Numerical Schemes

Thematic School Math-Info-HPC

Thierry Dumont

Institut Camille Jordan, Lyon.

May 9, 2016

Sketch of the talk

Introduction

- Recall some definitions

One of the oldest PDE and it's numerical solution

- Reduce to ODEs using finite differences
- Solving ODEs
- Finite elements.
- Finite volumes

Linear Algebra

We solve Partial Differential Equation (and Ordinary Differential Equations, too).

We solve Partial Differential Equation (and Ordinary Differential Equations, too).

Focus on *simple* problems.

- *Simple* problems from the mathematical point of view (theory and numerical analysis is about 50 years old).

We solve Partial Differential Equation (and Ordinary Differential Equations, too).

Focus on *simple* problems.

- *Simple* problems from the mathematical point of view (theory and numerical analysis is about 50 years old).
- But *not so simple* if we want to obtain interesting performances.

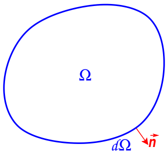
We solve Partial Differential Equation (and Ordinary Differential Equations, too).

Focus on *simple* problems.

- *Simple* problems from the mathematical point of view (theory and numerical analysis is about 50 years old).
- But *not so simple* if we want to obtain interesting performances.
- Most part of numerical methods have been invented at a time where machine architecture did not matter.

– Computers at the begining of my career–

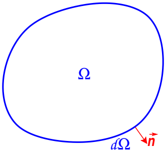
Let us recall some definitions



Ω an open, bounded,... domain in $\mathbb{R}^n$.
$\vec{x} = (x_1, \ldots, x_n) \in \Omega$.
$u(\vec{x}) : \Omega \mapsto \mathbb{R}$.

Let us recall some definitions

Ω an open, bounded,... domain in $\mathbb{R}^n$.
$\vec{x} = (x_1, \ldots, x_n) \in \Omega$.
$u(\vec{x}) : \ \Omega \mapsto \mathbb{R}$.

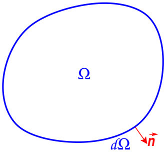


Definition (Gradient)

$$\vec{\mathrm{grad}}\ \ u = (\ldots, \frac{\partial u}{\partial x_i}, \ldots)^t \in \mathbb{R}^n.$$

Let us recall some definitions

Ω an open, bounded,... domain in $\mathbb{R}^n$.
$\vec{x} = (x_1, \ldots, x_n) \in \Omega$.
$u(\vec{x}) : \ \Omega \mapsto \mathbb{R}$.



Definition (Gradient)

$$\vec{\text{grad}} \ \ u = (\ldots, \frac{\partial u}{\partial x_i}, \ldots)^t \in \mathbb{R}^n.$$

$v_i(\vec{x}), i = 1, \ldots, n.$

Definition (Divergence)

$$\text{div v} = \sum_{i=1}^{n} \frac{\partial v_i}{\partial x_i} \ \ \in \mathbb{R}.$$

Definition (Laplacian operator)

$$\Delta u = \sum_{i=1}^{n} \frac{\partial^2 u}{\partial x_i^2} \quad \in \mathbb{R}.$$

Property

Let $u(\vec{x})\ \Omega \mapsto \mathbb{R}$*; then*

$$\Delta u = \operatorname{div}\ \vec{\operatorname{grad}}\, u.$$

Green Formula

Theorem (Green Formula)

Consider $\vec{u}(\vec{x}) = (u_1(\vec{x}), \ldots, u_n(\vec{x}))^t$ *and* $v(\vec{x})$. *Then:*

$$\int_\Omega \operatorname{div} \vec{u}.v \; dx_1 \ldots dx_n + \int_\Omega \vec{u}.\,\vec{\operatorname{grad}}\, v \; dx_1 \ldots dx_n = \int_{\partial\Omega} (\vec{u}.\vec{n}).vds.$$

Green Formula

Theorem (Green Formula)

Consider $\vec{u}(\vec{x}) = (u_1(\vec{x}), \ldots, u_n(\vec{x}))^t$ *and* $v(\vec{x})$. *Then:*

$$\int_\Omega \operatorname{div} \vec{u}.v \; dx_1 \ldots dx_n + \int_\Omega \vec{u}.\,\vec{\operatorname{grad}}\, v \; dx_1 \ldots dx_n = \int_{\partial\Omega} (\vec{u}.\vec{n}).v ds.$$

Property (in dimension $n = 1$)

In dimension 1, *Green formula is nothing but the integration by part formula!*

The Heat equation (Joseph Fourier, 1822)

Let $u(\vec{x}, t)$ be the density at $x \in \Omega$ and at time t of something which *diffuses* in Ω (heat, chemical product,...).

The Heat equation (Joseph Fourier, 1822)

Let $u(\vec{x}, t)$ be the density at $x \in \Omega$ and at time t of something which *diffuses* in Ω (heat, chemical product,...).

To simplify, set $n = 1$ so that $\Omega \subset \mathbb{R}$.

Joseph Fourier
1768–1830.

The Heat equation (Joseph Fourier, 1822)

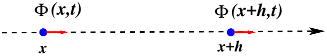
Let $u(\vec{x}, t)$ be the density at $x \in \Omega$ and at time t of something which *diffuses* in Ω (heat, chemical product,...).

Joseph Fourier
$1768–1830.$

To simplify, set $n = 1$ so that $\Omega \subset \mathbb{R}$.

The idea is: on any interval $[x, x + h]$ the amount of u is only modified by a *flux* at the boundary of the interval:

$$\frac{d}{dt} \int_x^{x+h} u(s, t)\, ds = \phi(x, t) - \phi(x + h, t).$$



The Heat equation (Joseph Fourier, 1822)

Let $u(\vec{x}, t)$ be the density at $x \in \Omega$ and at time t of something which *diffuses* in Ω (heat, chemical product,...).

To simplify, set $n = 1$ so that $\Omega \subset \mathbb{R}$.

Joseph Fourier
1768–1830.

The idea is: on any interval $[x, x+h]$ the amount of u is only modified by a *flux* at the boundary of the interval:

$$\frac{d}{dt}\int_x^{x+h} u(s,t)\, ds = \phi(x,t) - \phi(x+h,t).$$

But using the integration by part formula, we get:

$$\frac{d}{dt}\int_x^{x+h} u(s,t) = -\int_x^{x+h} \frac{\partial \phi}{\partial x}(s,t) ds.$$

So that we have a *conservation law*:

$$\frac{\partial u}{\partial t}(x,t) + \frac{\partial \phi}{\partial x}(x,t) = 0.$$

So that we have a *conservation law*:

$$\frac{\partial u}{\partial t}(x,t) + \frac{\partial \phi}{\partial x}(x,t) = 0.$$

We must close this equation. For this, the Fourier law is:

$$\phi(x,t) = -k\frac{\partial u}{\partial x}(x,t).$$

So that we have a *conservation law*:

$$\frac{\partial u}{\partial t}(x,t) + \frac{\partial \phi}{\partial x}(x,t) = 0.$$

We must close this equation. For this, the Fourier law is:

$$\phi(x,t) = -k\frac{\partial u}{\partial x}(x,t).$$

So that we get the:

Heat equation $(n = 1)$

$$\frac{\partial u}{\partial t}(x,t) - k\frac{\partial^2 u}{\partial x^2}(x,t) = 0.$$

So that we have a *conservation law*:

$$\frac{\partial u}{\partial t}(x,t) + \frac{\partial \phi}{\partial x}(x,t) = 0.$$

We must close this equation. For this, the Fourier law is:

$$\phi(x,t) = -k\frac{\partial u}{\partial x}(x,t).$$

So that we get the:

Heat equation $(n = 1)$

$$\frac{\partial u}{\partial t}(x,t) - k\frac{\partial^2 u}{\partial x^2}(x,t) = 0.$$

The Heat equation

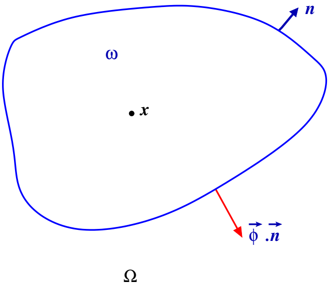
If $n > 1$, consider a domain ω around any point $x \in \Omega$. Then the Fourier law is:

$$\vec{\phi}(\vec{x}) = -k\,\vec{\mathrm{grad}}\,u(\vec{x}).$$

Repeat the same computation as for $d = 1$ using Green formula, to get:

Heat equation

$$\frac{\partial u}{\partial t}(\vec{x}, t) - k\ \Delta u(\vec{x}, t) = 0.$$



Remarks

1. If there is some *surface heating* we get:

$$\frac{\partial u}{\partial t}(\vec{x}, t) - k\ \Delta u(\vec{x}, t) = f(x, t).$$

Remarks

1. If there is some *surface heating* we get:

$$\frac{\partial u}{\partial t}(\vec{x}, t) - k \, \Delta u(\vec{x}, t) = f(x, t).$$

2. The equation must be equipped with initial values $u(x, 0) = u_0(x)$ and boundary conditions, that is:
   2.1 Neuman conditions: $\vec{\mathrm{grad}}\, u.\vec{n} = 0$ on $\partial\Omega$. Then, the integral of u on Ω is constant.

Remarks

1. If there is some *surface heating* we get:

$$\frac{\partial u}{\partial t}(\vec{x},t) - k\ \Delta u(\vec{x},t) = f(x,t).$$

2. The equation must be equipped with initial values $u(x,0) = u_0(x)$ and boundary conditions, that is:
   2.1 Neuman conditions: $\vec{\mathrm{grad}}\, u.\vec{n} = 0$ on $\partial\Omega$. Then, the integral of u on Ω is constant.
   2.2 Dirichlet condition: $u = g$ on $\partial\Omega$.

Remarks

1. If there is some *surface heating* we get:

$$\frac{\partial u}{\partial t}(\vec{x}, t) - k\ \Delta u(\vec{x}, t) = f(x, t).$$

2. The equation must be equipped with initial values $u(x, 0) = u_0(x)$ and boundary conditions, that is:
   2.1 Neuman conditions: $\vec{\text{grad}}\, u.\vec{n} = 0$ on $\partial\Omega$. Then, the integral of u on Ω is constant.
   2.2 Dirichlet condition: $u = g$ on $\partial\Omega$.
   2.3 Robin conditions: $\vec{\text{grad}}\ u.\vec{n} = c.(u - g)$.

Remarks

1. If there is some *surface heating* we get:

$$\frac{\partial u}{\partial t}(\vec{x},t) - k\ \Delta u(\vec{x},t) = f(x,t).$$

2. The equation must be equipped with initial values $u(x,0) = u_0(x)$ and boundary conditions, that is:
   2.1 Neuman conditions: $\vec{\mathrm{grad}}\, u.\vec{n} = 0$ on $\partial\Omega$. Then, the integral of u on Ω is constant.
   2.2 Dirichlet condition: $u = g$ on $\partial\Omega$.
   2.3 Robin conditions: $\vec{\mathrm{grad}}\ u.\vec{n} = c.(u-g)$.
3. Changing the flux ϕ **can change completely the nature of the problem, both mathematically and numerically!** Examples: $\phi(x,t) = u(x,t)$ or something non linear $\phi(x,t) = f(u(x,t))$.

Numerical solution

Replace $u(x,t)$ by a finite dimensional approximation $U(t)$ and Laplace operator Δ by a linear finite dimensional operator (matrix) so that the problems becomes:

$$\frac{dU}{dt} = AU.$$

Numerical solution

Replace $u(x,t)$ by a finite dimensional approximation $U(t)$ and Laplace operator Δ by a linear finite dimensional operator (matrix) so that the problems becomes:

$$\frac{dU}{dt} = AU.$$

which is a (large) system of linear ordinary differential equations.

Remarks

- this new problem is mathematically trivial, if you know the eigen values,vectors of A. This is not the case, except on parallelograms: Fourier introduced his series for this case.

Remarks

- this new problem is mathematically trivial, if you know the eigen values,vectors of A. This is not the case, except on parallelograms: Fourier introduced his series for this case.
- So, we must use a numerical method to solve the system of ODEs and we have to define 2 methods:
  1. How to reduce the heat equation to a system of ODEs?
  2. How to solve this system.

Remarks

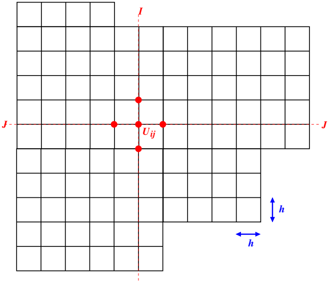
- this new problem is mathematically trivial, if you know the eigen values,vectors of A. This is not the case, except on parallelograms: Fourier introduced his series for this case.
- So, we must use a numerical method to solve the system of ODEs and we have to define 2 methods:
  1. How to reduce the heat equation to a system of ODEs?
  2. How to solve this system.
- We will look also at the stationary problem:

Poisson equation

$$\Delta u(\vec{x}) = f,$$

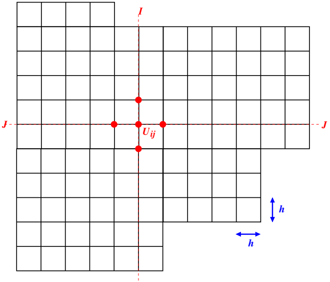
$+$ boundary conditions.

Spatial discretization (reducing to a system of ODEs). 1: finite differences



$$\frac{\partial u}{\partial x}(x_i, y_j) \simeq \frac{u_{i+1,j} - u_{ij}}{h}.$$

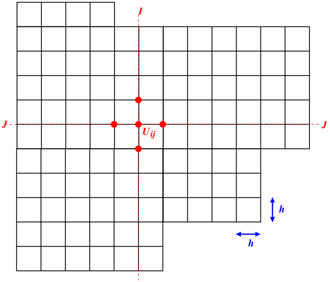
Spatial discretization (reducing to a system of ODEs). 1: finite differences



$$\frac{\partial u}{\partial x}(x_i, y_j) \simeq \frac{u_{i+1,j} - u_{ij}}{h}.$$

$$\frac{\partial^2 u}{\partial x^2}(x_i, y_j) \simeq \frac{\frac{u_{i+1,j} - u_{ij}}{h} - \frac{u_{ij} - u_{i-1,j}}{h}}{h}$$

Spatial discretization (reducing to a system of ODEs). 1: finite differences

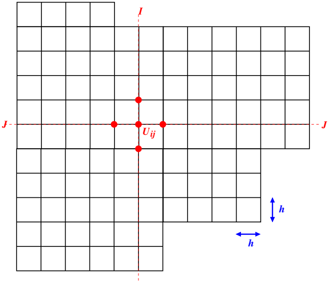


$$\frac{\partial u}{\partial x}(x_i, y_j) \simeq \frac{u_{i+1,j} - u_{ij}}{h}.$$

$$\frac{\partial^2 u}{\partial x^2}(x_i, y_j) \simeq \frac{\frac{u_{i+1,j} - u_{ij}}{h} - \frac{u_{ij} - u_{i-1,j}}{h}}{h}$$

$$\frac{\partial^2 u}{\partial x^2}(x_i, y_j) \simeq \frac{u_{i+1,j} - 2\ u_{ij} + u_{i-1,j}}{h^2}.$$

Spatial discretization (reducing to a system of ODEs). 1: finite differences



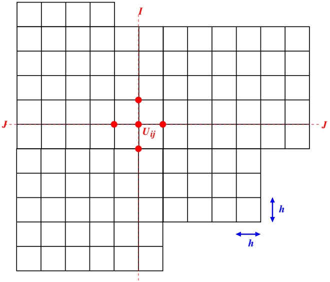
$$\frac{\partial u}{\partial x}(x_i, y_j) \simeq \frac{u_{i+1,j} - u_{ij}}{h}.$$

$$\frac{\partial^2 u}{\partial x^2}(x_i, y_j) \simeq \frac{\frac{u_{i+1,j}-u_{ij}}{h} - \frac{u_{ij}-u_{i-1,j}}{h}}{h}$$

$$\frac{\partial^2 u}{\partial x^2}(x_i, y_j) \simeq \frac{u_{i+1,j} - 2\ u_{ij} + u_{i-1,j}}{h^2}.$$

$$\Delta u(x_i, x_j) \simeq \frac{u_{i+1,j} + u_{i,j+1} - 2\ u_{ij} + u_{i-1,j} + u_{i,j-1}}{h^2}. \qquad (1)$$

Spatial discretization (reducing to a system of ODEs). 1: finite differences



$$\frac{\partial u}{\partial x}(x_i, y_j) \simeq \frac{u_{i+1,j} - u_{ij}}{h}.$$

$$\frac{\partial^2 u}{\partial x^2}(x_i, y_j) \simeq \frac{\frac{u_{i+1,j}-u_{ij}}{h} - \frac{u_{ij}-u_{i-1,j}}{h}}{h}$$

$$\frac{\partial^2 u}{\partial x^2}(x_i, y_j) \simeq \frac{u_{i+1,j} - 2\ u_{ij} + u_{i-1,j}}{h^2}.$$

$$\Delta u(x_i, x_j) \simeq \frac{u_{i+1,j} + u_{i,j+1} - 2\ u_{ij} + u_{i-1,j} + u_{i,j-1}}{h^2}. \qquad (1)$$

Then, choose an order of the points on the grid and store all u_{ij} in a vector U, using this order.Equation (1) defines a matrix A. The Heat equation is approached by:

$$\frac{dU}{dt} = AU,$$

(plus initial condition).

Properties of the finite difference matrix

- A is sparse. Only 3, 5 or 7 (if dimension$=1,2,3$) terms $\neq 0$ (by line).

Properties of the finite difference matrix

- A is sparse. Only 3, 5 or 7 (if dimension$=1, 2, 3$) terms $\neq 0$ (by line).
- We do not need to store A to apply it to a vector ($y = Ax$).

Properties of the finite difference matrix

- A is sparse. Only 3, 5 or 7 (if dimension$=1, 2, 3$) terms $\neq 0$ (by line).
- We do not need to store A to apply it to a vector ($y = Ax$).
- A is symmetric. Moreover, with Dirichlet or Robin boundary conditions, A is positive definite.

Properties of the finite difference matrix

- A is sparse. Only 3, 5 or 7 (if dimension$=1, 2, 3$) terms $\neq 0$ (by line).
- We do not need to store A to apply it to a vector ($y = Ax$).
- A is symmetric. Moreover, with Dirichlet or Robin boundary conditions, A is positive definite.
- Condition number: $\kappa = ||A||.||A^{-1}||$.

Properties of the finite difference matrix

- A is sparse. Only 3, 5 or 7 (if dimension$=1, 2, 3$) terms $\neq 0$ (by line).
- We do not need to store A to apply it to a vector ($y = Ax$).
- A is symmetric. Moreover, with Dirichlet or Robin boundary conditions, A is positive definite.
- Condition number: $\kappa = ||A||.||A^{-1}||$.
  When solving $AX = B$, the relative errors are given by:

$$\frac{\|\delta X\|}{\|X\|} \leq \frac{\kappa(A)}{1 - \kappa(A)\|\delta A\|/\|A\|} \left(\frac{\|\delta A\|}{\|A\|} + \frac{\|\delta b\|}{\|b\|} \right).$$

  Using the Euclidean norm, as A is symmetric, we have

$$\kappa(A) = |\lambda|_{\max}/|\lambda|_{\min}.$$

Properties of the finite difference matrix

- A is sparse. Only 3, 5 or 7 (if dimension$=1,2,3$) terms $\neq 0$ (by line).
- We do not need to store A to apply it to a vector ($y = Ax$).
- A is symmetric. Moreover, with Dirichlet or Robin boundary conditions, A is positive definite.
- Condition number: $\kappa = ||A||.||A^{-1}||$.
  When solving $AX = B$, the relative errors are given by:

$$\frac{\|\delta X\|}{\|X\|} \leq \frac{\kappa(A)}{1 - \kappa(A)\|\delta A\|/\|A\|}\left(\frac{\|\delta A\|}{\|A\|} + \frac{\|\delta b\|}{\|b\|}\right).$$

  Using the Euclidean norm, as A is symmetric, we have

$$\kappa(A) = |\lambda|_{\max}/|\lambda|_{\min}.$$

  If Ω is a $\{$segment, square, cube$\}$, computing the eigenvalues is easy; then one find:

$$\kappa(A) = \mathcal{O}(h^{-2}).$$

- Bad news:
  - say good bye to `float` and use `double`.
  - the system $dU/dt = Au$ is stiff.
- Good news:
  - $\kappa(A) = \mathcal{O}(h^{-2})$ independently of the dimension n (and of the discretization).

Time discretization

How to solve $dU/dt = Au$?

Time discretization

How to solve $dU/dt = Au$?
A is diagonalizable and has eigenvalues $\lambda \in [-1/h^2, -\varepsilon]$.

So the solution is a combination of $\exp(-\lambda_k t)U_k$.

Time discretization

How to solve $dU/dt = Au$?
A is diagonalizable and has eigenvalues $\lambda \in [-1/h^2, -\varepsilon]$.

So the solution is a combination of $\exp(-\lambda_k t) U_k$.

Classical explicit methods cannot be used.

Example: explicit Euler method applied to $du/dt = -\lambda u$ (with $\lambda > 0$).

$$\frac{u_{n+1} - u_n}{\delta t} = -\lambda u_n.$$

How to solve $dU/dt = Au$?
A is diagonalizable and has eigenvalues $\lambda \in [-1/h^2, -\varepsilon]$.

So the solution is a combination of $\exp(-\lambda_k t)U_k$.

Classical explicit methods cannot be used.

Example: explicit Euler method applied to $du/dt = -\lambda u$ (with $\lambda > 0$).

$$\frac{u_{n+1} - u_n}{\delta t} = -\lambda u_n.$$

$=> u_{n+1} = (1 - \delta t\ \lambda)\ u_n$ $=> u_n$ is bounded only if $\delta t < 1/|\lambda|$.

How to solve $dU/dt = Au$?
A is diagonalizable and has eigenvalues $\lambda \in [-1/h^2, -\varepsilon]$.

So the solution is a combination of $\exp(-\lambda_k t)U_k$.

Classical explicit methods cannot be used.

Example: explicit Euler method applied to $du/dt = -\lambda u$ (with $\lambda > 0$).

$$\frac{u_{n+1} - u_n}{\delta t} = -\lambda u_n.$$

$=> u_{n+1} = (1 - \delta t\ \lambda)\ u_n$ $=> u_n$ is bounded only if $\delta t < 1/|\lambda|$.

Do not use explicit methods

For $du/dt = Au$, one must choose $\delta t < 1/|\lambda|_{\max}$.
That is to say, the smallest time scales of the problem must be integrated.

For the Heat equation, this means $\delta t < h^2$! The Heat equation is **stiff**.

Implicit methods

Example: implicit Euler method applied to $du/dt = -\lambda u$ (with $\lambda > 0$).

$$\frac{u_{n+1} - u_n}{\delta t} = -\lambda u_{n+1}.$$

Implicit methods

Example: implicit Euler method applied to $du/dt = -\lambda u$ (with $\lambda > 0$).

$$\frac{u_{n+1} - u_n}{\delta t} = -\lambda u_{n+1}.$$

=> $u_{n+1} = u_n/(1 + \delta t\ \lambda)$ and u_n are bounded.

Definition (A-stability)

A method is said to be A-stable when, applied to $dy/dt = \lambda y$, the sequence $(u_n)_n$ is bounded for any $\lambda \in \mathbb{C}$ such that $\Re(\lambda) < 0$.

Implicit methods

Example: implicit Euler method applied to $du/dt = -\lambda u$ (with $\lambda > 0$).

$$\frac{u_{n+1} - u_n}{\delta t} = -\lambda u_{n+1}.$$

=> $u_{n+1} = u_n/(1 + \delta t \ \lambda)$ and u_n are bounded.

Definition (A-stability)

A method is said to be A-stable when, applied to $dy/dt = \lambda y$, the sequence $(u_n)_n$ is bounded for any $\lambda \in \mathbb{C}$ such that $\Re(\lambda) < 0$.

Properties:

- all A-stable methods are implicit.
- the time step is only bounded by precision considerations, and we do not need to integrate the fastest time scales.

Definition (order of an ODE solver)

Consider $dy/dt = f(y)$ *starting from* y_0 *at time* $t = 0$.
Apply the solver with a time step $\delta t => y_1$ *and compare* y_1 *and the exact solution* $y(\delta t)$.

Definition (order of an ODE solver)

Consider $dy/dt = f(y)$ starting from y_0 at time $t = 0$.
Apply the solver with a time step $\delta t => y_1$ and compare y_1 and the exact solution $y(\delta t)$.
Method is of order p iff the first p coefficients of the Taylor expansions of y_1 and $y(\delta t)$ as functions of δt are equal.

Order, A-stable methods

Definition (order of an ODE solver)

Consider $dy/dt = f(y)$ starting from y_0 at time $t = 0$.
Apply the solver with a time step $\delta t => y_1$ and compare y_1 and the exact solution $y(\delta t)$.
Method is of order p iff the first p coefficients of the Taylor expansions of y_1 and $y(\delta t)$ as functions of δt are equal.

Examples of A-stable methods:

- The Crank-Nicolson method:

$$(u_{n+1} - u_n)/\delta t = (f(u_{n+1} + f(u_n))/2.$$

  Order 2; extremely popular, but has some instabilities (not L-stable, see literature).
- The backward-differentiation formulas (Gear methods).
- Some well designed (diagonally) implicit Runge-Kutta methods.

At each time step, we need solve some linear systems

$$(A + \alpha\delta t\ I)U = B.$$

At each time step, we need solve some linear systems

$$(A + \alpha\delta t\ I)U = B.$$

For the Poisson equation: solve

$$AU = B.$$

We will go back to these linear systems later.

An other spatial discretization.

Finite elements. 1) Weak form.

An other spatial discretization.

Discretize the *weak form* of the equation:

- $\Delta u = f$.
- multiply by v, integrate on Ω:

$$\int_\Omega \operatorname{div} \vec{\operatorname{grad}}\, u(x).v(x)\ dx = \int_\Omega f(x)v(x)dx.$$

Finite elements. 1) Weak form.

An other spatial discretization.

Discretize the *weak form* of the equation:

- $\Delta u = f$.
- multiply by v, integrate on Ω:

$$\int_\Omega \operatorname{div} \vec{\operatorname{grad}}\, u(x).v(x)\ dx = \int_\Omega f(x)v(x)dx.$$

- Use the Green formula, to obtain the

Weak form:

Find u such that for *any* v:

$$\int_\Omega \vec{\operatorname{grad}}\, u(x).\, \vec{\operatorname{grad}}\, v(x)\ dx = \int_\Omega f(x)v(x)dx$$

$+$ some boundary terms which are null with homogeneous bc.

Finite elements. 1) Galerkin method

- Start from the weak form of $\Delta u = f$:

$$\int_\Omega \vec{\text{grad}}\, u(x) . \vec{\text{grad}}\, v(x)\ dx = \int_\Omega f(x) v(x) dx.$$

- Take a finite dimensional space

$$H = \text{span}\{\phi_1(x), \dots, \phi_k(x), \dots, \phi_m(x)\}.$$

- and approach u by $\sum_{i=1}^m U_i \phi_i(x)$ and choose $v \in H$.

Finite elements. 1) Galerkin method

- Start from the weak form of $\Delta u = f$:

$$\int_\Omega \vec{\mathrm{grad}}\, u(x). \vec{\mathrm{grad}}\, v(x)\ dx = \int_\Omega f(x)v(x)dx.$$

- Take a finite dimensional space

$$H = \mathrm{span}\{\phi_1(x), \ldots, \phi_k(x), \ldots, \phi_m(x)\}.$$

- and approach u by $\sum_{i=1}^m U_i\phi_i(x)$ and choose $v \in H$.
- That is to say find $U = (U_1, \ldots, U_k, \ldots, U_m)^t$ such that:

$$\forall i \in 1, m : \int_\Omega (\sum_{i=1}^m U_j\, \vec{\mathrm{grad}}\, \phi_j). \vec{\mathrm{grad}}\, \phi_i dx = \int_\Omega f\phi_i dx.$$

Finite elements. 1) Galerkin method

- Start from the weak form of $\Delta u = f$:

$$\int_\Omega \vec{\operatorname{grad}}\, u(x) . \vec{\operatorname{grad}}\, v(x)\ dx = \int_\Omega f(x) v(x) dx.$$

- Take a finite dimensional space

$$H = \operatorname{span}\{\phi_1(x), \ldots, \phi_k(x), \ldots, \phi_m(x)\}.$$

- and approach u by $\sum_{i=1}^m U_i \phi_i(x)$ and choose $v \in H$.
- That is to say find $U = (U_1, \ldots, U_k, \ldots, U_m)^t$ such that:

$$\forall i \in 1, m : \int_\Omega (\sum_{i=1}^m U_j \vec{\operatorname{grad}}\, \phi_j). \vec{\operatorname{grad}}\, \phi_i dx = \int_\Omega f \phi_i dx.$$

- This is a symmetric linear system $\mathcal{A} U = \mathcal{F}$ with:

$$\mathcal{A}_{i,j} = \int_\Omega \vec{\operatorname{grad}}\, \phi_i . \vec{\operatorname{grad}}\, \phi_j dx \quad \text{and} \quad \mathcal{F}_i = \int_\Omega f \phi_i dx.$$

Idea: use the Galerkin method and choose $H = \mathrm{span}\{\phi_1(x), \dots, \phi_m(x)\}$ such that:

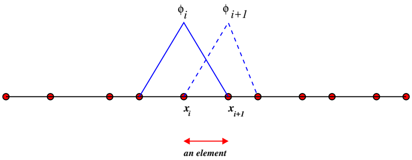
1. The matrix $\mathcal{A}$ is sparse.
2. The coefficients $\mathcal{A}_{i,j}$ and $\mathcal{F}_i$ are easy to compute.
3. The method is adapted to complex geometries.

Finite elements

Idea: use the Galerkin method and choose $H = \mathrm{span}\{\phi_1(x), \ldots, \phi_m(x)\}$ such that:

1. The matrix $\mathcal{A}$ is sparse.
2. The coefficients $\mathcal{A}_{i,j}$ and $\mathcal{F}_i$ are easy to compute.
3. The method is adapted to complex geometries.

The simplest case degree 1 in dimension 1



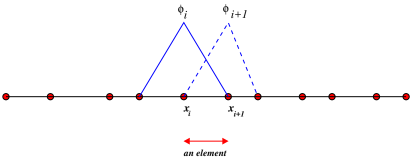
- elements have variable sizes
- functions ϕ verify $\phi_i(x_j) = \delta_{ij}$.
- functions ϕ are polynomial on each element (here degree $= 1$).
- functions ϕ are continuous.

Finite elements

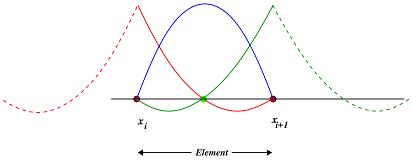
Idea: use the Galerkin method and choose $H = \mathrm{span}\{\phi_1(x), \ldots, \phi_m(x)\}$ such that:

1. The matrix $\mathcal{A}$ is sparse.
2. The coefficients $\mathcal{A}_{i,j}$ and $\mathcal{F}_i$ are easy to compute.
3. The method is adapted to complex geometries.

The simplest case degree 1 in dimension 1



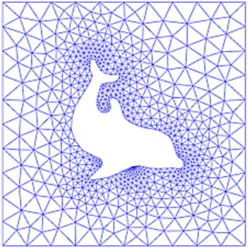
- elements have variable sizes
- functions ϕ verify $\phi_i(x_j) = \delta_{ij}$.
- functions ϕ are polynomial on each element (here degree $= 1$).
- functions ϕ are continuous.



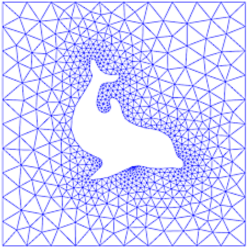
Elements of degree 2 in dimension 1.

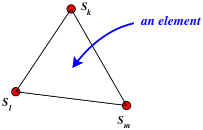
Finite elements. Dimension 2 and more

Dimension 2, and the simplest case: degree 1 in triangles.



Dimension 2, and the simplest case: degree 1 in triangles.

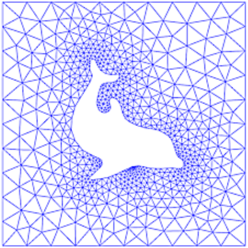


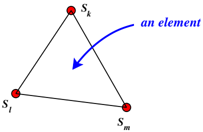


$$\phi_k(s_l) = \delta_{kl}$$

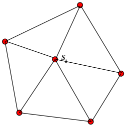
Finite elements. Dimension 2 and more

Dimension 2, and the simplest case: degree 1 in triangles.





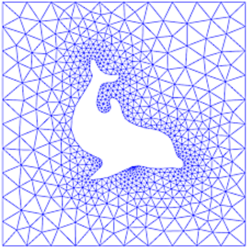
$\phi_k(s_l) = \delta_{kl}$

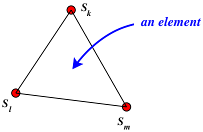


Support of ϕ_k.

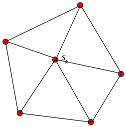
Finite elements. Dimension 2 and more

Dimension 2, and the simplest case: degree 1 in triangles.

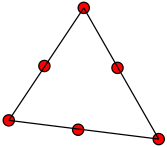




$\phi_k(s_l) = \delta_{kl}$



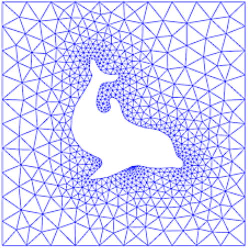
Support of ϕ_k.

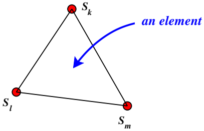


Degree 2 in dimension 2.

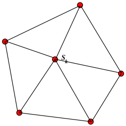
Finite elements. Dimension 2 and more

Dimension 2, and the simplest case: degree 1 in triangles.

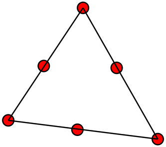




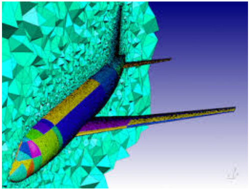
$\phi_k(s_l) = \delta_{kl}$



Support of ϕ_k.



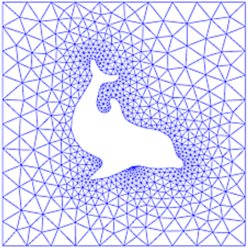
Degree 2 in dimension 2.

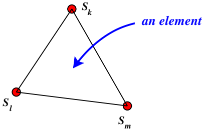


Dimension 3.

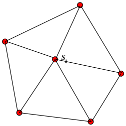
Finite elements. Dimension 2 and more

Dimension 2, and the simplest case: degree 1 in triangles.

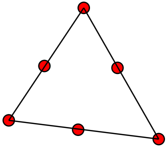




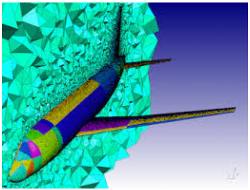
$$\phi_k(s_l) = \delta_{kl}$$



Support of ϕ_k.



Degree 2 in dimension 2.



Dimension 3.

Finite elements: why are they so popular?

- The FEM is well adapted to Navier equations (elasticity, solids).

$$(\lambda+\mu)\,\vec{\mathrm{grad}}\,\mathrm{div}\,\vec{u}(\vec{x})+\mu\Delta\vec{u}(\vec{x})+\vec{f}=0.$$

$=>$ the first large industrial computing codes (Nastran).

Finite elements: why are they so popular?

- The FEM is well adapted to Navier equations (elasticity, solids).

$$(\lambda + \mu)\, \vec{\mathrm{grad}}\, \mathrm{div}\, \vec{u}(\vec{x}) + \mu \Delta \vec{u}(\vec{x}) + \vec{f} = 0.$$

  => the first large industrial computing codes (Nastran).
- The weak form is the correct mathematical framework to study these sorts of PDEs.
- => Very well established mathematical analysis (error bounds, convergence).

Finite elements: why are they so popular?

- The FEM is well adapted to Navier equations (elasticity, solids).

$$(\lambda+\mu)\,\vec{\mathrm{grad}}\,\mathrm{div}\,\vec{u}(\vec{x})+\mu\Delta\vec{u}(\vec{x})+\vec{f}=0.$$

  $=>$ the first large industrial computing codes (Nastran).
- The weak form is the correct mathematical framework to study these sorts of PDEs.
- $=>$ Very well established mathematical analysis (error bounds, convergence).
- Interesting programming problems.
- Huge codes (some 10^6 lines, often in fortran).

Finite elements: why are they so popular?

- The FEM is well adapted to Navier equations (elasticity, solids).

$$(\lambda+\mu)\,\vec{\mathrm{grad}}\,\mathrm{div}\,\vec{u}(\vec{x})+\mu\Delta\vec{u}(\vec{x})+\vec{f}=0.$$

  $=>$ the first large industrial computing codes (Nastran).
- The weak form is the correct mathematical framework to study these sorts of PDEs.
- $=>$ Very well established mathematical analysis (error bounds, convergence).
- Interesting programming problems.
- Huge codes (some 10^6 lines, often in fortran).
- Nice programs available (Freefem).

Finite elements: why are they so popular?

- The FEM is well adapted to Navier equations (elasticity, solids).

$$(\lambda+\mu)\,\vec{\mathrm{grad}}\,\mathrm{div}\,\vec{u}(\vec{x})+\mu\Delta\vec{u}(\vec{x})+\vec{f}=0.$$

  => the first large industrial computing codes (Nastran).
- The weak form is the correct mathematical framework to study these sorts of PDEs.
- => Very well established mathematical analysis (error bounds, convergence).
- Interesting programming problems.
- Huge codes (some 10^6 lines, often in fortran).
- Nice programs available (Freefem).

The mathematical analysis of FE is used for the analysis of many other numerical methods.

Cherchez la FEM (G. Strang and G.J. Fix,in the first book analyzing the FEM (1973)).

Finite volumes: back to the origin.

Recall that the Heat equation can be written:

$$\frac{du}{dt}(\vec{x},t) + \operatorname{div}\vec{\phi}(\vec{x},t) = 0$$

with (omitting coefficient k):

$$\vec{\phi}(\vec{x},t) = -\overrightarrow{\operatorname{grad}}\,u(\vec{x},t).$$

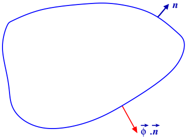
Finite volumes: back to the origin.

Recall that the Heat equation can be written:

$$\frac{du}{dt}(\vec{x},t) + \operatorname{div}\vec{\phi}(\vec{x},t) = 0$$

with (omitting coefficient k):

$$\vec{\phi}(\vec{x},t) = -\,\vec{\operatorname{grad}}\,u(\vec{x},t).$$



On any *volume* $\omega \subset \Omega$ we have:

$$\frac{d\int_\omega u d\vec{x}}{dt} = \int_{\partial\omega} \vec{\phi}(s,t).\vec{n}\ ds.$$

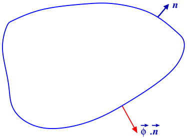
Finite volumes: back to the origin.

Recall that the Heat equation can be written:

$$\frac{du}{dt}(\vec{x},t) + \operatorname{div}\vec{\phi}(\vec{x},t) = 0$$

with (omitting coefficient k):

$$\vec{\phi}(\vec{x},t) = -\,\vec{\operatorname{grad}}\,u(\vec{x},t).$$

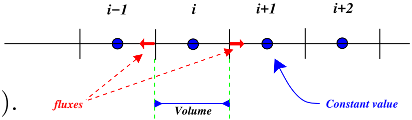


On any *volume* $\omega \subset \Omega$ we have:

$$\frac{d\int_\omega u d\vec{x}}{dt} = \int_{\partial\omega} \vec{\phi}(s,t).\vec{n}\ ds.$$

In dimension 1, this is:

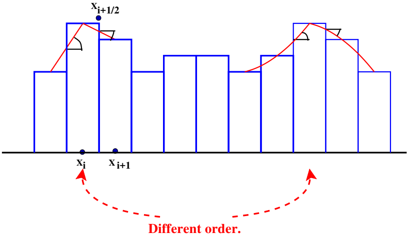
$$\frac{d}{dt}\int_x^{x+h} u(s,t)\,ds = \phi(x,t) - \phi(x+h,t).$$



Finite volumes

Define the fluxes by interpolation.
Most simple case:

$$\phi(x_{i+1/2}) = \frac{v_{i+1} - v_i}{h}.$$

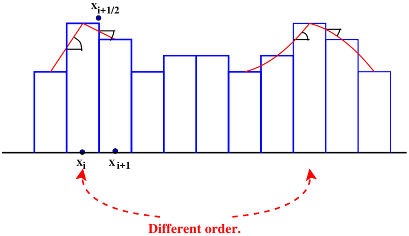


Can be generalized in dimension 2 and 3 to more sophisticated volumes (triangles...).

Finite volumes

Define the fluxes by interpolation.
Most simple case:

$$\phi(x_{i+1/2}) = \frac{v_{i+1} - v_i}{h}.$$



Can be generalized in dimension 2 and 3 to more sophisticated volumes (triangles...).

The idea is interesting: *all* the art is in the definition of the fluxes; for example, for first order problems:
$\partial u/\partial t = \partial u/\partial x$ or $\partial u/\partial t = \partial f(u)/\partial x$
this is a difficult task.

Linear systems, with sparse matrices; iterative methods

Krylov methods:

$$K_n = \{B, AB, A^2B, \ldots, A^nB\}.$$

All methods involve:

- matrix $\times$ vector products.
- linear combinations.
- dot products.

Most popular: Conjugate Gradient (symmetric systems), GMRES, BICGSTAB, MINRES..
With Conjugate Gradient, no need to store K_n.

Linear systems, with sparse matrices and iterative methods: preconditioning

Idea: the convergence of iterative methods depends of the condition number of the matrix.
For the conjugate gradient:

$$||u_k - u_*|| = 2(\frac{\sqrt{\kappa(A)} - 1}{\sqrt{\kappa(A)} + 1})^k ||u_0 - u_*||.$$

Idea: the convergence of iterative methods depends of the condition number of the matrix.
For the conjugate gradient:

$$||u_k - u_*|| = 2\left(\frac{\sqrt{\kappa(A)} - 1}{\sqrt{\kappa(A)} + 1}\right)^k ||u_0 - u_*||.$$

Preconditioning

Find a matrix P such that $\kappa(PA) << \kappa(A)$.

A lot of methods have been studied!

Linear systems, with sparse matrices and iterative methods: preconditioning

Most common idea: incomplete factorization.

Linear systems, with sparse matrices and iterative methods: preconditioning

Most common idea: incomplete factorization.

- Observe that the best preconditioner would be $U^{-1}.L^{-1}$ where $L.U = A$ is the LU factorization of A. This is not a good idea!

Linear systems, with sparse matrices and iterative methods: preconditioning

Most common idea: incomplete factorization.

- Observe that the best preconditioner would be $U^{-1}.L^{-1}$ where $L.U = A$ is the LU factorization of A. This is not a good idea!
- Observe that, in the LU factorization of A, many non zero terms are created.

Linear systems, with sparse matrices and iterative methods: preconditioning

Most common idea: incomplete factorization.

- Observe that the best preconditioner would be $U^{-1}.L^{-1}$ where $L.U = A$ is the LU factorization of A. This is not a good idea!
- Observe that, in the LU factorization of A, many non zero terms are created.
- The idea is to use the LU factorization algorithm, but to compute only some well chosen terms (at least those corresponding to (i, j) where $A_{ij} \neq 0$). This needs some art (and science).
- if A is symmetric, you can replace LU by Cholesky.

Linear systems, with sparse matrices and iterative methods: preconditioning

Most common idea: incomplete factorization.

- Observe that the best preconditioner would be $U^{-1}.L^{-1}$ where $L.U = A$ is the LU factorization of A. This is not a good idea!
- Observe that, in the LU factorization of A, many non zero terms are created.
- The idea is to use the LU factorization algorithm, but to compute only some well chosen terms (at least those corresponding to (i, j) where $A_{ij} \neq 0$). This needs some art (and science).
- if A is symmetric, you can replace LU by Cholesky.

- Good: relatively efficient methods. Existing libraries.
- Not so good:
  - never a universal method.
  - not very parallel!
  - low arithmetic intensity.

Linear systems, with sparse matrices and iterative methods: preconditioning

An other idea: Chebyshev preconditioning. This is an old idea!

An other idea: Chebyshev preconditioning. This is an old idea!

Consider the sequence:

$$\frac{u_k - u_{k-1}}{\tau_k} = Au_k - F.$$

where the τ_k are chosen so that the sequence converges.

Linear systems, with sparse matrices and iterative methods: preconditioning

An other idea: Chebyshev preconditioning. This is an old idea!

Consider the sequence:

$$\frac{u_k - u_{k-1}}{\tau_k} = Au_k - F.$$

where the τ_k are chosen so that the sequence converges.

If you know an interval which contains $[\lambda_{\min}, \lambda_{\max}]$ (the eigenvalues of A), you can build an optimal sequence of $\{\tau_k\}_k$. The $\{\tau_k\}_k$ are function of the roots of the kth Chebyshev polynomial.

An other idea: Chebyshev preconditioning. This is an old idea!

Consider the sequence:

$$\frac{u_k - u_{k-1}}{\tau_k} = Au_k - F.$$

where the τ_k are chosen so that the sequence converges.

If you know an interval which contains $[\lambda_{\min}, \lambda_{\max}]$ (the eigenvalues of A), you can build an optimal sequence of $\{\tau_k\}_k$. The $\{\tau_k\}_k$ are function of the roots of the kth Chebyshev polynomial.

Compute the k first steps: u_k is an approximation of the inverse of A. You can use it as preconditionner.

An other idea: Chebyshev preconditioning. This is an old idea!

Consider the sequence:

$$\frac{u_k - u_{k-1}}{\tau_k} = Au_k - F.$$

where the τ_k are chosen so that the sequence converges.

If you know an interval which contains $[\lambda_{\min}, \lambda_{\max}]$ (the eigenvalues of A), you can build an optimal sequence of $\{\tau_k\}_k$. The $\{\tau_k\}_k$ are function of the roots of the kth Chebyshev polynomial.

Compute the k first steps: u_k is an approximation of the inverse of A. You can use it as preconditionner.

- Good: simple and parallel!
- Not so good:
  - difficult to adapt to non symmetric problems.
  - not as fast as incomplete preconditioning in terms of speed of convergence.

Linear systems, sparse matrices and iterative methods: preconditioning: let us be a bit more concrete

Non Cartesian meshes (finite elements in non Cartesian domains)

The infamous CSR/CSL format:

	0	1	2	3	4
0	2.0		3.5		6.7
1		8.2		9.2	
2		1.1	2.8		
3	3.0		1.5	4.5	
4		2.5		8.9	

	0	1	2	3	4	5			
rowptr	0	3	5	7	10	12			

	0	1	2	3	4	5	6	7	8	9	10	11
colind	0	2	4	1	3	1	2	0	2	3	1	3

	0	1	2	3	4	5	6	7	8	9	10	11
values	2.0	3.5	6.7	8.2	9.2	1.1	2.8	3.0	1.5	4.5	2.5	8.9

Linear systems, sparse matrices and iterative methods: preconditioning: let us be a bit more concrete

Non Cartesian meshes (finite elements in non Cartesian domains)

The infamous CSR/CSL format:

	0	1	2	3	4
0	2.0		3.5		6.7
1		8.2		9.2	
2		1.1	2.8		
3	3.0		1.5	4.5	
4		2.5		8.9	

	0	1	2	3	4	5			
rowptr	0	3	5	7	10	12			

	0	1	2	3	4	5	6	7	8	9	10	11
colind	0	2	4	1	3	1	2	0	2	3	1	3

	0	1	2	3	4	5	6	7	8	9	10	11
values	2.0	3.5	6.7	8.2	9.2	1.1	2.8	3.0	1.5	4.5	2.5	8.9

All operations are memory bounded.

An experience on my Sandy-Bridge machine (16 core):

Take the 7 points stencil of the Laplace operator in dimension 3 and store the matrix in CSR format. How fast is a matrix $\times$ vector product?

(double: 64 bits, int: 32, OpenMP).

An experience on my Sandy-Bridge machine (16 core):

Take the 7 points stencil of the Laplace operator in dimension 3 and store the matrix in CSR format. How fast is a matrix $\times$ vector product?

(double: 64 bits, int: 32, OpenMP).

- Algorithm Bdwth: $37/2$ double; Flops: $13 => I_a \simeq 0.7$ flops/double.

An experience on my Sandy-Bridge machine (16 core):

Take the 7 points stencil of the Laplace operator in dimension 3 and store the matrix in CSR format. How fast is a matrix $\times$ vector product?

(double: 64 bits, int: 32, OpenMP).

- Algorithm Bdwth: $37/2$ double; Flops: $13 => I_a \simeq 0.7$ flops/double.
- Machine Bdwth: 8.73 Giga doubles/s (measured with `Stream`).

$=>$ Attainable $= 0.7 \times 8.73 = 6.11$ Gflops.

An experience on my Sandy-Bridge machine (16 core):

Take the 7 points stencil of the Laplace operator in dimension 3 and store the matrix in CSR format. How fast is a matrix $\times$ vector product?

(double: 64 bits, int: 32, OpenMP).

- Algorithm Bdwth: $37/2$ double; Flops: $13 => I_a \simeq 0.7$ flops/double.
- Machine Bdwth: 8.73 Giga doubles/s (measured with `Stream`).

$=>$ Attainable $= 0.7 \times 8.73 = 6.11$ Gflops.

Measured: 6.42 Gflops.

An experience on my Sandy-Bridge machine (16 core):

Take the 7 points stencil of the Laplace operator in dimension 3 and store the matrix in CSR format. How fast is a matrix $\times$ vector product?

(double: 64 bits, int: 32, OpenMP).

- Algorithm Bdwth: $37/2$ double; Flops: $13 =>$ $I_a \simeq 0.7$ flops/double.
- Machine Bdwth: 8.73 Giga doubles/s (measured with `Stream`).

=> Attainable $= 0.7 \times 8.73 = 6.11$ Gflops.

Measured: 6.42 Gflops.

But CSR is the sort of data structure you need to use with non Cartesian meshes and incomplete factorization preconditioning

Cartesian meshes (finite differences, finite volumes or finite elements on Cartesian meshes.)

Actually, A is made of blocks, all equals ($=>$ stencils).

Cartesian meshes (finite differences, finite volumes or finite elements on Cartesian meshes.)

Actually, A is made of blocks, all equals ($=>$ stencils).

No need to store the matrix (so, no CSR format) if we perform only matrix $\times$ vector products.

Cartesian meshes (finite differences, finite volumes or finite elements on Cartesian meshes.)

Actually, A is made of blocks, all equals ($=>$ stencils).

No need to store the matrix (so, no CSR format) if we perform only matrix $\times$ vector products.

- We cannot use incomplete factorization preconditioning if we refuse to use the CSR format.
- But we can use Chebyshev preconditioning.

Cartesian meshes (finite differences, finite volumes or finite elements on Cartesian meshes.)

Actually, A is made of blocks, all equals ($=>$ stencils).

No need to store the matrix (so, no CSR format) if we perform only matrix $\times$ vector products.

- We cannot use incomplete factorization preconditioning if we refuse to use the CSR format.
- But we can use Chebyshev preconditioning.

See the results of Wim Vanroose using Pluto $+$ Chebyshev preconditioning on finite differences discretization.

Trends

- High order methods (high order is order > 2 :-)).

Trends

- High order methods (high order is order > 2 :-)).
- Accept to loose some optimality to be more respectful of the "Physics" (Conserve energy if the system is conservative, positivity is the solution is positive, waves...).

Trends

- High order methods (high order is order > 2 :-)).
- Accept to loose some optimality to be more respectful of the "Physics" (Conserve energy if the system is conservative, positivity is the solution is positive, waves...).
- Take account of multiscale character of the problems.

Trends

- High order methods (high order is order > 2 :-)).
- Accept to loose some optimality to be more respectful of the "Physics" (Conserve energy if the system is conservative, positivity is the solution is positive, waves...).
- Take account of multiscale character of the problems.
- Compute in dimension ≥ 3 (for Boltzman equations and related problems, it would be good to compute in dimension 6!).

Trends

- High order methods (high order is order > 2 :-)).
- Accept to loose some optimality to be more respectful of the "Physics" (Conserve energy if the system is conservative, positivity is the solution is positive, waves...).
- Take account of multiscale character of the problems.
- Compute in dimension ≥ 3 (for Boltzman equations and related problems, it would be good to compute in dimension 6!).
- Data assimilation.
- ...