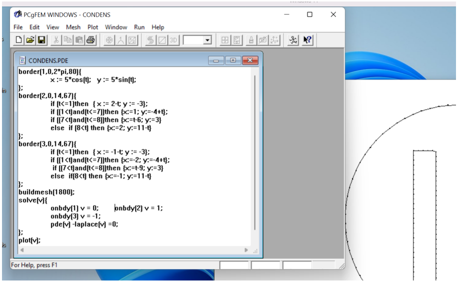
FreeFEM: A Success Story by 40 years of developments

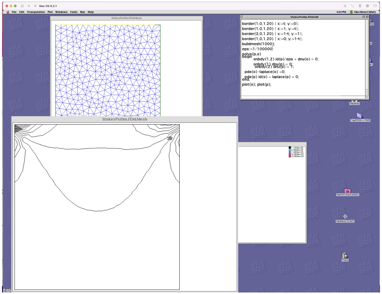
F. Hecht & Olivier Pironneau[1]

[1] LJLL Sorbonne Université

CANUM, Evian June 2022

- From 1985 to 1995: MacFEM, The PDE solver FreeFEM is separated.
  5 hours/day of code development from 9pm to 2am (great fun, actually!).

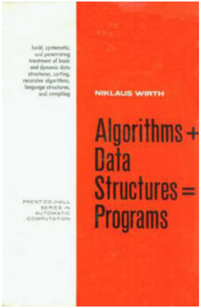


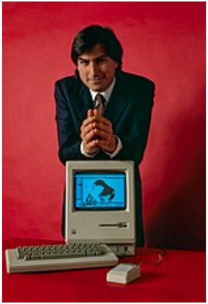


- From 1995 to 2001 MacGFEM, PCGFEM, FreeFEM+
- From 2001 to 2022 FreeFEM++: parallel, links to open source libraries

Evolution of Computing Tools from 1978 to 1986

- In 1978, All computations are in Fortran on mainframes.
  But the Apple-II has a `pascal` compiler, ideal for teaching.
- Niklaus Wirth's book: anyone can write his own `pascal` compiler.
- In 1983 the Macintosh+ is released: 128Mo, boots from a floppy disk!
- Access to the Mac Toolbox in `pascal`
- Very basic `pascal` compiler called the *p-system*. Memory link $\Longrightarrow$ a Bomb

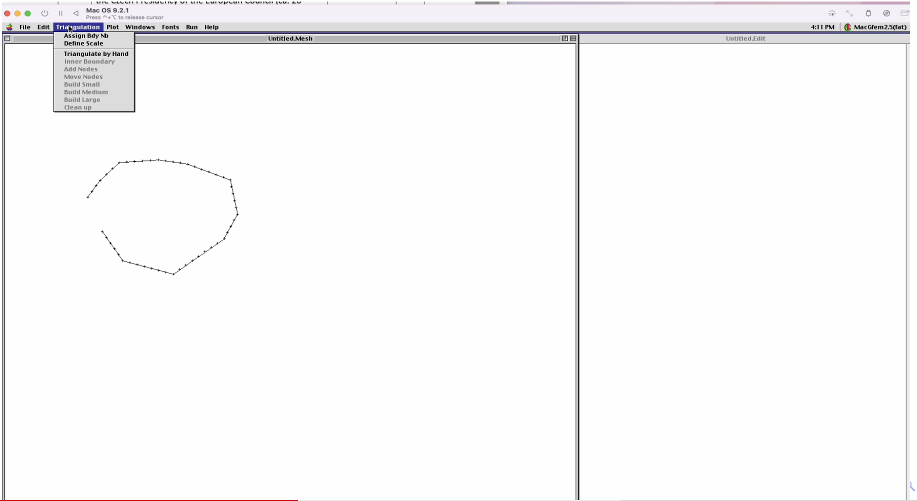



First attempt in 1978

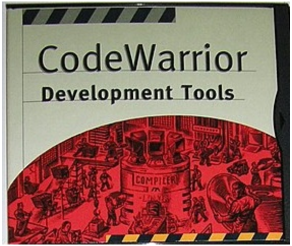
RUN

At last the College-course potatoe can be triangulated by a few clicks.



Development Tools

- Niklaus Wirth's book : anyone can write his own `pascal` compiler.
- In 1986 an efficient `pascal` compiler is released for the Macintosh+: Symantec Think Pascal.
- CodeWarrior is the best C++ compiler released in 1993, both Mac and PC.
- In 1987 D. Bernardi and O. Pironneau teach a course "Outils informatiques pour le calcul scientifique".
- FreeFEM is written to ease the learning curve of the Finite Element Method
- And tach teach how to write a user defined language as in N. Wirth's book.



The First MacFEM/FreeFEM Had Unusual Features

- Delaunay-Voronoi Automatic meshing (non convex case by F. Hecht)
- Graphic User Interface
- High Level Dedicated Language
- Multiphysics, Nonlinear systems

```
border a(t=0,1){ x=t; y=0}; // the unit square
border b(t=0,1){ x=1; y=t};
border c(t=1,0){ x=t; y=1};
border d(t=1,0){ x=0; y=t};
n:=20;  mesh th= buildmesh(a(n)+b(n)+c(n)+d(n));
solve(u,v,p) with B(i){
    pde(u)  u/dt- laplace(u)*nu + dx(p) = convect(u,v,dt,u)/dt;
        on(a,b,d) u =0;
        on(c) u = 1;
    pde(v) v/dt- laplace(v)*nu + dy(p) =  convect(u,v,dt,v)/dt;
        on(a,b,c,d) v=0;
    pde(p) p*0.1*dt - laplace(p)*0.1*dt + dx(u)+dy(v) = 0;
        on(a,b,c,d) dnu(p)=0;
};
```

- FreeFEM was really useful for research too!

Clever Choices and Lacunas

Good

- Open source multi-plateform freeware
- Like a kernel Lego: laplace and convect
- Algorithms are designed by users, e.g. for time dependent PDEs.

Bad

- Poor graphics, but only one small file is machine dependent
- No Riemann solvers or others for hyperbolic PDE
- No CAD for the definition of the domain.
- No compiler to get a C code from a FreeFEM script (FF is interpreted).
- No automatic differentiation
- No link to Matlab.

The Revolution of 1996: Operator Overloading[2]

```
#include <assert.h>
template <class T> class A{  public: T *cc; long size;
       A(long csize = 0)
          { size = csize;
            if (size > 0 ) cc = new T[size]; else cc = 0;
          }
T&     operator [] (long i) const
          {assert ( cc&&(i >= 0) && (i < size) ); return cc[i]; }
int    no( T* t){ return t - cc;}  // return the place in array
      ~A() { delete [] cc;size = 0; }
void   init(long ssize);  // allocates already declared arrays
};
```

Program a Laplace equation with an undefined base type.
$\Longrightarrow$ Get a vector Laplacian in 2D/3D/complex by changing the base type.
The same code can solve the Lamé system and the Stokes system[1].
Adapted mesh refinement (F. Hecht): Use the Hessian of the solution to define a scalar product & norm for the Delaunay-Voronoi algorithm.

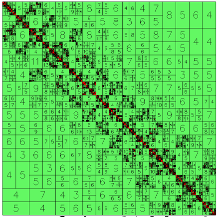
---

[1] **F. Hecht and O. Pironneau**, *Vector Finite Elements and C++*, Advanced Methods for PDE. R. Glowinski's 60^{th} anniversary, Blois. J. Periaux et al eds. 1997, Springer.

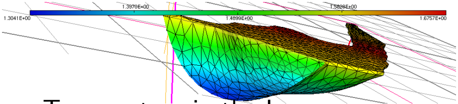
[2] **I. Danaila, F. Hecht and O. Pironneau**, *Simulation Numerique en C++*, Dunod, 2003.

Completly Rewritten in 2001: FreeFEM++

- Use state of the art tools: `Bison`, `FleX`
- Link to Linear Algebra Libraries
- Parallelism: link with `PetSc` etc.
- After 17 years I give up, F. Hecht takes over, what you see of `FreeFEM++` is entirely due to him & collaborators: F. Nataf, P. Jolivet, P.H. Tournier, G. Sadaka, etc.
- Our latest app requiring FF development: Radiative Transfer + Navier-Stokes



Evian ↓



Compress rate of the $\mathcal{H}$-Matrix. Temperature in the Leman, on a sunny day

The (virtual) floor is now to Frédéric Hecht...

Now FreeFEM Canum

O. Pironneau, F. Hecht

LJLL, Sorbonne Université, Paris   projet Alpines, Inria de Paris

with P. Jolivet, P-H. Tournier, F. Nataf, X. Claeys

Evian, 13, 17 juin 2022, Canum

http://www.freefem.org

mailto:frederic.hecht@sorbonne-universite.fr

mailto:olivier.pironneau@gmail.com

Outline

1 Introduction

2 News

3 Academic Examples

Outline

1. Introduction
2. News
3. Academic Examples

FreeFem++ is a software to solve numerically partial differential equations (PDE) in $\mathbb{R}^2$) , $\mathbb{R}^3$) , on Curve or on surface with finite elements methods. We used a domain-specific language (DSL) to set and control the problem. The FreeFEM language allows for a quick specification of linear PDE's, with the variational formulation of a linear steady state problem and the user can write they own script to solve no linear problem and time depend problem. You can solve coupled problem, problem with moving domain, eigenvalue problem, do mesh adaptation , compute error indicator, etc ...

By the way, FreeFEM is build to play with abstract linear, bilinear form on Finite Element Space and interpolation operator.

FreeFem++ is a freeware and this run on Mac, Unix and Window architecture, in parallel with MPI.
To try of cell phone https://www.ljll.math.upmc.fr/lehyaric/ffjs/

Info: FreeFem++ solve a problem with $22\,10^9$ unknowns in 200 s on 12,000 proc (Thank to P. Jolivet).
A Community https://community.freefem.org

- Wide range of finite elements: continuous P1,P2 elements, discontinuous P0, P1, RT0,RT1,BDM1, elements ,Edge element, vectorial element, mini-element, ...
- Automatic interpolation of data from a mesh to an other one (with matrix construction if need), so a finite element function is view as a function of (x, y, z) or as an array.
- Definition of the problem (complex or real value) with the variational form with access to the vectors and the matrix.
- Discontinuous Galerkin formulation (2d , 3d , not on curve and surface).
- LU, Cholesky, Crout, CG, GMRES, UMFPACK, SuperLU, MUMPS , Dissection, PETSc. ... sparse linear solver; eigenvalue and eigenvector computation with ARPACK or SLEPc.
- Online graphics with OpenGL/GLUT/VTK, C++ like syntax.
- Javascript version works straight out of an HTML page, both online or offline (here).

The main characteristics II/II (2D)(3D)

- Analytic description of boundaries, with specification by the user of the intersection of boundaries in 2d.
- Automatic mesh generator, based on the Delaunay-Voronoï algorithm. (2d,3d (tetgen))
- load and save Mesh, solution
- Mesh adaptation based on metric, possibly anisotropic, with optional automatic computation of the metric from the Hessian of a solution. (2d,Surface,3d) (bamg, mmgs, mmg3d).
- Link with other soft: parview, gmsh , vtk, medit, gnuplot
- Dynamic linking to add plugin.
- Full MPI interface
- Nonlinear Optimisation tools: CG, Ipopt, NLOpt, stochastic
- Wide range of examples: Navier-Stokes 3d, elasticity 3d, fluid structure, eigenvalue problem, Schwarz' domain decomposition algorithm, residual error indicator ...

Outline

1. Introduction
2. News
3. Academic Examples

In progress (Near futur or not)

- to get the last develop Version 4.10, do:

```
git clone -b develop https://github.com/FreeFem/FreeFem-sources ff++
```

- Mesh intersection of get conservative formulation in 2d (too hard=> no)
- fast interpolation on Surface and Line Mesh, (done)
- New graphics interface , (in progress, but slow)
- Element on curve and Surface (Ok)
- BEM method a first version (see X. Claeys and P-H Tournier, in very good progress)
- DG in 3d, surface, curve (first version)
- cmake (????)
- rewrite of sparse matrix kernel (Done) ($\Longrightarrow$ new GMRES, new CG, ...), done.
- coupling 3d, surface Finite element and BEM (in progress)
- rewrite of the Finite element kernel for isoparametric FE and to be able mixte surface FE and 3d FE (in futur) in a problem (no template).
- debugger (client-server graphics services) (Good idea but technical)
- more general problem (possible with new matrix, starting)
- OpenMP interface ???
- Quadruple precision floating type (see A. Suzuki)

Outline

1 Introduction

2 News

3 Academic Examples

Let a domain Ω with a partition of $\partial\Omega$ in Γ_2, Γ_e.
Find u a solution in such that:

$$-\Delta u = 1 \text{ in } \Omega, \quad u = 2 \text{ on } \Gamma_2, \quad \frac{\partial u}{\partial \vec{n}} = 0 \text{ on } \Gamma_e \tag{1}$$

Denote $V_g = \{v \in H^1(\Omega) / v_{|\Gamma_2} = g\}$.
The Basic variationnal formulation with is: find $u \in V_2(\Omega)$, such that

$$\int_\Omega \nabla u . \nabla v = \int_\Omega 1v + \int_\Gamma \frac{\partial u}{\partial n} v, \quad \forall v \in V_0(\Omega) \tag{2}$$

The finite element method is just: replace V_g with a finite element space, and the `FreeFem++` code:

Poisson equation in a fish with FreeFem++

The finite element method is just: replace V_g with a finite element space, and the `FreeFem++` script:

```
mesh3 Th("fish-3d.msh");                      //    read a mesh 3d
fespace Vh(Th,P1);                       //    define the P1 EF space

 Vh u,v;                    //   set test and unknown function in Vh.
macro Grad(u) [dx(u),dy(u),dz(u)]                      //   EOM Grad def
solve laplace(u,v,solver=CG) =
   int3d(Th)(  Grad(u)'*Grad(v)   )
   - int3d(Th) ( 1*v)
   + on(2,u=2);                                          //    int on γ2
plot(u,fill=1,wait=1,value=0,wait=1);
```

Run:fish.edp   Run:fish3d.edp

Let S a surface (here a torus), we want to compute the eigen value of Laplace Beltrami operator, plot the eigen value as the deformation of the surface on the direction of the normal.

The variational form is: find (u_λ, λ) such than

$$\int_S \nabla_S u . \nabla_S v = \lambda \int_S uv \tag{3}$$

where ∇_S is the tangential gradient and at discret level the basic function are constant in normal direction, so $\nabla_S \equiv \nabla$.
remark: Now in FreeFEM the word `Ns` is a global variable to get the normal at surface mesh (a constant by triangle).

To plot the solution, I propose to deform the surface with vector $u_\lambda N_h$ where N_h is a P_1 approximation of the the unit normal of S (L^2 projection of the normal) .

Run:LapEigenBeltrami.edp

Example of adaptation process in 3d with mmg3

Let a domain $\Omega =]0,1[^3 \backslash [\frac{1}{2},1[^3$ The border of $\partial\Omega$ is split in 2 part

- Γ_2, if $x=1$, $y=1$, or $z=1$
- Γ_1, else.

Find u a solution in such that:

$$\begin{aligned} -\Delta u &= 1 && \text{in } \Omega, \\ \frac{\partial u}{\partial \vec{n}} &= 0 && \text{on } \Gamma_2, \\ u &= 0 && \text{on } \Gamma_1. \end{aligned}$$

Thank to mmg v5 tools to do 3d mesh adaptation see `http://www.mmgtools.org` .

without mmg (isotrope): Run:Laplace-Adapt-3d.edp

with mmg (anisotrope): Run:Laplace-Adapt-aniso-3d.edp

First get the data, the border and the deep

1. Get the data file of the lac border and the iso deep line form the Suisse
2. Clean the data file (too much info)
3. Build a 2d mesh form all iso lines build a fonction form the 2d mesh.
4. Use buildlayer to build the 3d mesh .

Run:build-leman-2d.edp Run:build-leman-3d.edp

Freefem++ v4 is

- very good tool to solve non standard PDE in 2D/3D and of surface
- to try new domain decomposition domain algorithm

The the futur we try to do:

- Build more graphic with VTK, paraview , ... (P. Jolivet,)
- 3d anisotrope mesh adaptation (see new version mmg3d software at page 13)
- automate the parallel tool (in progress)
- generalized the problem definition (X Claeys, P-H Tournier, J Morice)
- finite volume (See G. Sadaka).

Thank for you attention.