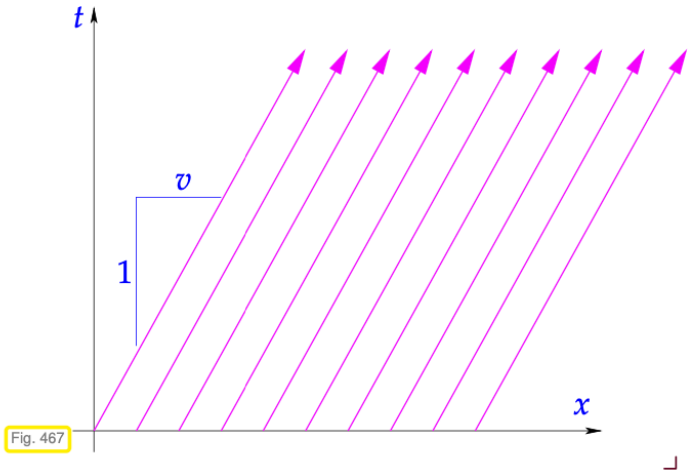
Constant linear advection (11.1.1.11): $f(u) = vu$

➳ characteristics $\gamma(\tau) = v\tau + c, c \in \mathbb{R}$.

solution (11.1.1.12) $u(x, t) = u_0(x - vt)$

meaningful for *any* u_0 ! (*cf.* Section 10.3.2)

▷



- 我们可以看到，对于characteristics表达的curve，其解u保持不变$u_\gamma = \text{const}\ \forall \text{characteristics}\ \gamma$
  - By some derivations, we know that it is a more general fact, because by chain rule:

$$\frac{d}{dt}u(y(t),t) = \frac{\partial u}{\partial x}(y(t),t)\,\dot{y}(t) + \frac{\partial u}{\partial t}(y(t),t)$$
$$= \frac{\partial u}{\partial x}(y(t),t)\,f'(u(y(t),t)) + \frac{\partial u}{\partial t}(y(t),t)$$
$$= \frac{\partial u}{\partial t}(y(t),t) + \frac{\partial}{\partial x}f(u)_{|(x,t)=(y(t),t)}$$
$$= 0$$

provided that u is C^1-smooth.

- Therefore, we have following important lemma:

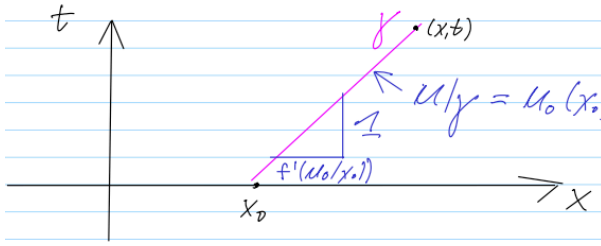
Lemma 11.2.2.6. Classical solutions and characteristic curves

Smooth solutions of (11.2.2.1) *are constant along characteristic curves.*

- 由steramline的概念，我们知道这应该是在source function=0情况下的特列。

- 其实，characteristic cure are a true generalization of the streamline.
  - 在第十章，streamline表达的是particles的trajectory，有明确的物理意义。而streamline对时间的导数就是velocity。

- $u|_\gamma \equiv$ const, if γ characteristic
  [$\dot{\gamma}(\tau) = f'(u(\gamma(t),t))$]
  - ▷ γ has constant slope
  - ▷ Characteristics are straight lines in the x-t plane
    $\gamma = \{(x_0 + tf'(u_0(x_0)), t),\ 0 \le t \le T\}$
  - ▷ Geometric construction

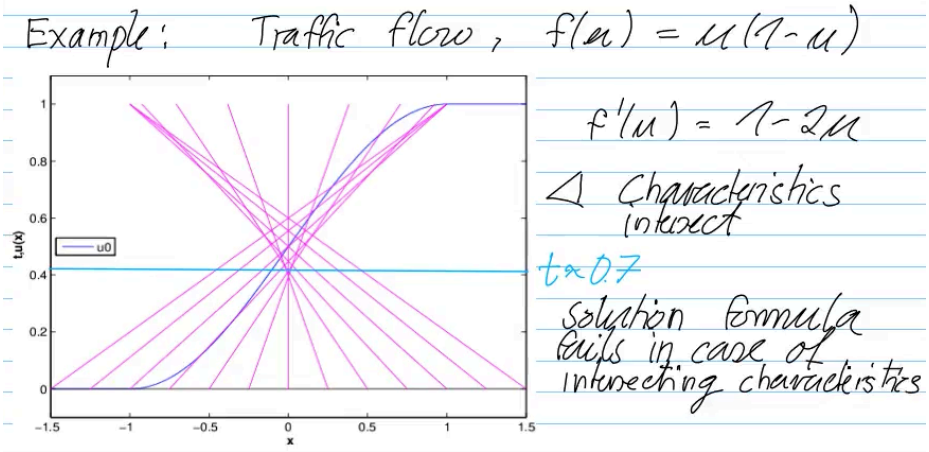


  - ▷ Implicit solution formula

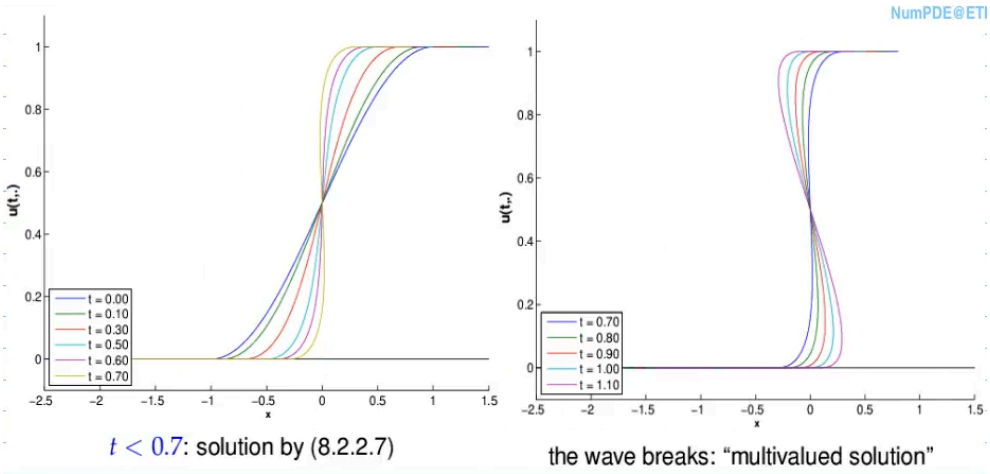
$$u(x,t) = u_0(x - f'(u(x,t))t) \quad (11.2.2.7)$$

  - implicit的意思是右端可能包含$u(x,t)$
  - 注意nnlinear equation need not have a solution
  - 这个implicit solution可能会有问题

下面，我们举几个examples来看看：

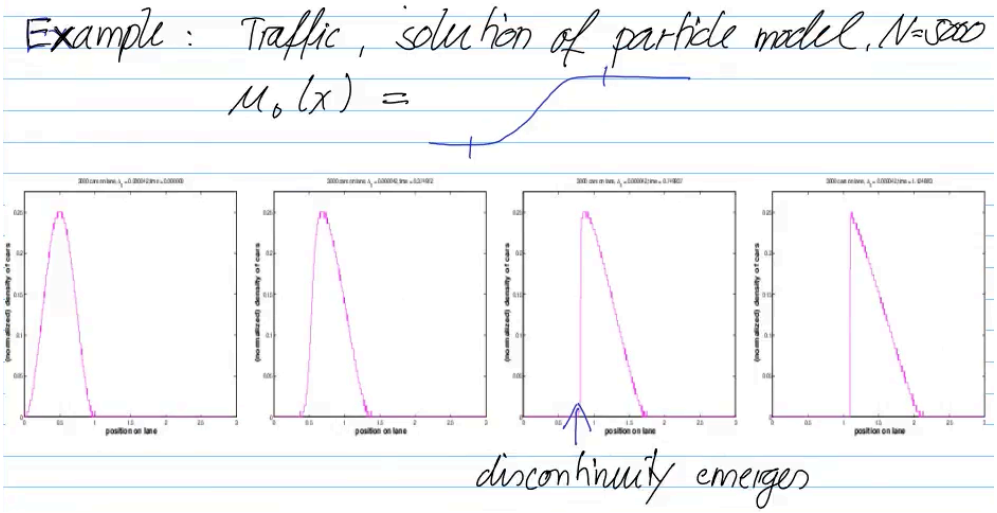


- 可以看到在$t \approx 0.7$的时候，solution formula break down，对于小于0.7的点，还能正确，但对于大于0.7的点，则是完全break down了。



- 我们发现当t>0.7时，一个x可能对应多个解，这就是我们称的the wave breaks.

下面，我们想知道具体发生了什么，我们拿一个traff的例子，模拟看看



- 我们发现，当解不存在的时候，solution里面会产生jump。因此，这个解不满足我们的假设，我们需要寻求能存在的解，这与我们的weak solution很相似。

最后，一些关于meaning of characteristic curves的总结：

- information propagates along characteristics.

Process:

感觉LehrFEM++的核心思路就是把mesh这个object里面的成分拿出来然后分别组成相应的object.

1. Define **mesh** object.
2. Initialize the **dof_handler** object.
3. Define the sparese Galerkin matrix `lf::assemble::COOMatrix<double> mat(N_dofs, N_dofs)`
   (a) Where `size_type N_dofs(dot_handler.NumDofs)`
4. Initialize the Provider objects for local computations
   (a) If we want to consider the numman boundary condition, we should give another parameter `&bd_flags` when we initialize our provider.
5. use the assemble function to build the Galerkin matrix
6. Convert the matrix from triplet format to CRS format
   (a) `const Eigen::SparseMatrix<double> A(mat.makeSparse());`
7.

Assembly

Local → Global mapping: Define a particular object to store the mappping between local informaiton and global information.

- object: `lf::assemble::DofHandler`
- Need to be initiated.

Assembly

Local Computation

- Object: `entity_matrix_provider`: Achieve the local computation for the element matrix.
  - `.isActive(*entity)`
  - `.Eval(*entity)`
- Object: `TEMPMATRIX &matrix`: a matrix type supporting the efficient entry-wise initalization of large sparse Galerkin matrices. Used for storing the Galerkin matrices based on the calculation of the element matrix and the DofHandler.
  - `.AddTOEntry(gdof_idx_t i, gdof_idx_t j, SCALAR increment);`

  - a suitable type meeting this requirement is `lf::assemble::COOMatrix`.
- Object: `entity_vector_provider`
  - `.isActive(*entity)`
  - `.Eval(*entity)`
- Object: `VECTOR &resultvector`

Local Quadrature

- mapping between the local coordinates and the global coordinates is stored in the `lf::geometry::Geometry` object invariably attached to every `lf::mesh::Entity` object. The key function of the interface class `lf::geometry::Geometry` encoding the mapping Φ_K is `lf::geometry::Geometry::Global`.

Local quadrature rules on triangles.

Local quadrature rules on quadrilaterals.