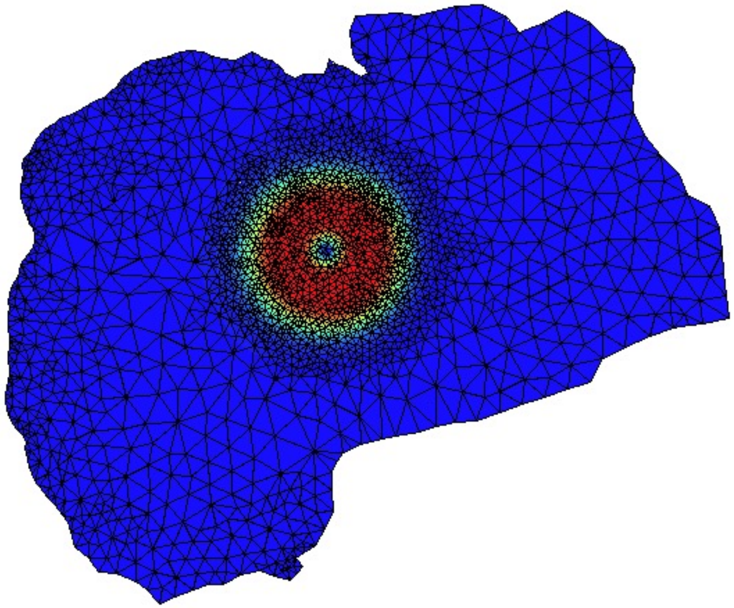
The Finite Element Method



is a way of computing approximate solutions to a mathematical model describing a physical process.

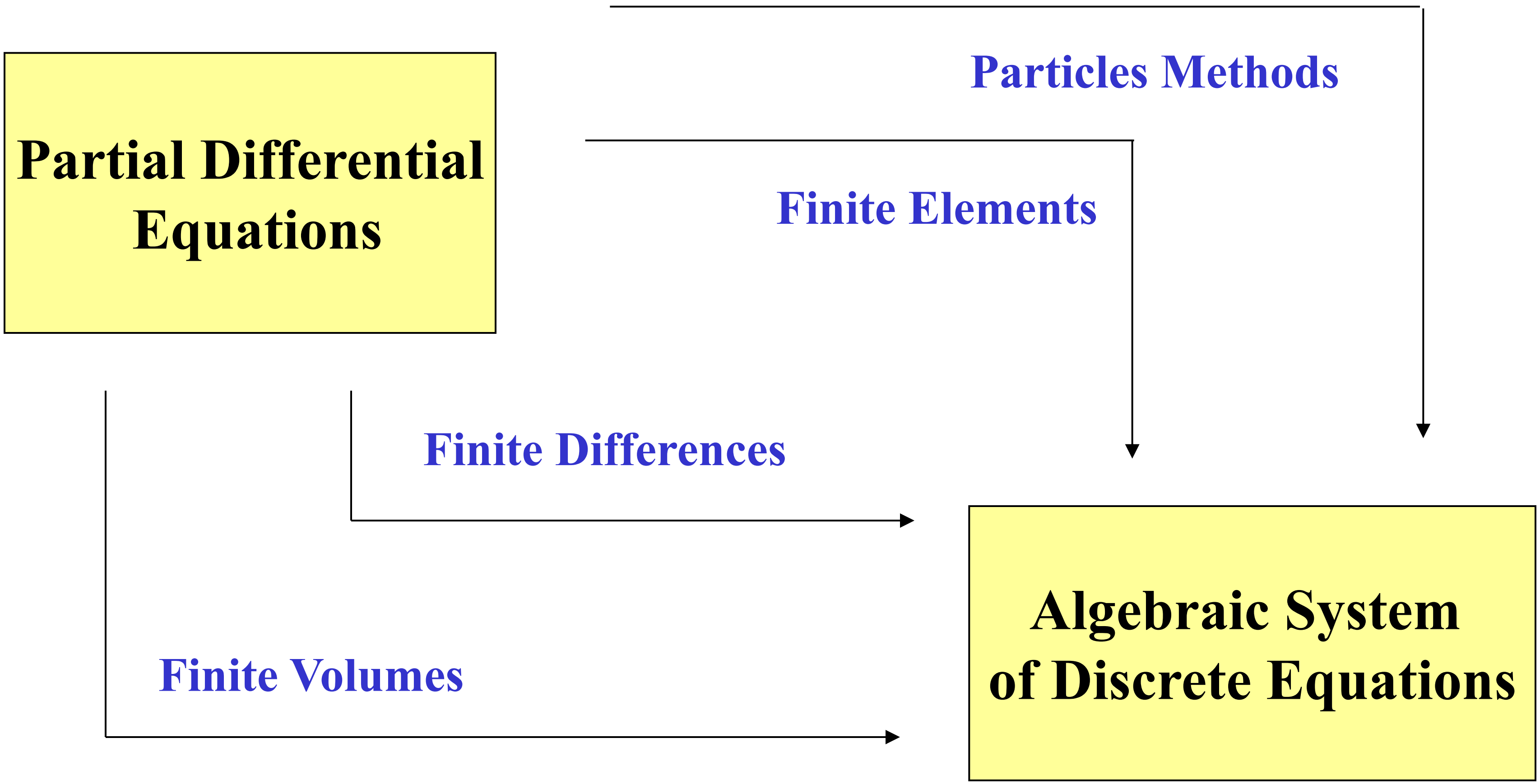
What is a mathematical model ?
A boundary value problem.

Typical applications

- Deformable solids mechanics
- Fluid dynamics (CFD)
- Electromagnetism
- Transport phenomena
- Climatology

What is a boundary value problem ?
A set of partial differential equations with boundary and initial conditions.

Finite Elements, Finite Differences, Finite Volumes etc.

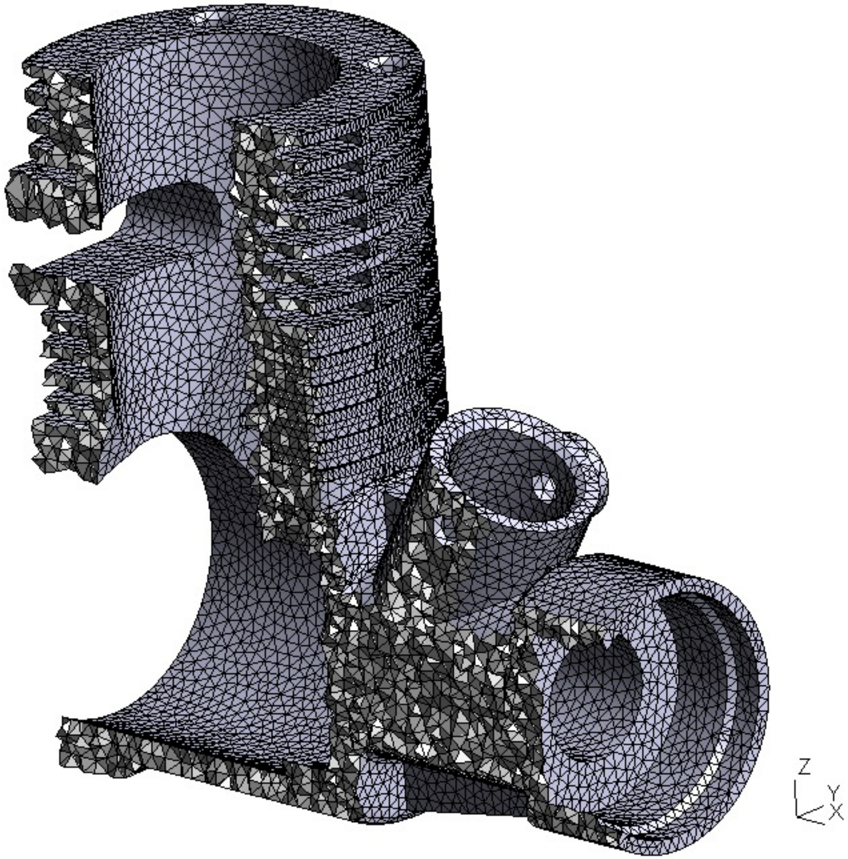


The Finite Elements Method is a discretization method

The problem geometry is divided in small finite elements.

On each element, the solution is approximated by means of unknown nodal values and given polynomials

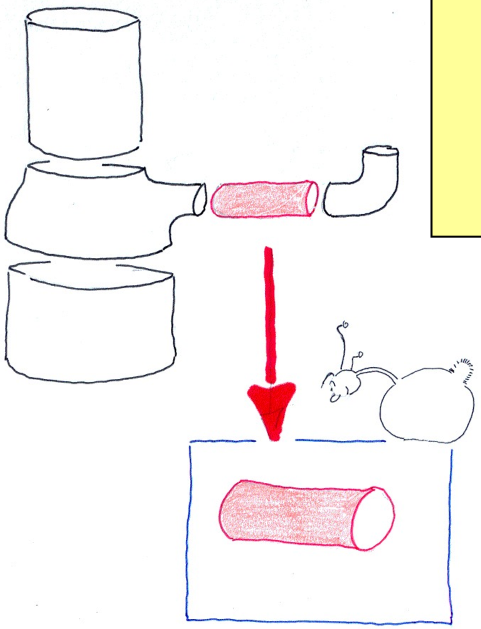
$$u(\mathbf{x}) \approx u^h(\mathbf{x}) = \sum_{j=1}^{n} U_j \tau_j(\mathbf{x})$$



Classical Engineering Analysis

Analysis through simple geometries and a limited combination of approximate models :

- **Lubrication theory**
- **Bars**
- **Beams**
- **Plates and shells**



Exact solution to approximate problems

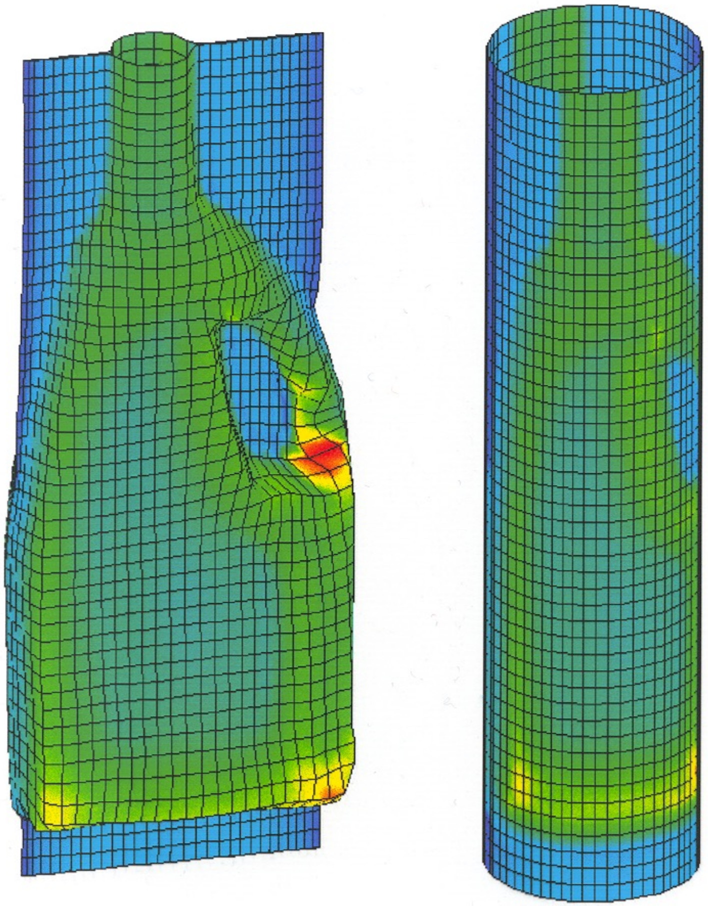
Low computer's cost
Good physical understanding

Simplicity of models
Complex geometries and loads cannot be handled
Complex materials cannot be analyzed

Computer Aided Engineering Analysis

The Finite Element Method provides approximate solutions to more realistic problems

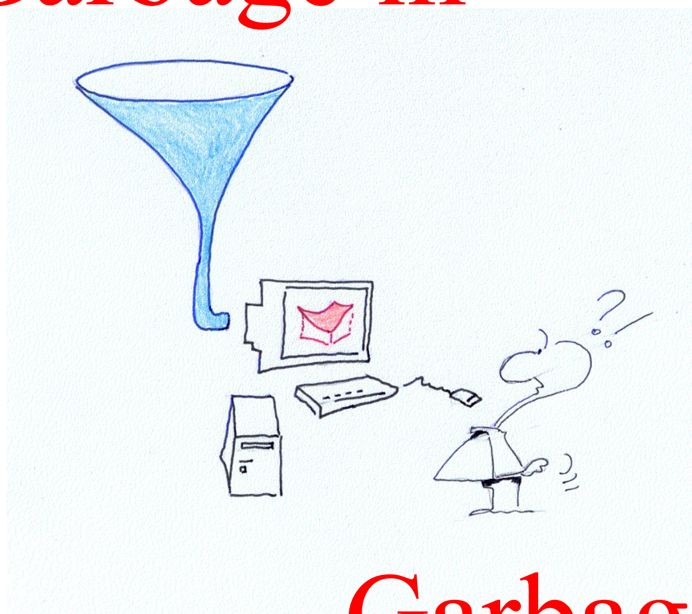
FEM developed in the sixties for linear elasticity and generalized to many other applications…



Powerful and flexible
High (cheap) computer's cost
Low (expensive) engineer's cost
Complex processes can be analyzed
Complex material laws can be included

However...

Garbage in



Garbage out

Illusion of non-qualified users to be able to analyze everything

New modeling issues requires higher qualifications…

How to define complex problems in an accurate and efficient way for the computer software ?

A l'issue de ce cours, vous serez capables de…

- **Comprendre la méthode des éléments finis**
- **Réaliser un petit programme en C**
- **Certifier et valider une simulation**
- **Choisir la voie numérique la plus efficace**
- **Estimer la précision d'un résultat**
- **Découvrir les joies et les aléas du numérique**

Non, non : ceci on ne fera pas !

- **Apprendre le génie logiciel de l'orienté-objet**
- **Utiliser des logiciels commerciaux**
- **Faire de l'analyse numérique théorique**
- **Faire du calcul parallèle**
- **Résoudre les équations de Navier-Stokes**
- **Créer automatiquement des maillages**

Learning Outcomes

Plan du cours et évaluation

Comment intégrer numériquement une fonction sur un carré ?

Comment prédire un tsunami ?

A = Evaluation continue

S2-S10 : 8 cours et 8 petits problèmes *100%*

B = Evaluation certificative

S2-S10 : séances d'exercices
S11-S13 : mini-projet *50%*
S11 : interrogation *50%*

En juin, note finale = min((A+B)/2,B)
En septembre, examen oral + projet spécial

Evaluation

Objectifs du projet

Réaliser	**Créer une application pour prédire un tsunami.**
Certifier	**Tester et valider le travail de votre groupe.**
Expliquer	**Expliquer de manière efficace et rapide à l'enseignant et aux autres étudiants ce que vous avez réalisé.**
Comprendre	**Comprendre ce que vous avez réalisé.** **Comprendre ce que d'autres groupes ont réalisé.**

Exercices : 8 petits problèmes

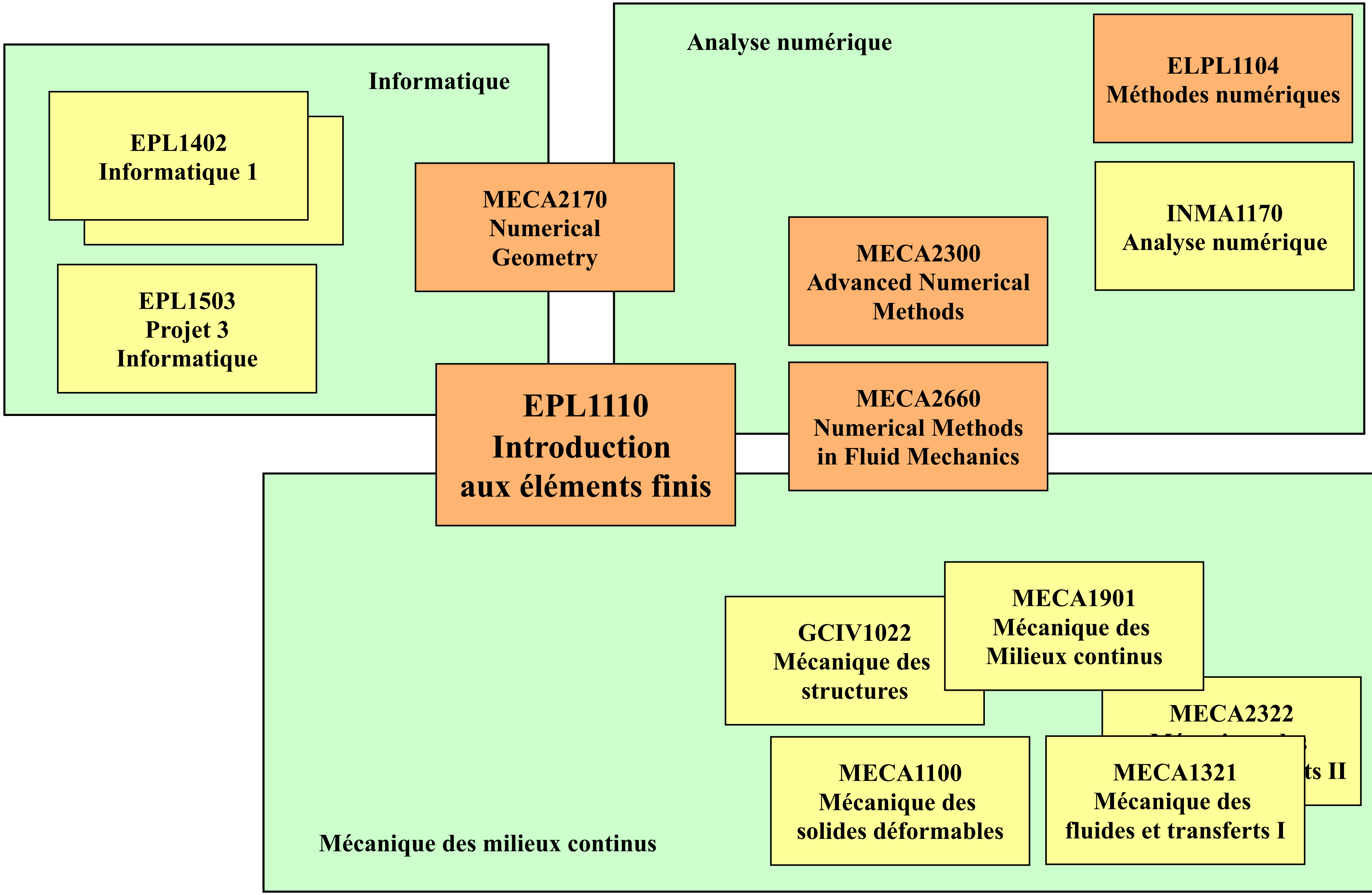
Quelques petits problèmes élémentaires pour apprivoiser le C

Projet en C :

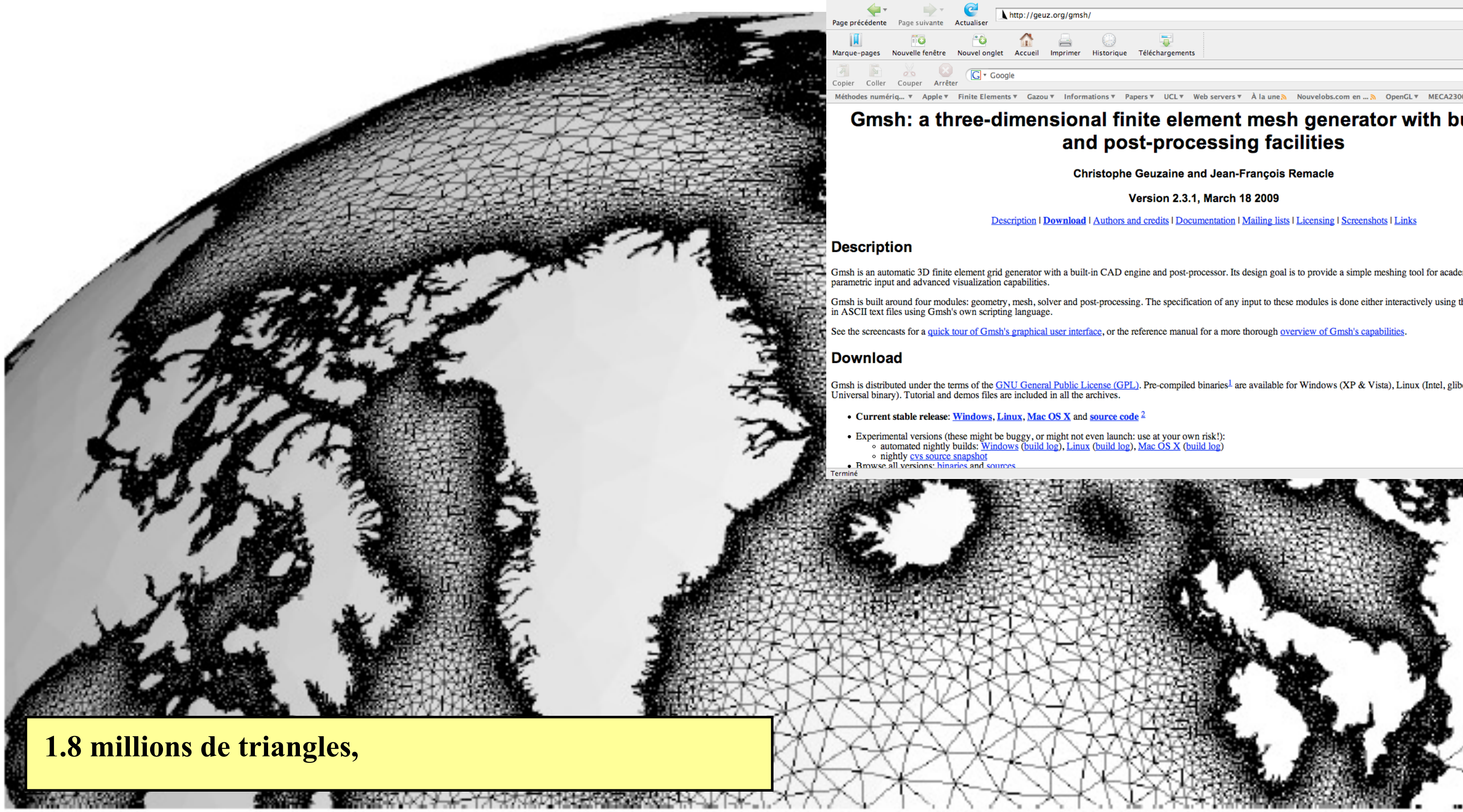
Une petite application efficace pour prédire un tsunami…

The Practice of Programming :
Simplicity, Clarity, Generality. (B.K Kernighan & R. Pike 99)

Et les autres cours….



Second-generation Louvain-la-Neuve Ice-ocean Model



Gmsh: a three-dimensional finite element mesh generator with built-in pre- and post-processing facilities

Christophe Geuzaine and Jean-François Remacle

Version 2.3.1, March 18 2009

Description | **Download** | Authors and credits | Documentation | Mailing lists | Licensing | Screenshots | Links

Description

Gmsh is an automatic 3D finite element grid generator with a built-in CAD engine and post-processor. Its design goal is to provide a simple meshing tool for academic problems with parametric input and advanced visualization capabilities.

Gmsh is built around four modules: geometry, mesh, solver and post-processing. The specification of any input to these modules is done either interactively using the graphical user interface or in ASCII text files using Gmsh's own scripting language.

See the screencasts for a quick tour of Gmsh's graphical user interface, or the reference manual for a more thorough overview of Gmsh's capabilities.

Download

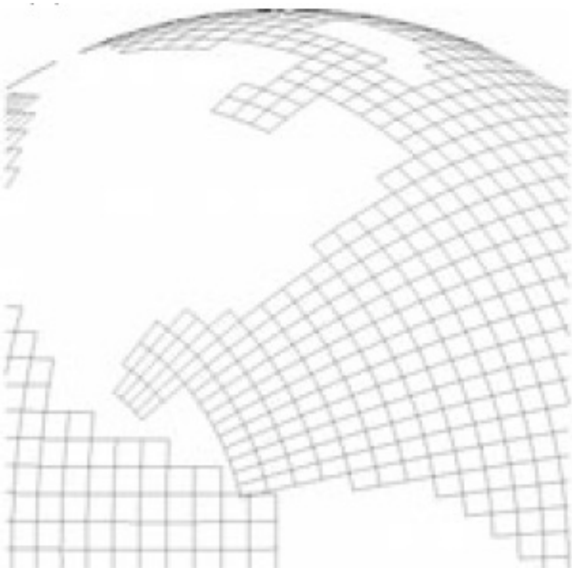
Gmsh is distributed under the terms of the GNU General Public License (GPL). Pre-compiled binaries[1] are available for Windows (XP & Vista), Linux (Intel, glibc 2.3) and Mac OS X (10.5, Universal binary). Tutorial and demos files are included in all the archives.

- **Current stable release**: **Windows**, **Linux**, **Mac OS X** and **source code** [2]
- Experimental versions (these might be buggy, or might not even launch: use at your own risk!):
  - automated nightly builds: Windows (build log), Linux (build log), Mac OS X (build log)
  - nightly cvs source snapshot
- Browse all versions: binaries and sources

Terminé

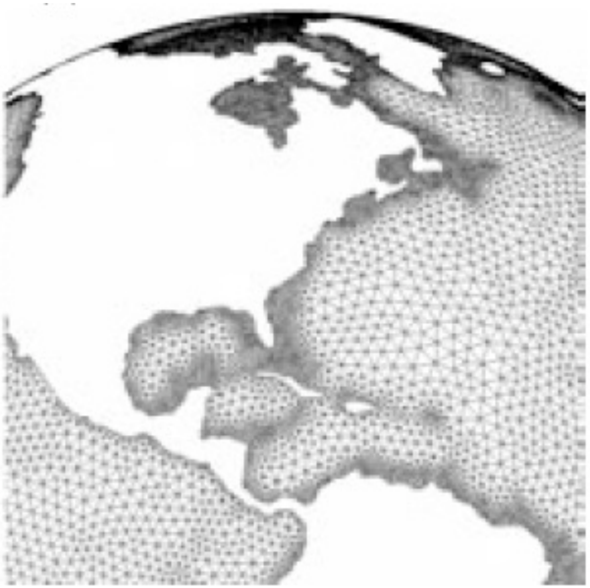
1.8 millions de triangles,

Structured grid …



- Finite differences are easy to implement
- Programming is easy
- Well known in the world of oceanography
- Bad representation of the coastlines
- Difficult to enhance locally the resolution
- Poles singularity

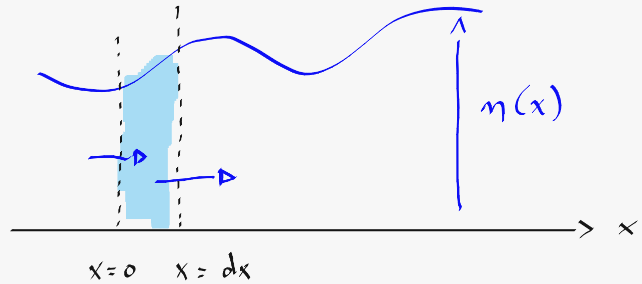
…versus unstructured grid



- Numerical methods are more complicated
- Programming is more complicated
- Not well known in the world of oceanography
- Accurate representation of the coastlines
- Enhancing the resolution is flexible
- No singular points

1 MODELE MATHEMATIQUE DU TSUNAMI

= MODELE DES EAUX PEU PROFONDES



- CONSERVATION DE LA MASSE

$$dx\,\rho\,\frac{\partial \eta}{\partial t} = \rho\,\eta_0\,u(0) - \rho\,\eta_0\,u(dx)$$

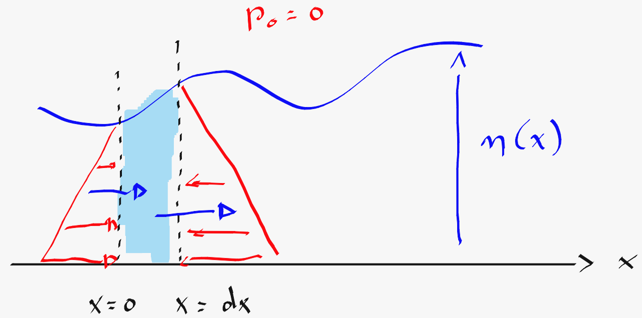
$[kg/m^2]$

$[kg/s]$

$$\frac{\partial \eta}{\partial t} = \eta_0\left[\frac{u(0) - u(dx)}{dx}\right]$$

$$\boxed{\frac{\partial \eta}{\partial t} + \eta_0\,\frac{\partial u}{\partial x} = 0}$$

• CONSERVATION QUANTITE DE MVT



$$dx\left(\rho\, \eta_0 \frac{\partial u}{\partial t}\right) = \rho g \left[\frac{\eta^2(0)}{2}\right] - \rho g \left[\frac{\eta^2(dx)}{2}\right]$$

$$\eta_0 \frac{\partial u}{\partial t} + g \frac{\partial}{\partial x}\left[\frac{\eta^2}{2}\right] = 0$$

$$g \eta \frac{\partial \eta}{\partial x} \simeq g \eta_0 \frac{\partial \eta}{\partial x}$$

$$\boxed{\frac{\partial u}{\partial t} + g \frac{\partial \eta}{\partial x} = 0}$$

$$\frac{\partial \eta}{\partial t} + \eta_0 \frac{\partial u}{\partial x} = 0 \quad \longrightarrow \quad \frac{\partial^2 u}{\partial t \, \partial x} + \frac{1}{\eta_0} \frac{\partial^2 \eta}{\partial t^2} = 0$$

$$\frac{\partial u}{\partial t} + g \frac{\partial \eta}{\partial x} = 0 \quad \longrightarrow \quad \frac{\partial^2 u}{\partial t \, \partial x} + g \frac{\partial^2 \eta}{\partial x^2} = 0$$

$$\frac{\partial^2 \eta}{\partial t^2} = \eta_0 \, g \, \frac{\partial^2 \eta}{\partial x^2}$$

EQUATION D'ONDE

$$c^2 = \eta_0 g$$

$$c = \sqrt{\eta_0 g}$$

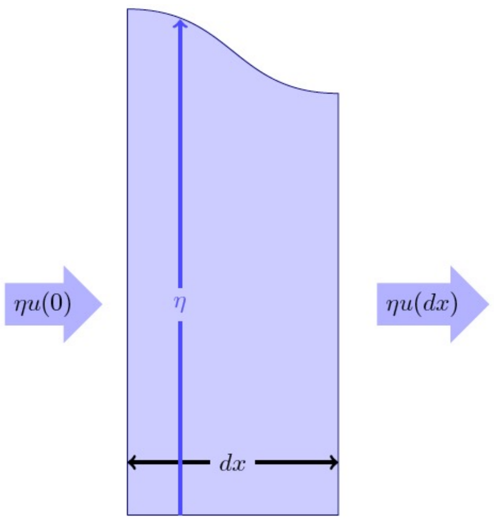
η_0: 4000 m ; g: 10 m/s^2

$$c = 200 \text{ m/s} = 200 \, \frac{3600}{1000}$$

720 km/h

Mass balance

$$\frac{\partial \eta}{\partial t} + \eta_0 \frac{\partial u}{\partial x} = 0$$



$$dx\left(\rho\frac{\partial \eta}{\partial t}\right) = \rho\eta u(0) - \rho\eta u(dx)$$

$$\frac{\partial \eta}{\partial t} + \eta_0\left(\frac{u(dx)-u(0)}{dx}\right) = 0$$

$$\downarrow$$

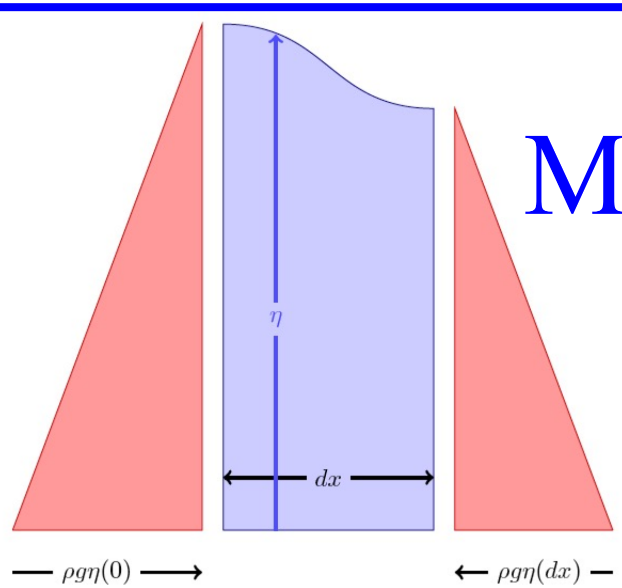
$$\frac{\partial \eta}{\partial t} + \eta_0 \frac{\partial u}{\partial x} = 0$$

$$dx\left(\rho\eta_0\frac{\partial u}{\partial t}\right) = \rho g\frac{\eta^2(0)}{2} - \rho g\frac{\eta^2(dx)}{2}$$

$$\eta_0\frac{\partial u}{\partial t} + g\eta_0\frac{\partial \eta}{\partial x} = 0$$

$$\downarrow$$

$$\frac{\partial u}{\partial t} + g\frac{\partial \eta}{\partial x} = 0$$



Momentum balance

$$\frac{\partial u}{\partial t} + g\frac{\partial \eta}{\partial x} = 0$$

$$\begin{cases} \dfrac{1}{\eta_0}\dfrac{\partial \eta}{\partial t} + \dfrac{\partial u}{\partial x} = 0 \\ \dfrac{\partial u}{\partial t} + g\dfrac{\partial \eta}{\partial x} = 0 \end{cases}$$

Linear Shallow Water Equations

$$\frac{\partial^2 \eta}{\partial t^2} = g\eta_0 \frac{\partial^2 \eta}{\partial x^2}$$

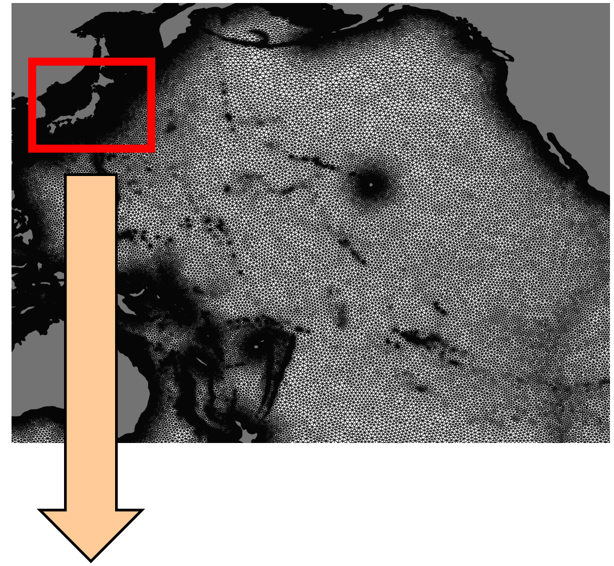
Wave Equation

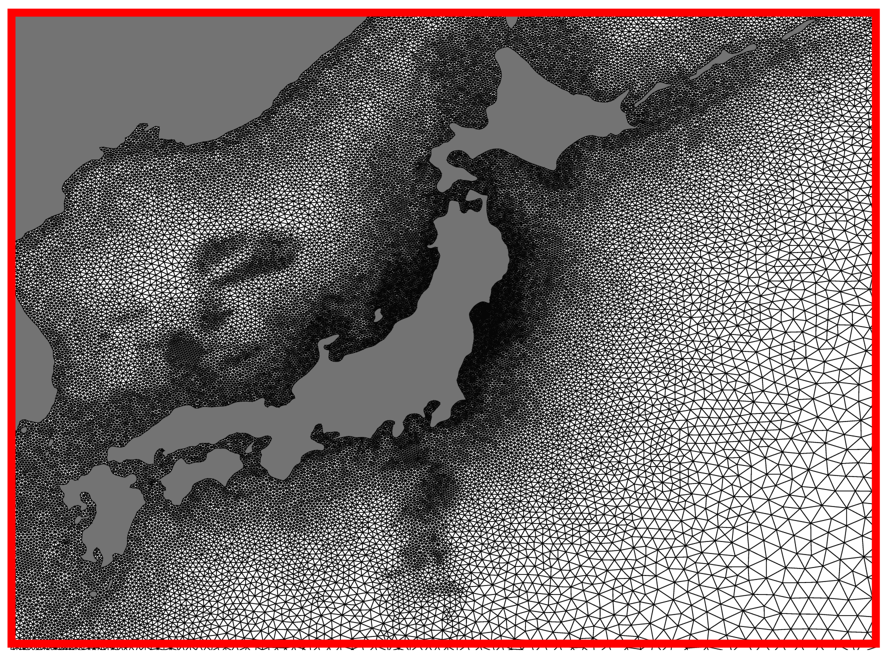
$$c = \sqrt{g\eta_0} \approx 200 \; [m/s]$$

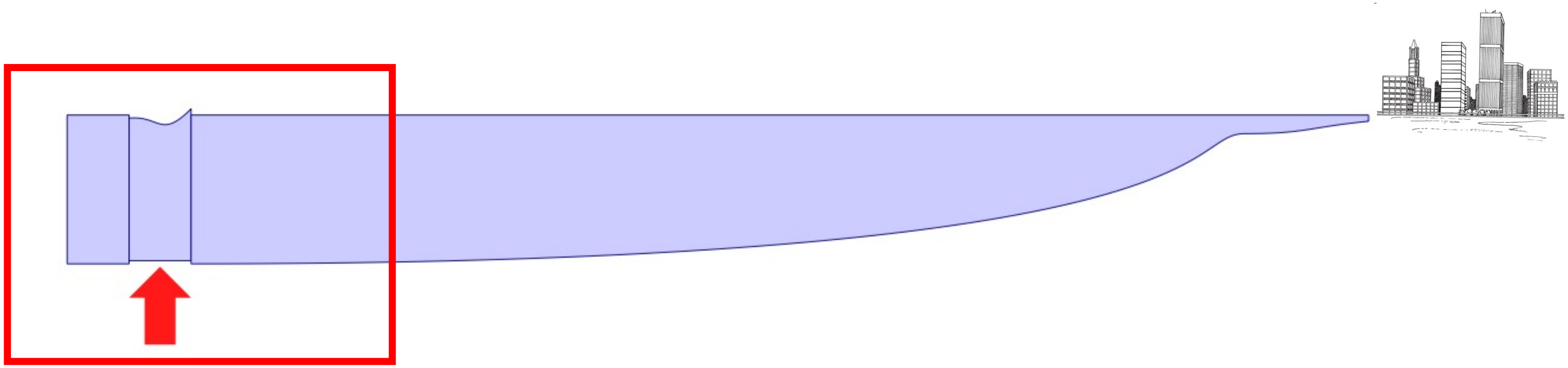
Gravity 9,81 m/s
Average depth of Pacific 4000 m

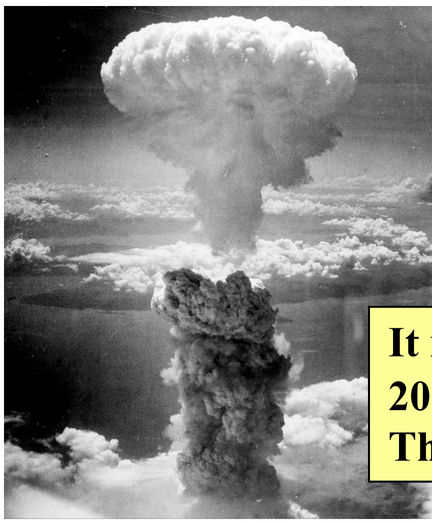
Waves are (very) fast !

And now,
we can zoom
on Japan !



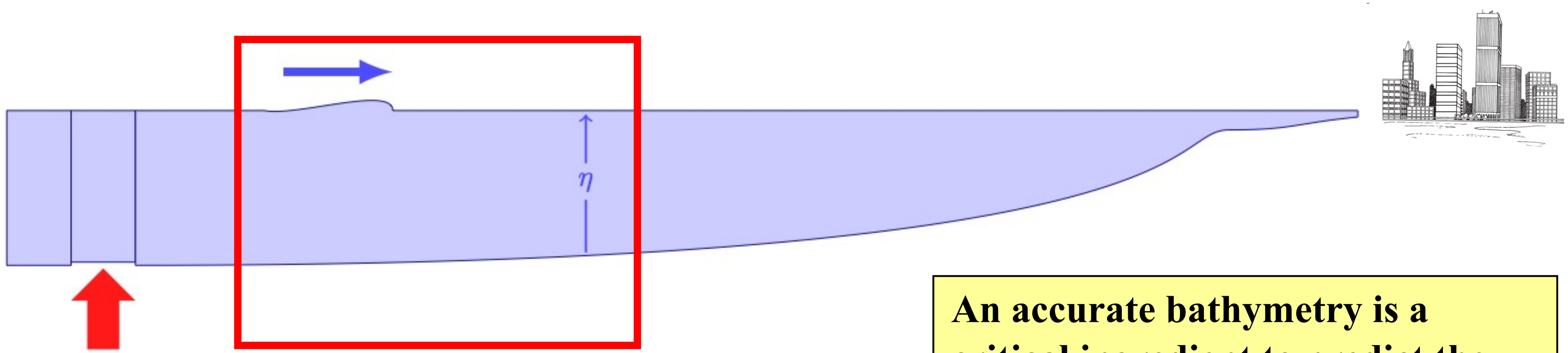






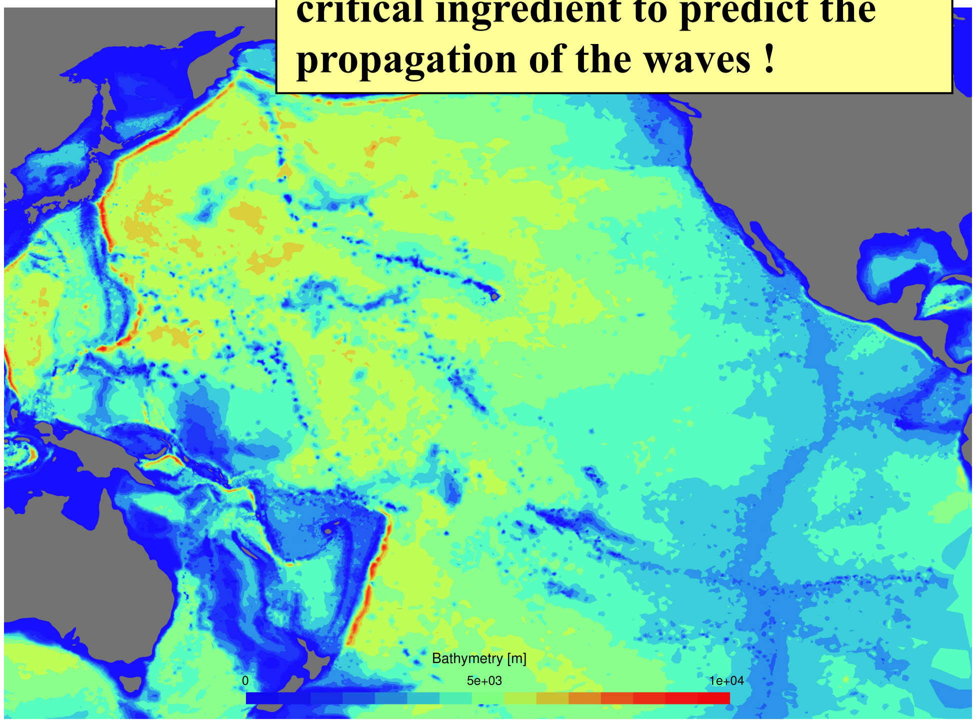
It is a huge energy !
20 x Energy of Hiroshima's bomb !
This initial condition must be provided to the model

The earthquake motion displaces a column water

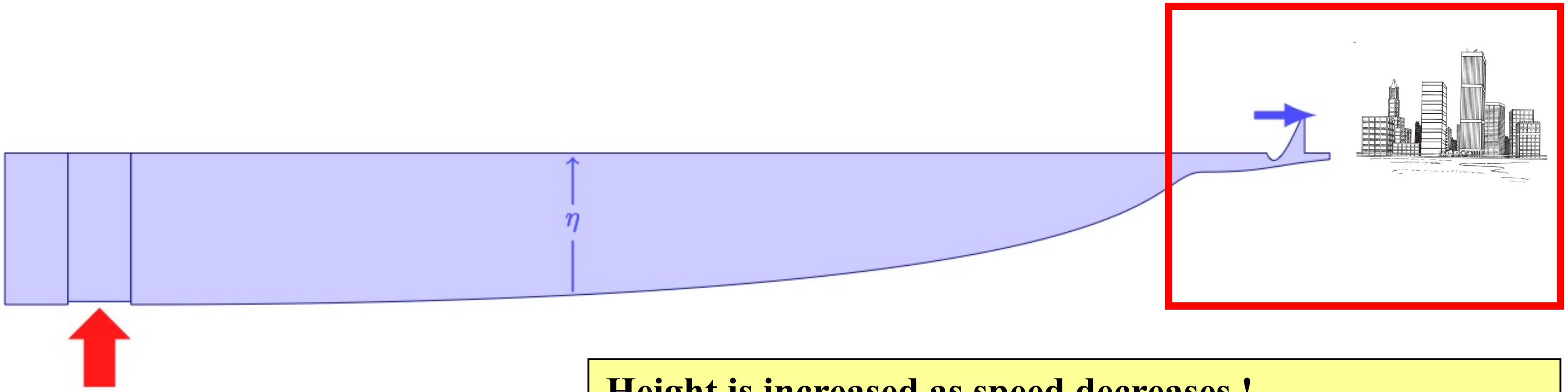


$$c = \sqrt{g\eta_0} \approx 200 \ [m/s]$$

An accurate bathymetry is a critical ingredient to predict the propagation of the waves !



Small waves travel fast as function of the bathymetry

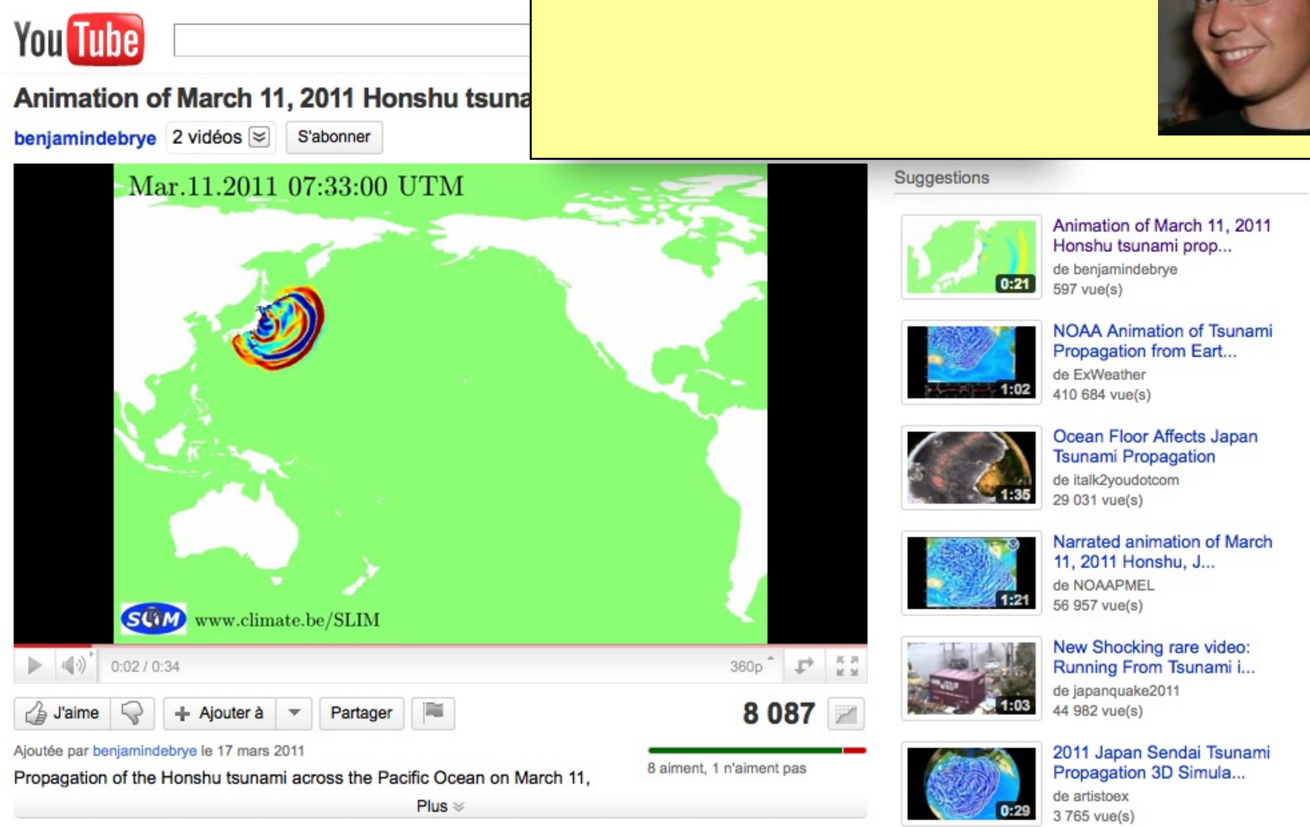


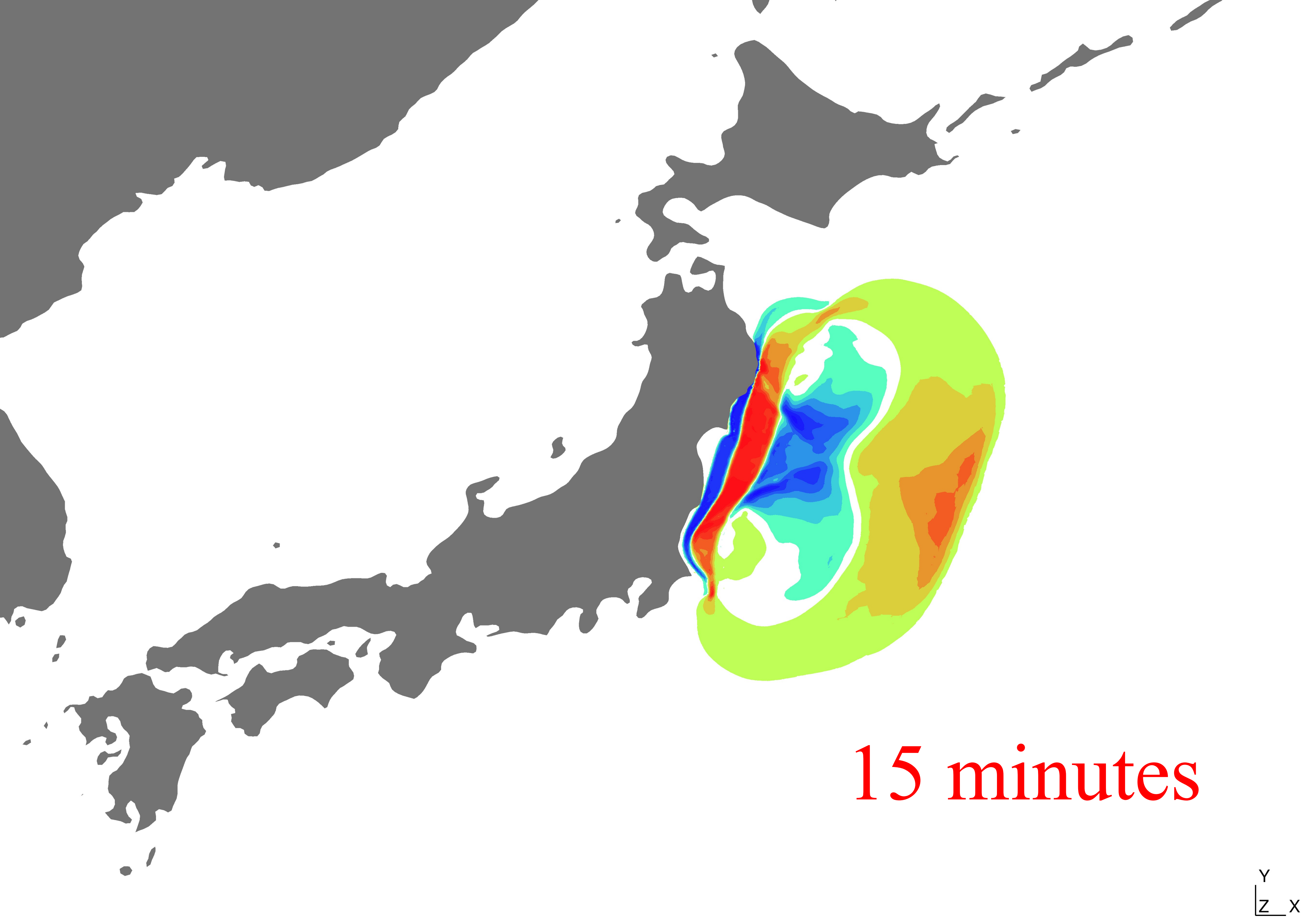
Height is increased as speed decreases !
An accurate shoreline description is required.

Waves compression forces waves to gain height !

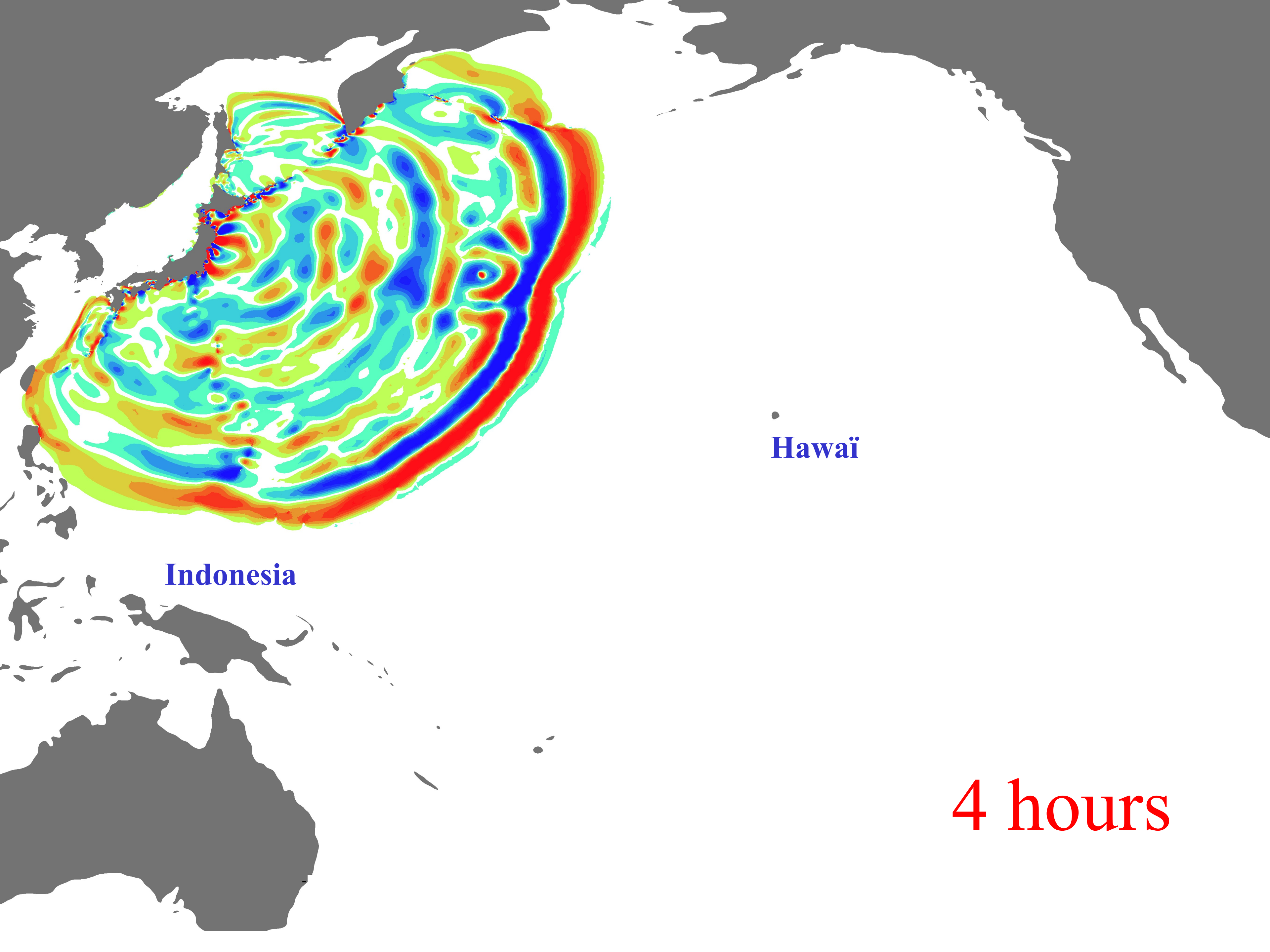
Simulating tsunamis is easy

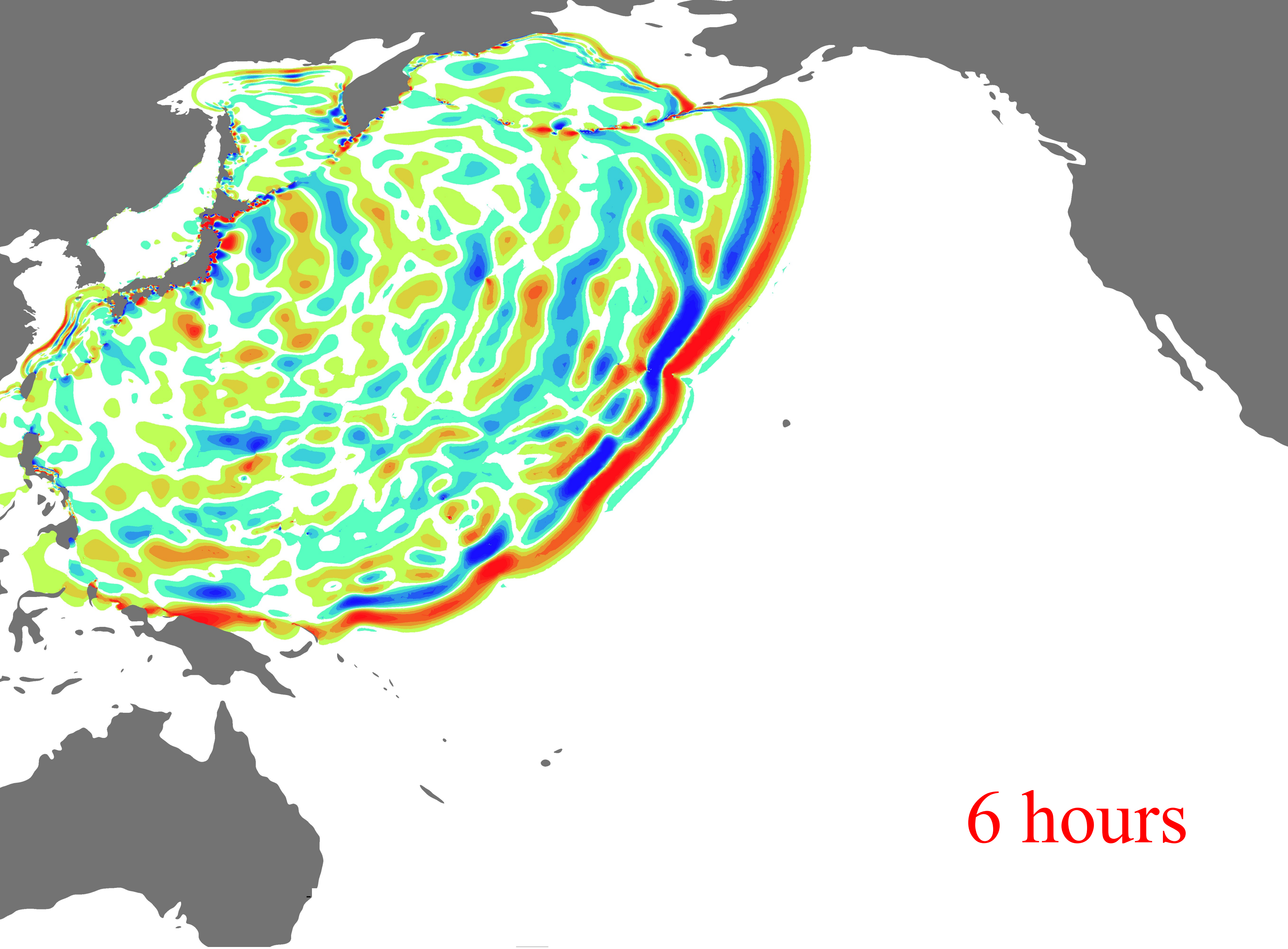
Simulation performed on May 17th by Jonathan Lambrechts and Benjamin de Brye



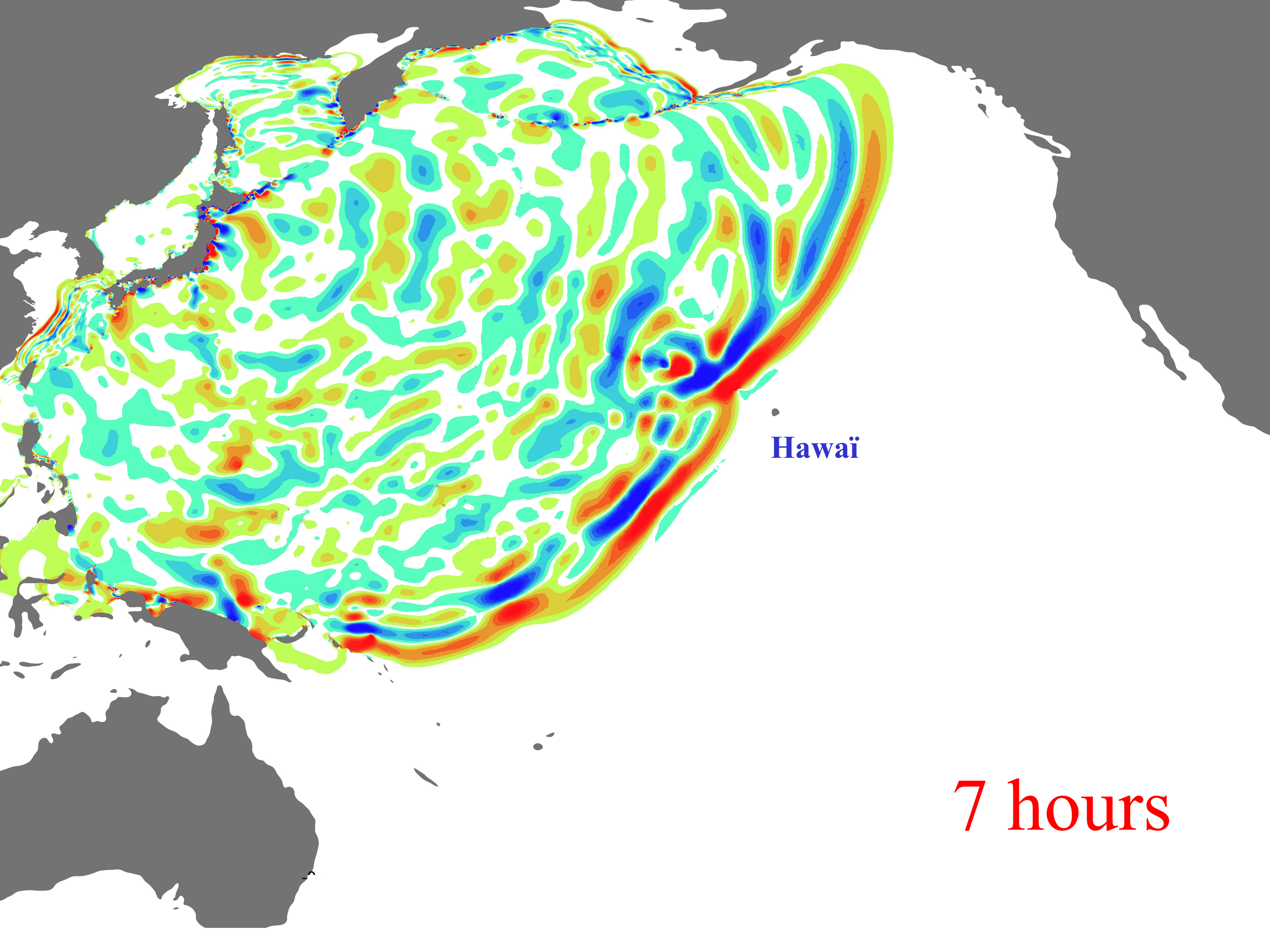

15 minutes
Y
Z
X

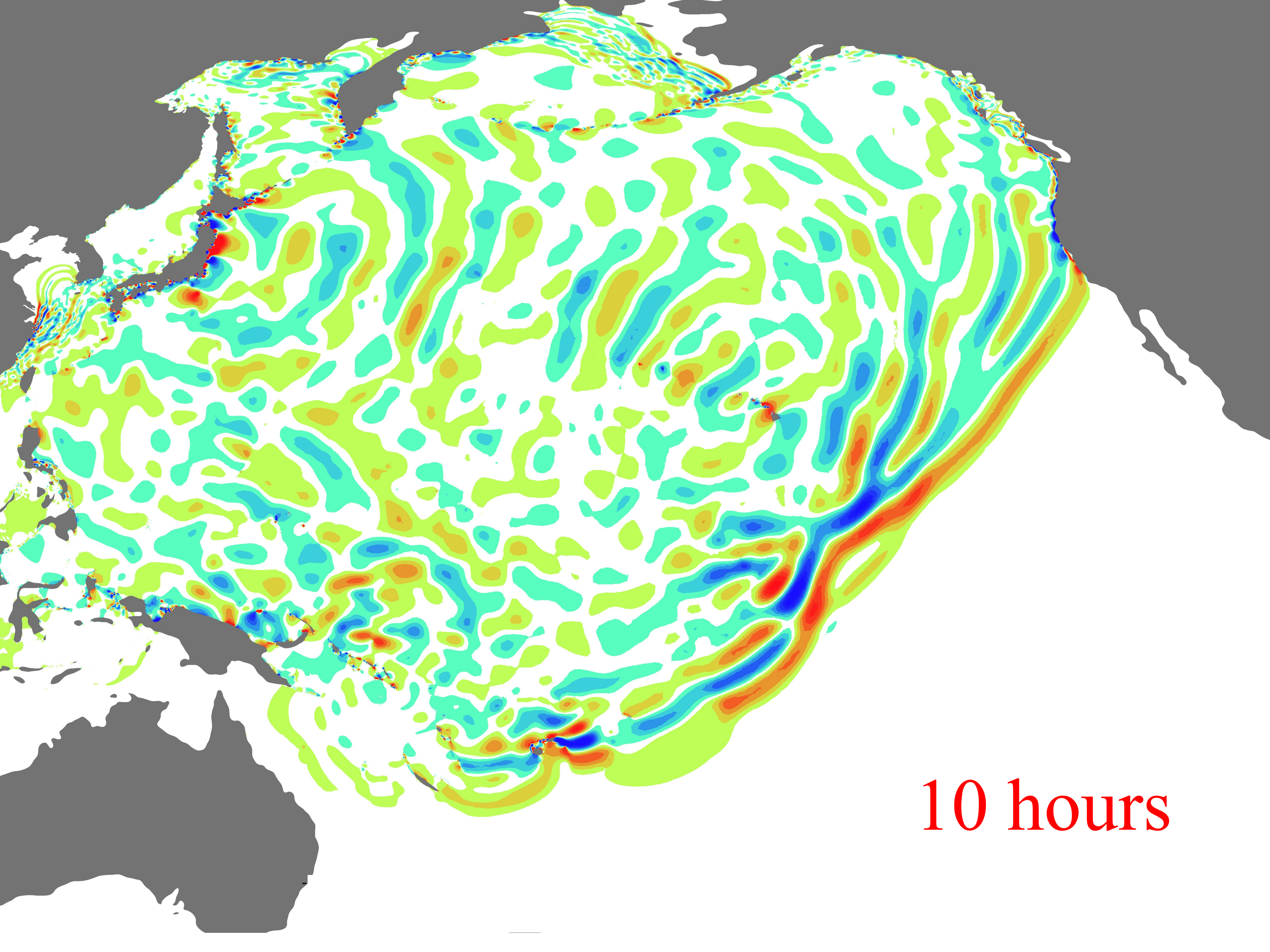

1 hour


Hawaï
Indonesia
4 hours



6 hours


Hawaï
7 hours


10 hours

hello.c

```c
#include <stdio.h>

int main()
{
  printf("hello, world\n");
  return 0;
}
```

Vous aimeriez apprendre à programmer,
mais vous ne savez pas par où commencer ?

(autrement dit : vous en avez marre des cours trop compliqués que vous ne comprenez pas ? :-)

```
Cours1 — -bash — 71×16
Last login: Fri Feb  4 08:32:27 on ttys001
(base) mac-1U0-352:~ vl$ cd LaTeX/epl1110-2122/Slides/Cours1
(base) mac-1U0-352:Cours1 vl$ ls
a.out   hello.c
(base) mac-1U0-352:Cours1 vl$ touch hello.c
(base) mac-1U0-352:Cours1 vl$ cc hello.c
(base) mac-1U0-352:Cours1 vl$ ./a.out
hello, world
(base) mac-1U0-352:Cours1 vl$
```

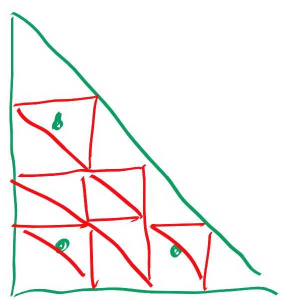
Comment compiler hello.c sur votre ordinateur ?

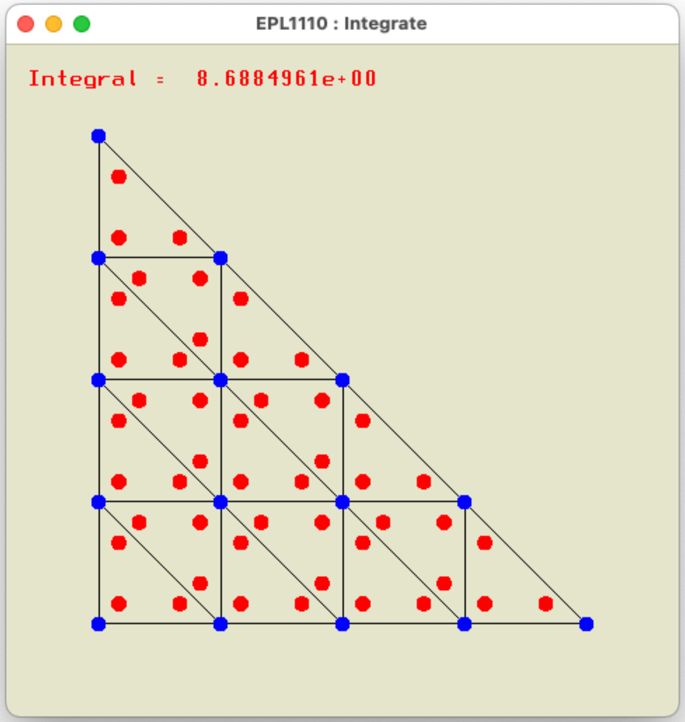
Cours1 — -bash — 93×27

```
(base) mac-1U0-352:Cours1 vl$
(base) mac-1U0-352:Cours1 vl$
(base) mac-1U0-352:Cours1 vl$
(base) mac-1U0-352:Cours1 vl$
(base) mac-1U0-352:Cours1 vl$
(base) mac-1U0-352:Cours1 vl$
(base) mac-1U0-352:Cours1 vl$
(base) mac-1U0-352:Cours1 vl$
(base) mac-1U0-352:Cours1 vl$
(base) mac-1U0-352:Cours1 vl$
(base) mac-1U0-352:Cours1 vl$
(base) mac-1U0-352:Cours1 vl$
(base) mac-1U0-352:Cours1 vl$
(base) mac-1U0-352:Cours1 vl$
(base) mac-1U0-352:Cours1 vl$
(base) mac-1U0-352:Cours1 vl$
(base) mac-1U0-352:Cours1 vl$ touch hello.c
(base) mac-1U0-352:Cours1 vl$ cc -c hello.c
(base) mac-1U0-352:Cours1 vl$ ls
hello.c hello.o
(base) mac-1U0-352:Cours1 vl$ cc -o hello hello.c
(base) mac-1U0-352:Cours1 vl$ ./hello
hello, world
(base) mac-1U0-352:Cours1 vl$ cc -o yep hello.o
(base) mac-1U0-352:Cours1 vl$ ./yep
hello, world
(base) mac-1U0-352:Cours1 vl$
```

Homework 1

$$\underbrace{\int_{\hat{\Omega}} f(x,y)\;dx\;dy}_{I} \approx \underbrace{\sum_{k=1}^{3} w_k\; f(x_k,y_k)}_{I_h}$$





Ecrire la règle de Hammer

mainBasic.c

```
#include <stdio.h>
#include <math.h>


double integrate(double x[3], double y[3], double(*f)(double,double));
double integrateRecursive(double x[3], double y[3], double(*f)(double,double), int n);

double fun(double x, double y)  { return cos(x) + y * y; }
double stupid(double x, double y)  { return 1.0; }


int main()
{
    double x[3] = {0, 1, 0};
    double y[3] = {0, 0, 1};
    int n;
    printf("Surface integration    : %14.7e \n", integrate(x,y,stupid));
    printf("More funny integration : %14.7e \n", integrate(x,y,fun));
    for (n=0;  n <= 4; n++) {
        double I = integrateRecursive(x,y,fun,n);
        printf("Recursive integration (n = %2d) : %14.7e \n", n,I);  }
    return 0;
}
```

homework.c et mainBasic.c

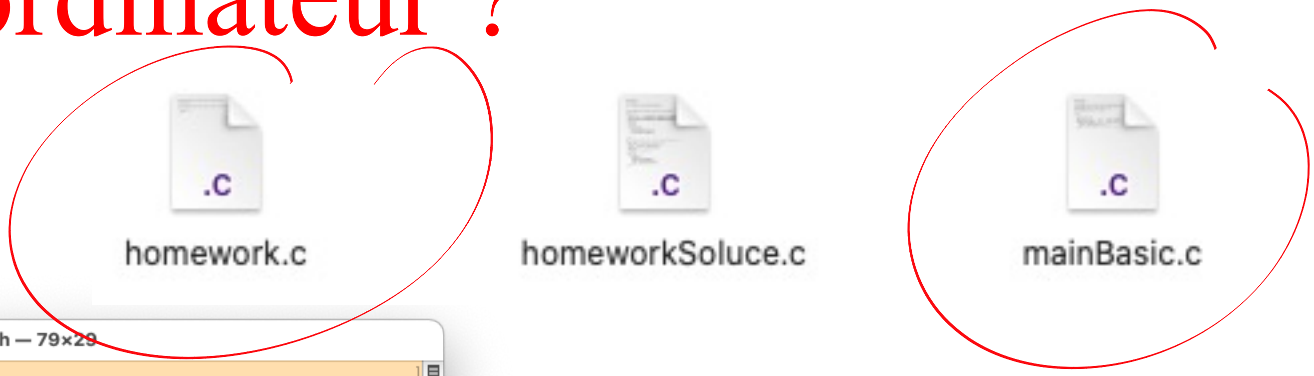
```
double integrate(double x[3], double y[3], double (*f) (double, double))
{
    double I = 3.14;
    return I;
}

double integrateRecursive(double x[3], double y[3], double (*f)(double,double), int n)
{
    double I = 0.0;
    return I;
}
```

```
double fun(double x, double y)  { return cos(x) + y * y; }
double stupid(double x, double y)  { return 1.0; }


int main()
{
    double x[3] = {0, 1, 0};
    double y[3] = {0, 0, 1};
    int n;
    printf("Surface integration    : %14.7e \n", integrate(x,y,stupid));
    printf("More funny integration : %14.7e \n", integrate(x,y,fun));
    for (n=0;  n <= 1; n++) {
        double I = integrateRecursive(x,y,fun,n);
        printf("Recursive integration (n = %2d) : %14.7e \n", n,I);  }
    return 0;
}
```

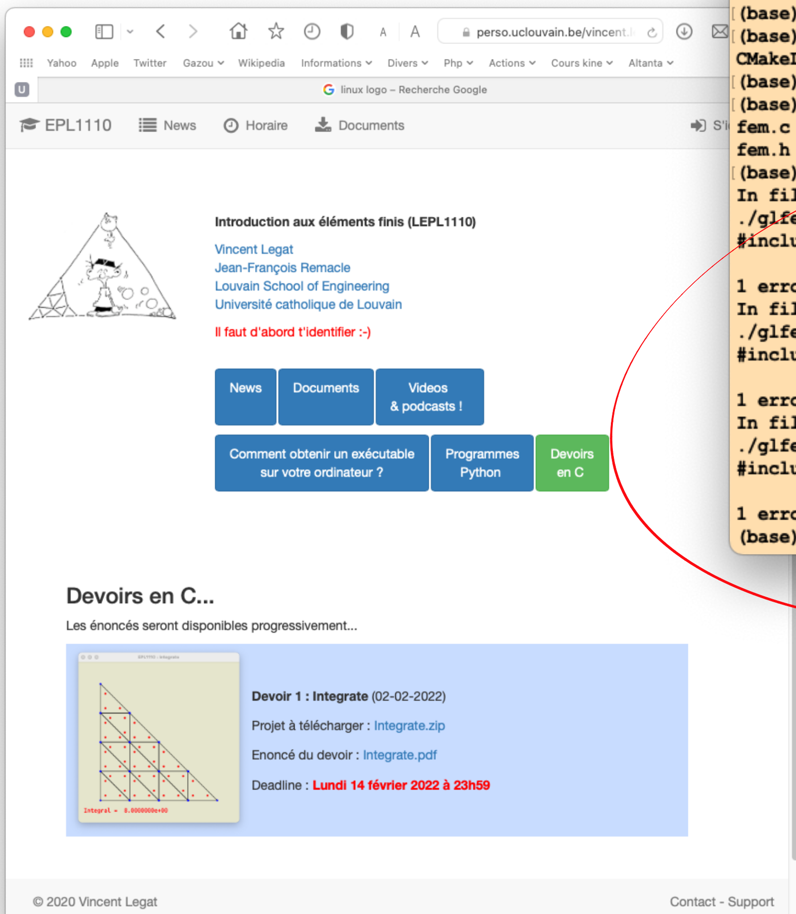
Comment compiler le devoir sur votre ordinateur ?

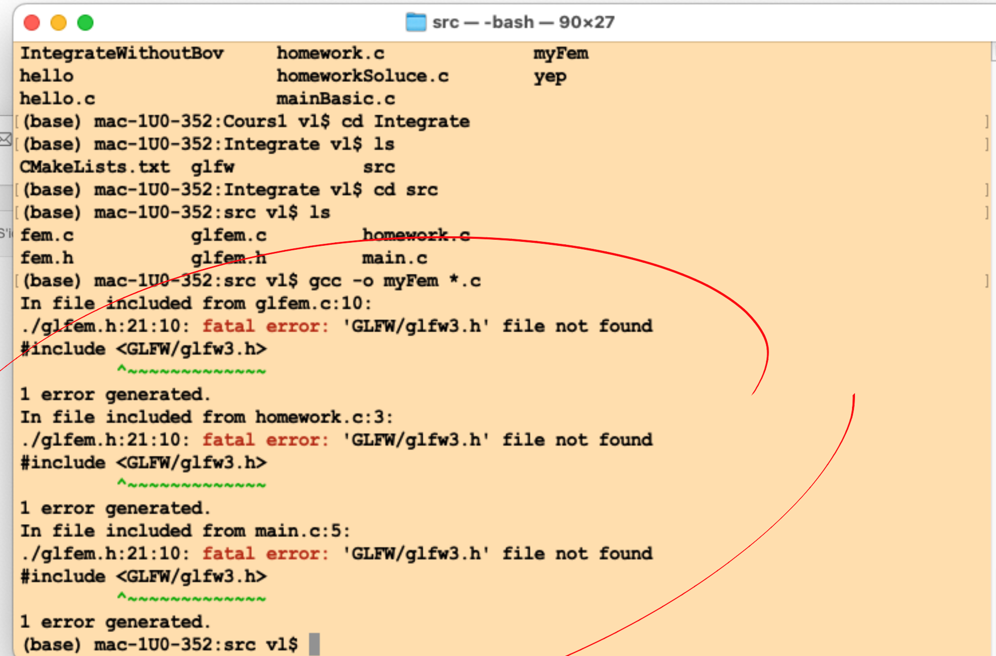


Cours1 — -bash — 79×29

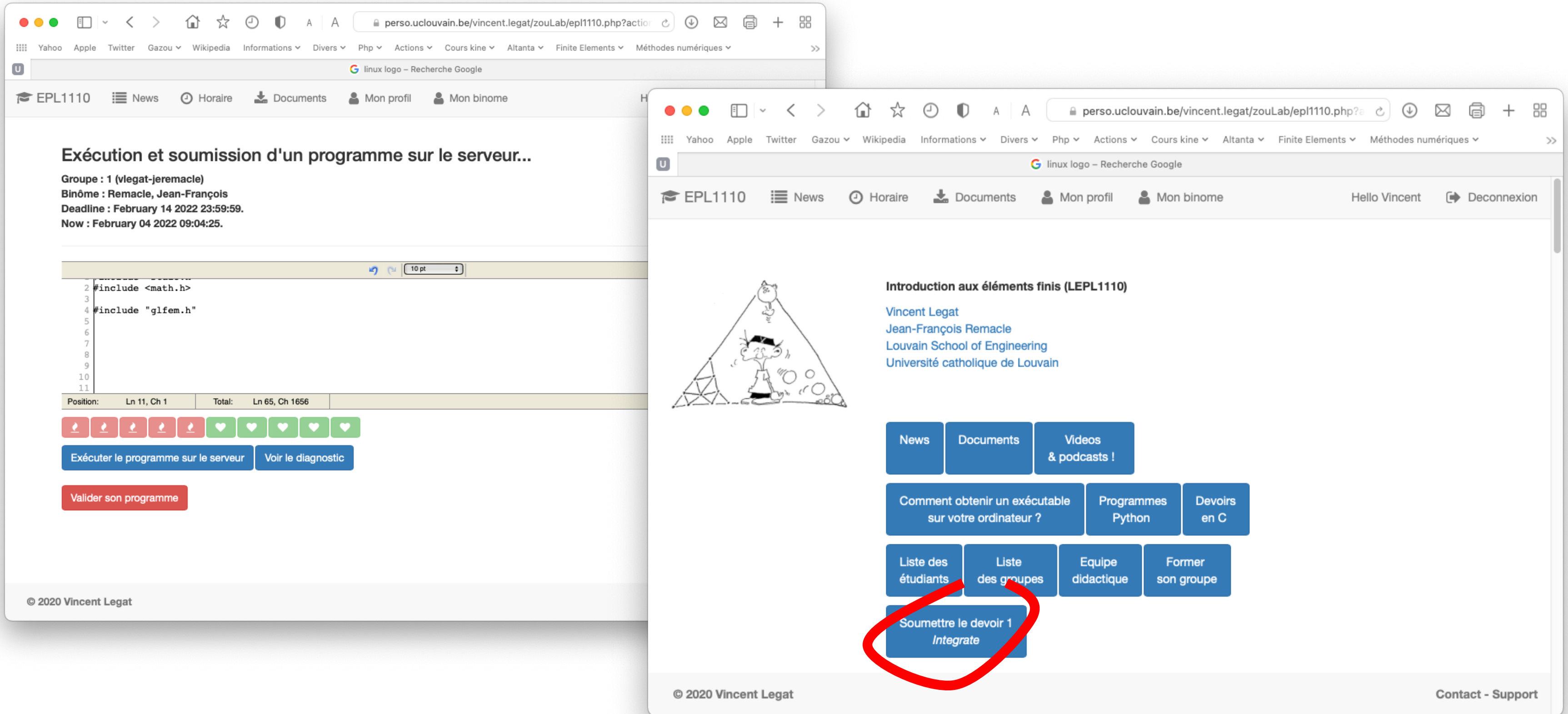
```
(base) mac-1U0-352:Cours1 vl$
(base) mac-1U0-352:Cours1 vl$
(base) mac-1U0-352:Cours1 vl$
(base) mac-1U0-352:Cours1 vl$
(base) mac-1U0-352:Cours1 vl$
(base) mac-1U0-352:Cours1 vl$
(base) mac-1U0-352:Cours1 vl$
(base) mac-1U0-352:Cours1 vl$
(base) mac-1U0-352:Cours1 vl$
(base) mac-1U0-352:Cours1 vl$
(base) mac-1U0-352:Cours1 vl$ cc -o myFem mainBasic.c homework.c
(base) mac-1U0-352:Cours1 vl$ ./myFem
Surface integration     :  3.1400000e+00
More funny integration :  3.1400000e+00
Recursive integration (n =  0) :  0.0000000e+00
Recursive integration (n =  1) :  0.0000000e+00
Recursive integration (n =  2) :  0.0000000e+00
Recursive integration (n =  3) :  0.0000000e+00
Recursive integration (n =  4) :  0.0000000e+00
(base) mac-1U0-352:Cours1 vl$ cc -o myFem mainBasic.c homeworkSoluce.c
(base) mac-1U0-352:Cours1 vl$ ./myFem
Surface integration     :  5.0000000e-01
More funny integration :  5.4302895e-01
Recursive integration (n =  0) :  5.4302895e-01
Recursive integration (n =  1) :  2.1721229e+00
Recursive integration (n =  2) :  8.6884961e+00
Recursive integration (n =  3) :  3.4753986e+01
Recursive integration (n =  4) :  1.3901594e+02
(base) mac-1U0-352:Cours1 vl$
```

Bon : c'est aussi simple ?



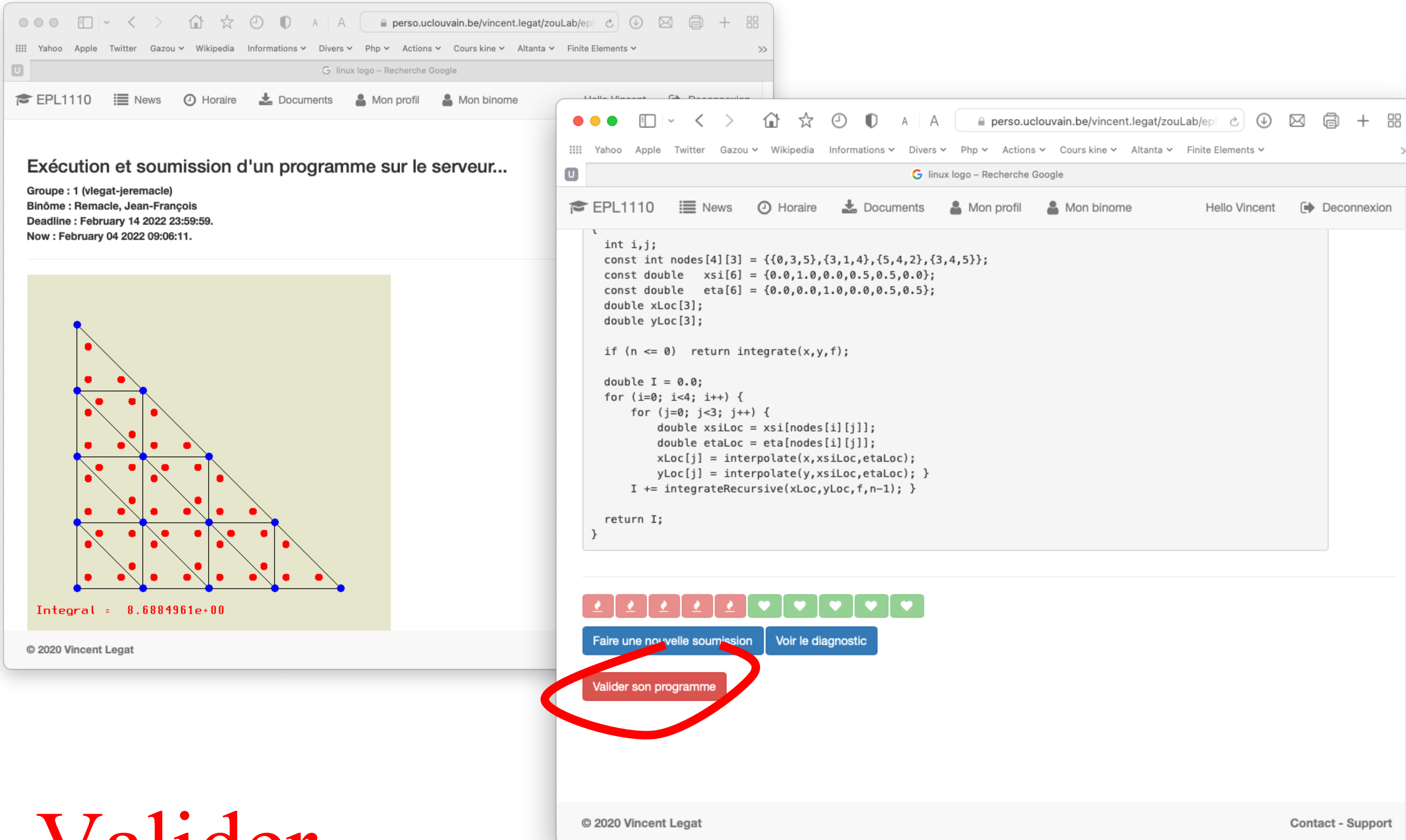


Eh non !



Soumettre votre devoir !

Tous les devoirs seront corrigés de manière automatique !
Bien veiller à ce que la version soumise soit bien compilée !
Aucune soumission tardive ne sera admise !
Les devoirs sont réalisés individuellement !



Valider
et verifier son devoir !

Le C est un langage de bas niveau : les pointeurs ;-(

```c
int main(void)
{
   int a = 4;
   printf(" ====   a === %d \n",a);
   printf(" ====  &a === %d \n",&a);
   int *b = &a;  // &&a do not exist : why ?
   printf(" ==== &&a === %d \n",&b);
   exit(0);
}
```

			Adresse de la case mémoire	Valeur
int	a		1606415436	4
*int	&a	b	1606415424	1606415436
**int	&&a	&b	...	1606415424

L'utilisation des pointeurs permet d'écrire des codes très efficaces et rapides…
Par contre, programmer est une tâche plus délicate et fastidieuse.
Mais, la rapidité d'un code est critique pour la simulation numérique (et le jeux !) :

Le langage C (ou C++) est bon choix ici !

Le C est un langage de bas niveau les pointeurs ;-(

```c
int a = 0;
int *b = &a;
b[0] = 4;
printf(" ====  *b  === %d \n",*b);
printf(" ==== b[0] === %d \n",b[0]);
printf(" ====   a  === %d \n",a);
printf(" ====   b  === %d \n",b);
```

Adresse de la case mémoire

int	a	*b	b[0]	1606415436	4
*int	&a	b		1606415424	1606415436

L'utilisation des pointeurs permet d'écrire des codes très efficaces et rapides…
Par contre, programmer est une tâche plus délicate et fastidieuse.
Mais, la rapidité d'un code est critique pour la simulation numérique (et le jeux !) :

Le langage C (ou C++) est bon choix ici !

On peut écrire n'importe où dans la mémoire de l'ordinateur ! Ouuuuuups : c'est pas joli

```c
int a = 0;
int *b = &a;
b[0] = 4;
b[1] = 3;
printf(" ====  *b   === %d \n",*b);
printf(" ==== b[0] === %d \n",b[0]);
printf(" ==== b[1] === %d \n",b[1]);
printf(" ====   a   === %d \n",a);
printf(" ====   b   === %d \n",b);
printf(" ==== &b[0] == %d \n",&b[0]);
printf(" ==== &b[1] == %d \n",&b[1]);
```

Et le pire, c'est que cela marche parfois…
Parfois pas : `Segmentation fault`

				Adresse de la case mémoire	
int			b[1]	1606415440	3
int	a	*b	b[0]	1606415436	4
*int	&a	b		1606415424	1606415436

homework.c

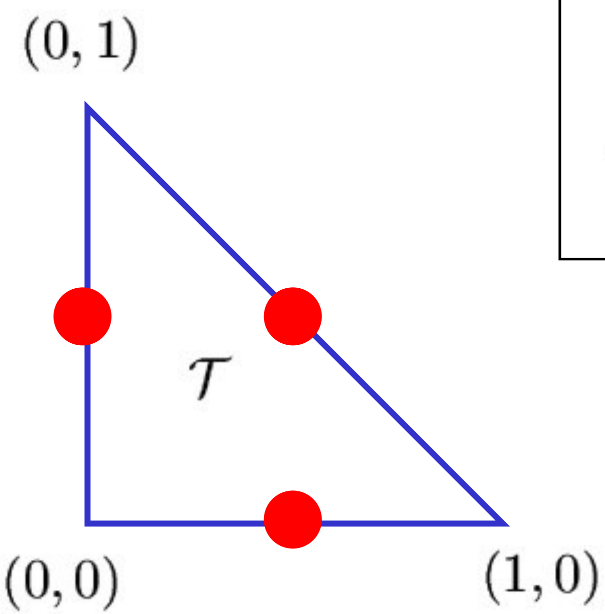
La solution doit se trouver dans le fichier homework.c uniquement.... On ne regarde jamais main.c !

```
#include <stdio.h>
#include <math.h>

double integrate(double x[3], double y[3], double (*f) (double, double))
{
    double I = 3.14;
    int I;
    for (i=0; i<3; i++)
        printf("   === node %d : %14.7e %14.7e \n",i+1,x[i],y[i]);
    return I;
}

double integrateRecursive(double x[3], double y[3], double (*f)(double,double), int n)
{
    double I = 0.0;
    return I;
}
```

Intégration sur un triangle : Règle de Hammer à 3 points



$$\underbrace{\int_{\mathcal{T}} f(x,y)\ dx\ dy}_{I} \approx \underbrace{\sum_{k=1}^{3} w_k f(X_k, Y_k)}_{I^h}$$

	X_k	Y_k	w_k
1	0.5	0.0	1/6
2	0.5	0.5	1/6
3	0.0	0.5	1/6

Interrogation (mai 2003)

Démontrer que la formule de Hammer à trois points permet d'intégrer exactement n'importe quel polynôme à deux variables de degré deux : *a + bx + cy + dx² +ey² + fxy*

Question

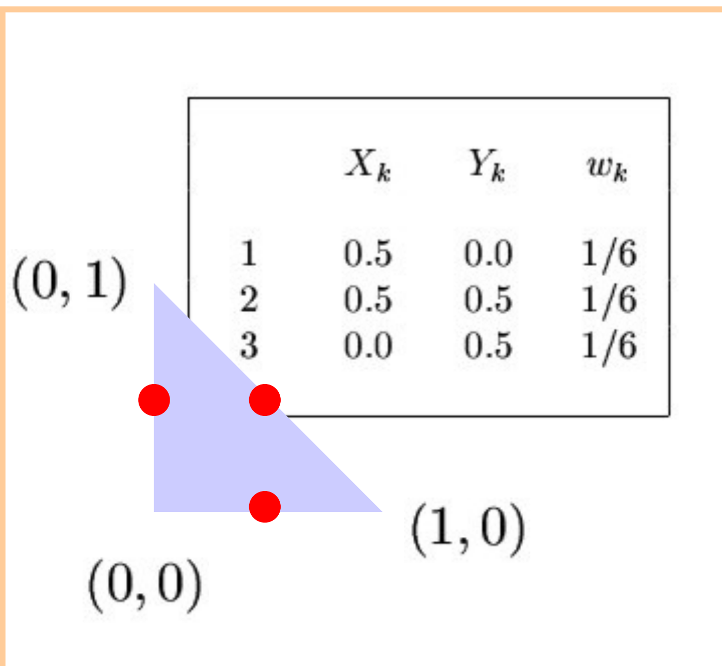
$$\begin{aligned}
I &= \frac{a}{2} + b\int_0^1 x \int_0^{1-x} dy\ dx + c\int_0^1 y\int_0^{1-y} dx\ dy \\
&\quad + d\int_0^1 x^2 \int_0^{1-x} dy\ dx + e\int_0^1 y^2 \int_0^{1-y} dx\ dy + f\int_0^1 x\int_0^{1-x} y\ dy\ dx \\
&= \frac{a}{2} + b\left[\frac{x^2}{2} - \frac{x^3}{3}\right]_0^1 + c\left[\frac{y^2}{2} - \frac{y^3}{3}\right]_0^1 \\
&\quad + d\left[\frac{x^3}{3} - \frac{x^4}{4}\right]_0^1 + e\left[\frac{y^3}{3} - \frac{y^4}{4}\right]_0^1 + f\left[\frac{x^2}{4} - \frac{x^3}{3} + \frac{x^4}{8}\right]_0^1 \\
&= \frac{a}{2} + \frac{b}{6} + \frac{c}{6} + \frac{d}{12} + \frac{e}{12} + \frac{f}{24} \\
&= I^h
\end{aligned}$$

$\square$

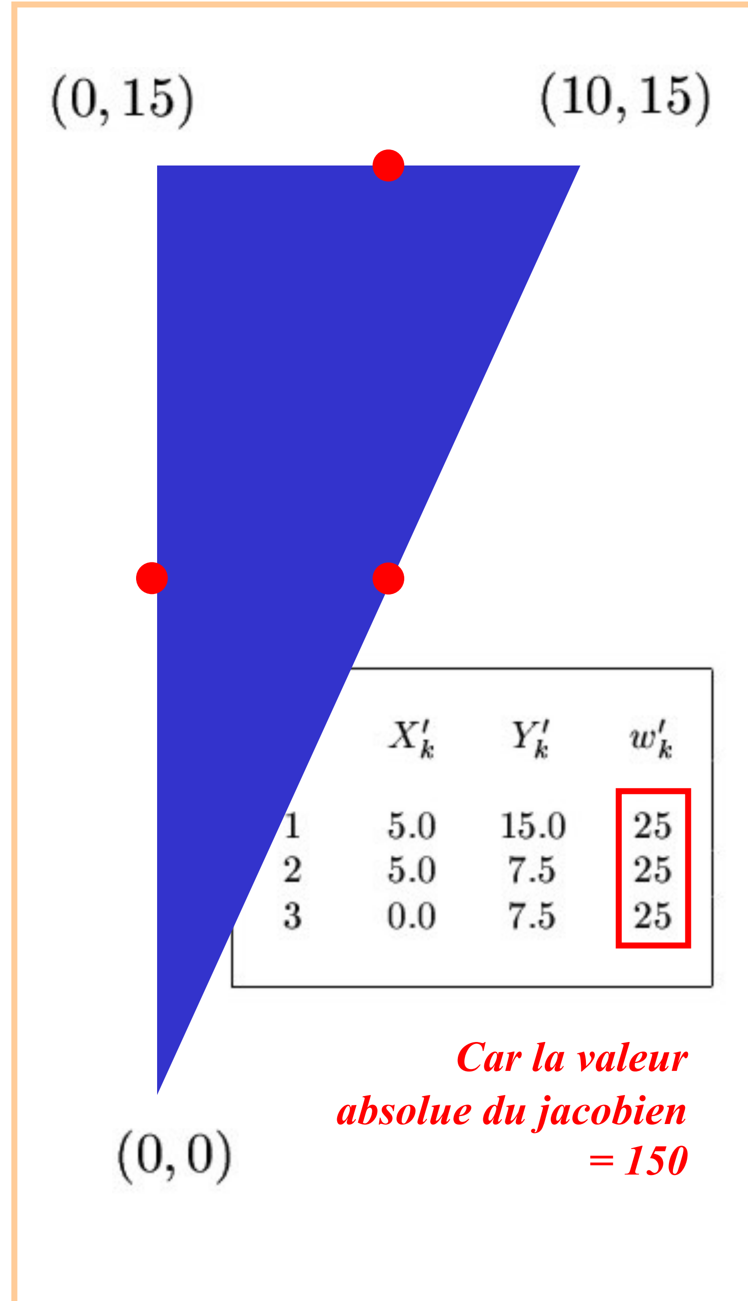
Degré de précision

Et un autre triangle ?

$$x' = 10x$$
$$y' = 15 - 15y$$



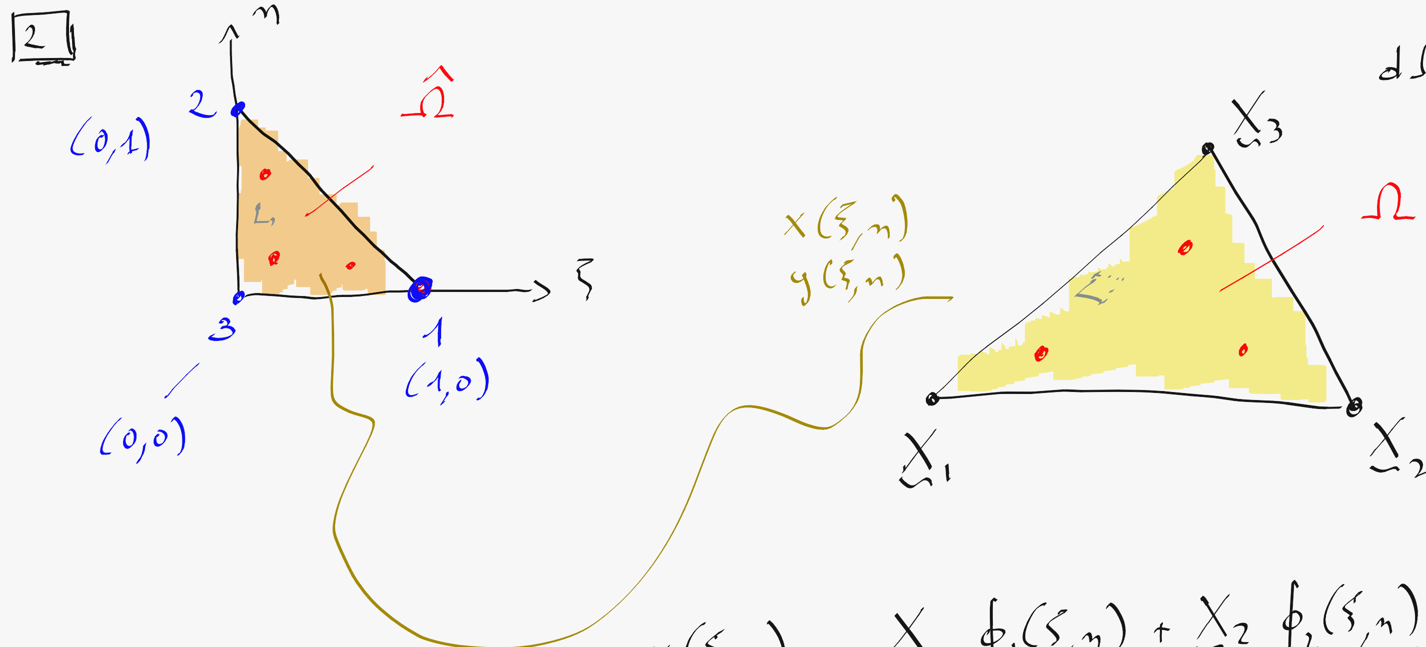
	X_k	Y_k	w_k
1	0.5	0.0	1/6
2	0.5	0.5	1/6
3	0.0	0.5	1/6



	X'_k	Y'_k	w'_k
1	5.0	15.0	25
2	5.0	7.5	25
3	0.0	7.5	25

Car la valeur absolue du jacobien = 150

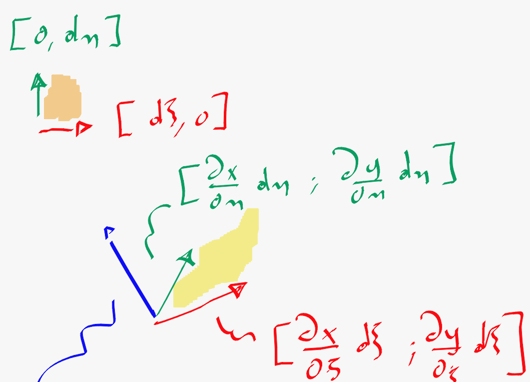
$$d\Omega = J(\xi,\eta)\, d\hat{\Omega}$$



$$\underset{\sim}{x}(\xi,\eta) = \underset{\sim}{X}_1\, \underbrace{\phi_1(\xi,\eta)}_{\xi} + \underset{\sim}{X}_2\, \underbrace{\phi_2(\xi,\eta)}_{\eta} + \underset{\sim}{X}_3\, \underbrace{\phi_3(\xi,\eta)}_{1-\xi-\eta}$$

$$\phi_1(1,0) = 1 \qquad \phi_3(1,0) = 0$$
$$\phi_1(0,0) = 0 \qquad \phi_3(0,0) = 1$$
$$\phi_1(0,1) = 0 \qquad \phi_3(0,1) = 0$$

$$d\hat{\Omega} = d\xi\, d\eta$$



$$d\xi\, d\eta \underbrace{\left[\frac{\partial x}{\partial \eta}\frac{\partial y}{\partial \xi} - \frac{\partial y}{\partial \eta}\frac{\partial x}{\partial \xi} \right]}_{J(\xi,\eta)}$$