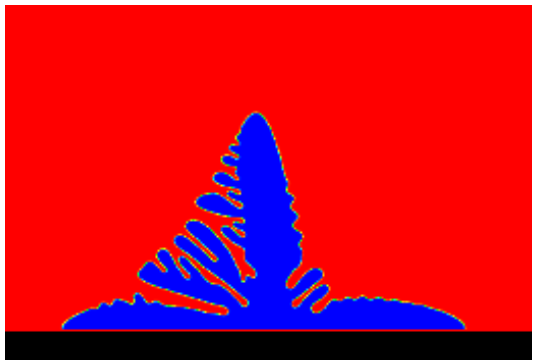


femLego

Numerical Simulation by Symbolic Computation

A set of Maple procedures and fortran subroutines that can be used to build complete simulation codes for partial differential equations, with the entire problem definition done in Maple.

www.mech.kth.se/~gustava

Amberg, G., Tönhardt, R, and Winkler, C. (1999) 'Finite Element simulations using symbolic computing', *Mathematics and Computers in Simulation*, **49**, pp. 149-165

Outline

- Some applications
- The Finite Element Method
- A simple example

- Try it yourself!

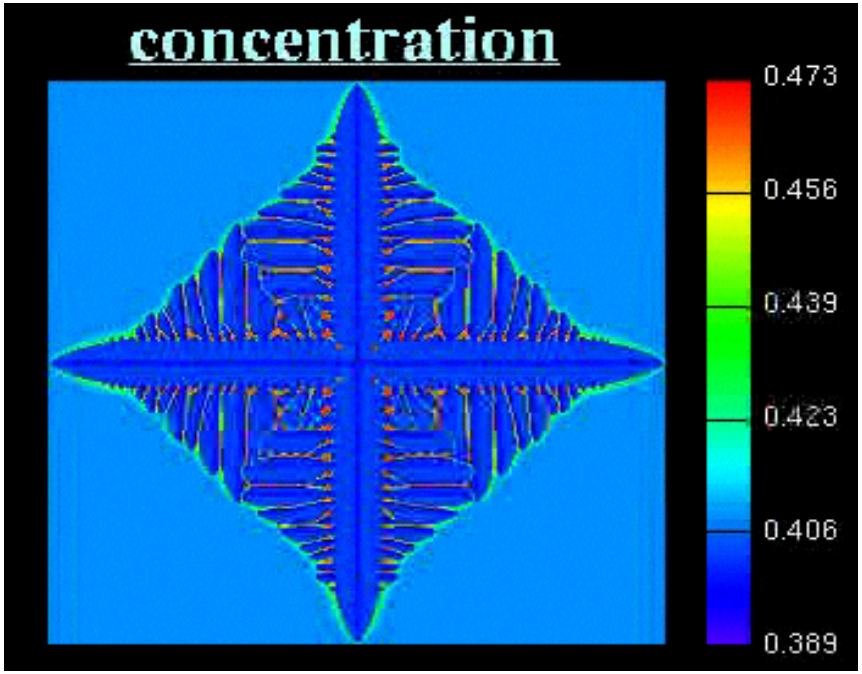
Phase field model

$$\tau w^2 \frac{\partial \phi}{\partial t} = \left[\phi - \lambda\Delta\ \theta \cdot \left(1-\phi^2\right)\right] \cdot \left(1-\phi^2\right) +$$

$$\nabla \cdot (w^2 \nabla \phi) - \left(\frac{\partial}{\partial x}\left(ww' \frac{\partial \phi}{\partial y} \right) - \frac{\partial}{\partial y}\left(ww' \frac{\partial \phi}{\partial x} \right) \right)$$

$$\frac{\partial \theta}{\partial t} + \mathbf{u} \cdot \nabla \theta = \nabla^2 \theta + \frac{1}{2\Delta} \frac{\partial \phi}{\partial t}$$

$$w = 1 + \varepsilon \cos\left(4\left(\beta - \beta_0\right)\right),\ \beta = \arctan\left(\frac{\partial \phi}{\partial x} / \frac{\partial \phi}{\partial y} \right)$$



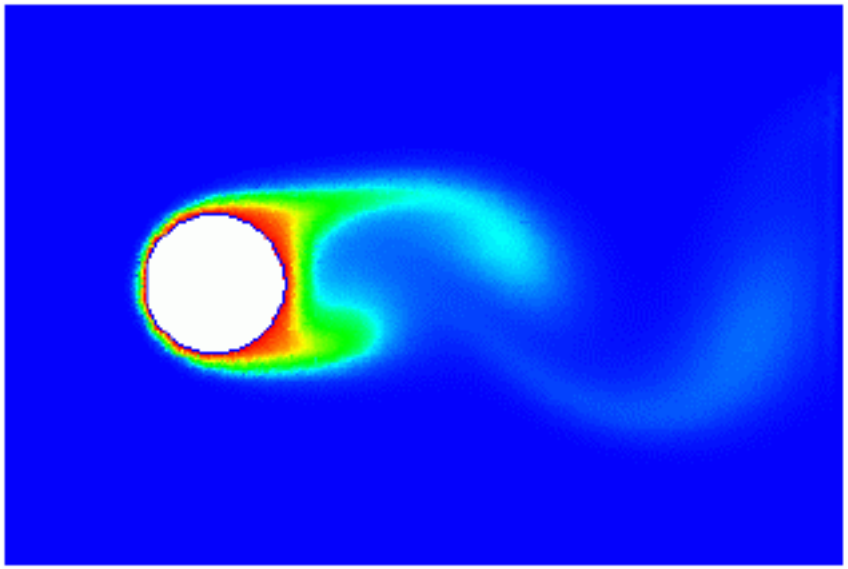
(Irina Loginova)

Fluid mechanics

Fluid flow past a cylinder

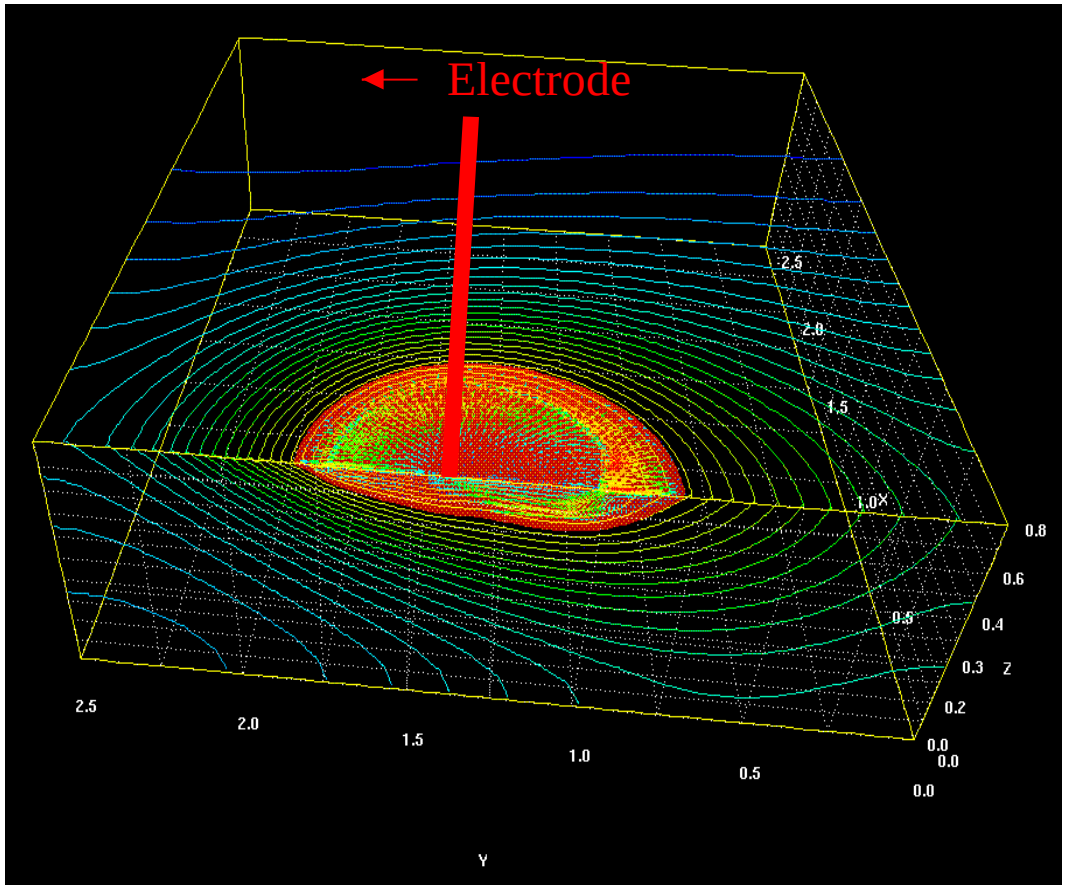
$$\rho\left(\frac{\partial u_i}{\partial t} + u_k \frac{\partial u_i}{\partial x_k}\right) = -\frac{\partial p}{\partial x_i} + \mu \nabla^2 u_i$$

$$\frac{\partial u_j}{\partial x_j} = 0$$



Welding

Melt flow
Electromagnetic forces
Phase change
Heat and mass transfer
+ more



(Christian Winkler, Minh Do-Quang)

Need to do such simulations

Systems of nonlinear PDEs:
time dependent
(1), 2 or 3 spatial dimensions

Our priorities are, in this order:

- Immediate access to the model
- Access to the numerics
- Numerical efficiency, speed

Finite Elements

The strategy is:

- Finite Elements
- Unstructured grids
- Finite differences in time.

One reason is that finite elements on unstructured grids can be formulated as one simple algorithm, regardless of what element is used, number of spatial dimensions, etc.

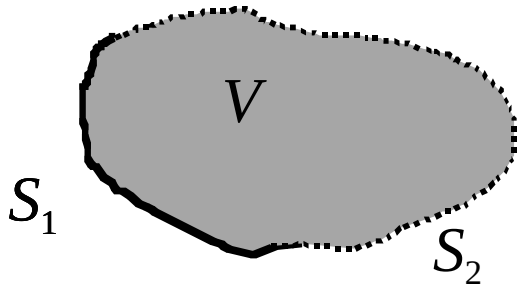
Unstructured meshes allow adaptivity.

Simple example

$$\frac{\partial v}{\partial t} = \nabla^2 v$$

$$v = f \quad \text{on } S_1$$

$$\frac{\partial v}{\partial n} = Q \quad \text{on } S_2$$



To solve this by Finite Elements:

$$\int_V \phi \frac{(v^{n+1} - v^n)}{dt} dV = -\int_V (\nabla\phi \cdot \nabla v) dV + \oint_{S_2} \phi Q dS$$

$$v = f \quad \text{on } S_1$$

- time derivative -> finite difference.
- Project onto a test function
- Integrate partially

To solve this by Finite Elements:

$$\int_V \phi \frac{(v^{n+1} - v^n)}{dt} dV = -\int_V (\nabla\phi \cdot \nabla v) dV + \oint_{S_2} \phi Q dS$$

$$v = f \quad \text{on } S_1$$

- time derivative -> finite difference.
- Project onto a test function
- Integrate partially

$$v = \sum_{j=1..N} v_j \phi_j(x)$$

$$\sum_{j=1..N} M_{ij} \frac{(v_j^{n+1} - v_j^n)}{dt} = \sum_{j=1..N} A_{ij} v_j^{n+1} + b_i$$

- Express the unknown in the *N* base functions
- Evaluate the integrals for all N independent test functions -> System for node values

To solve this by Finite Elements:

$$\int_V \phi \frac{(v^{n+1} - v^n)}{dt} dV = -\int_V (\nabla\phi \cdot \nabla v) dV + \oint_{S_2} \phi Q dS$$

$$v = f \quad \text{on } S_1$$

Supplied by user

- time derivative -> finite difference.
- Project onto a test function
- Integrate partially

Done symbo lically

$$v = \sum_{j=1..N} v_j \phi_j(x)$$

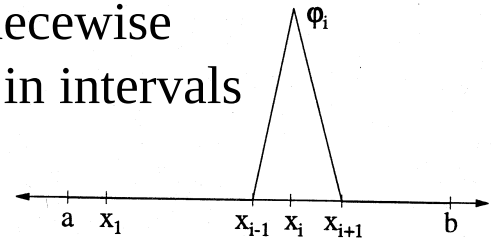
$$\sum_{j=1..N} M_{ij} \frac{(v_j^{n+1} - v_j^n)}{dt} = \sum_{j=1..N} A_{ij} v_j^{n+1} + b_i$$

- Express the unknown in the N base functions
- Evaluate the integrals for all N independent test functions -> System for node values

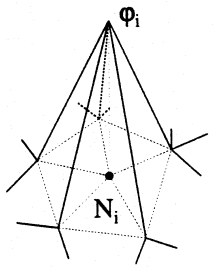
The Finite Elements

The most obvious examples of finite elements, i.e. base functions , are piecewise linear functions:

1D: piecewise linear in intervals



2D: piecewise linear in triangles

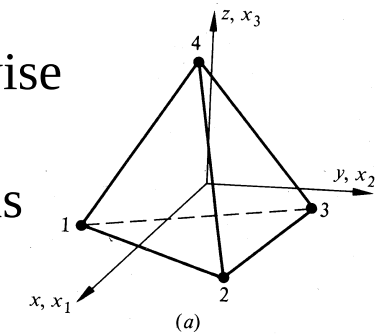


An element is defined by a separate file of Maple procedures.

The user chooses element by supplying the corresponding filename as input to the code generation.

Modular, easy to change.

3D: piecewise linear in tetrahedrons

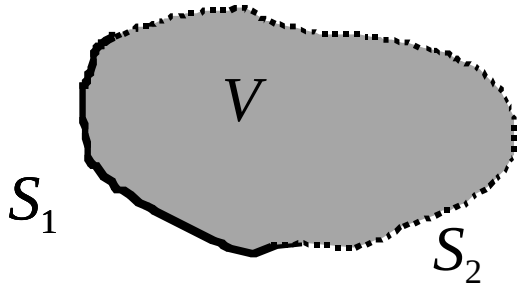


Simple example

$$\frac{\partial v}{\partial t} = \nabla^2 v$$

$$v = f \quad \text{on } S_1$$

$$\frac{\partial v}{\partial n} = Q \quad \text{on } S_2$$



FEM form:

$$\int_V \phi \frac{(v^{n+1} - v^n)}{dt} dV = -\int_V (\nabla\phi \cdot \nabla v) dV + \oint_{S_2} \phi Q dS$$

$$v = f \quad \text{on } S_1$$

In Maple:

slask.mws - [Server 1]

```
> read(femLego);
```

Specify PDEs

```
> # Diffusion equation, with gradient BC;
> veq:=
   ElementInt(phi(x,y)*(v(x,y) - vo(x,y))/dt)=
   ElementInt( -nab(phi(x,y))[i] &t nab(v(x,y))[i])
         + BoundaryInt(phi(x,y)*Qflux);
```

$$veq := \mathrm{ElementInt}\left(\frac{\phi(x,y)\,(v(x,y)-vo(x,y))}{dt}\right)=$$

$$\mathrm{ElementInt}\left(-\left(\frac{\partial}{\partial x}\phi(x,y)\right)\left(\frac{\partial}{\partial x}v(x,y)\right)-\left(\frac{\partial}{\partial y}\phi(x,y)\right)\left(\frac{\partial}{\partial y}v(x,y)\right)\right)+\mathrm{BoundaryInt}(\phi(x,y)\,Qflux)$$

Sections to put in Dirichlet BCs, specify pre/post processing, choose linear algebra solvers, etc.

Create the core of the solver

```
> # create residual computations etc:
> mkFem([veq],[v(x,y)],[vo(x,y)],[],phi(x,y), '2DP1_gsq_bsf');
```

mkresi

.

.

.

[1, 3, 2]

Other tools: Tensor and vector notation.

&t implements the Einstein summation convention:

```
> read(femLego);
>
> a[i] &t b[i];
```

$$a_1 b_1 + a_2 b_2 + a_3 b_3$$

The nabla operator:

```
>
  u[j](x,y,z) &t nab(u[i](x,y,z))[j];
```

$$u_1(x,y,z)\left(\frac{\partial}{\partial x}u_i(x,y,z)\right) + u_2(x,y,z)\left(\frac{\partial}{\partial y}u_i(x,y,z)\right) + u_3(x,y,z)\left(\frac{\partial}{\partial z}u_i(x,y,z)\right)$$

Partial integration, taking the variation, etc...

Other commands needed to build a complete solver

* Specify Dirichlet boundary conditions:

```
mkDirBC( [u(x,y) = sin(x) ], ... );
```

* Initial conditions:

```
mkICcopy( [u(x,y) = x^2 + y^2], ... );
```

* Specify how to read input (mesh etc):

```
getFemLabInput( [Reynold,Prandtl, .. ] );
```

* Specify how to save output:

```
AVSplotP1([u(x,y),v(x,y)]);
```

* Specify linear algebra solvers:

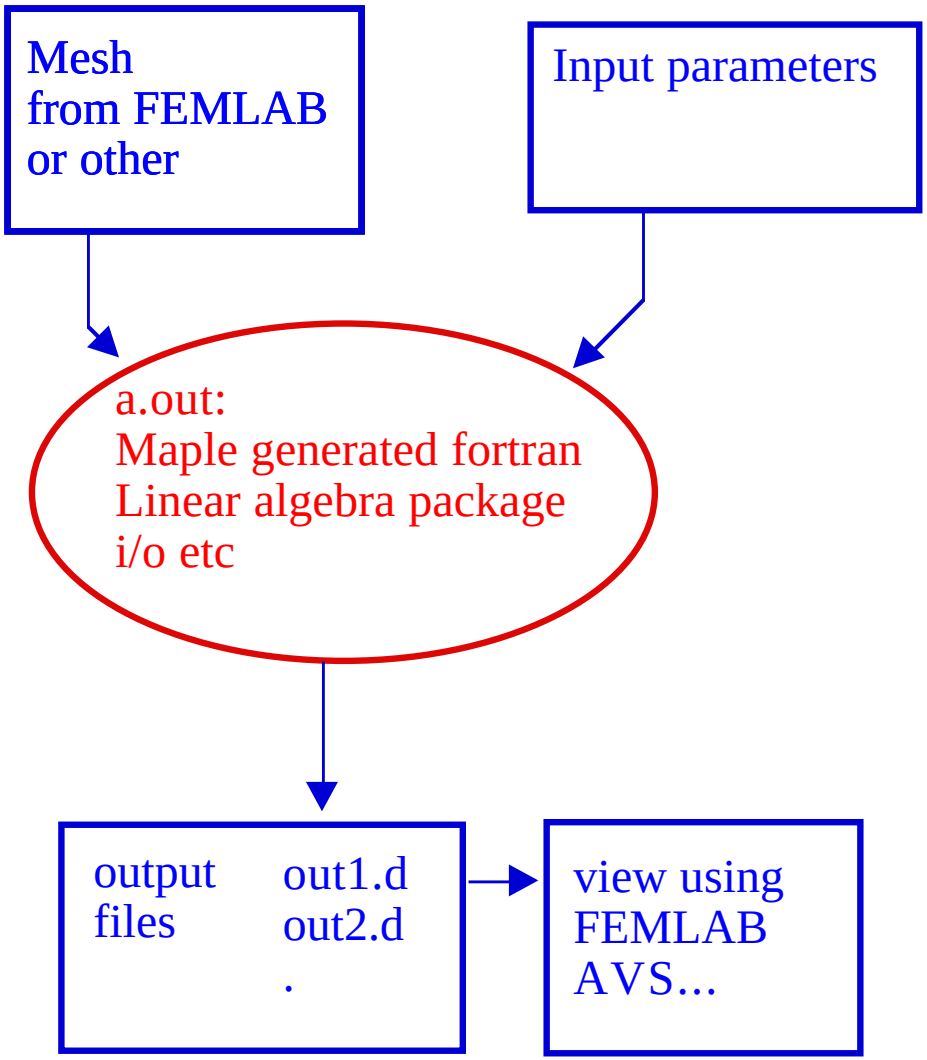
```
mkSolve([iccg,gmres],[eq1,eq2]);
```

* Create the core of the program:

```
mkFem([eq1,eq2],[u(x,y),v(x,y)], ... );
```

When all problem dependent parts have been generated in Maple, the code is compiled and run as usual.

Rely on other applications for pre- and postprocessing



Get started:

Open www.mech.kth.se/~gustava/femLego

- Notice ’commands’ and ’Examples’

Get a copy

Step through ’little_example’

- See how contents of ./source changes
  compile and run

Try one of the other examples, modify the examples or try some other problem

To get a copy:

Download and unpack files (see webpage)

Copy init file for Maple:

```
cp femLego_dist/.mapleinit ~
```

Find or install linear algebra lib, set environment var.

```
setenv SLAP_LIB $HOME/lib
```

(or put in ~/.cshrc)

Go to the directory you want to work in, start Maple:

```
cd little_example
xmaple example.mws &
```

Suggested modifications

- Add distributed heat source to little_example or 3D_example.
- Add a nonlinear diffusion coefficient to little_example or 3D_example.
- Add adaptivity to little_example.
- Put in your favorite PDE.