OpenFEM

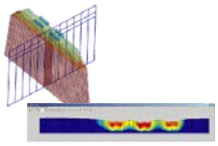
An open source finite element toolbox

SDTools : Etienne Balmes, Jean Michel Leclere
INRIA : D. Chapelle, C. Delforge, A. Hassim, M. Vidrascu, …

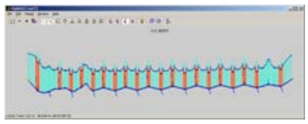
Target

OpenFEM is meant to let you use it's components to build your application

- General purpose FEM solver
- Multi-physic support
- Toolbox flexibility and state of the art performance



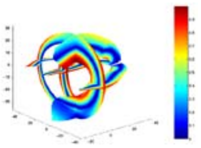
Dynavoie
SDTools/SNCF-DIR



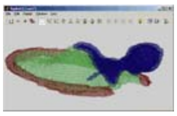
Oscar : catenary-pantograph interaction
SDTools/SNCF-DIR

3D model
explicit integration,
1 million of time step in 2 hours

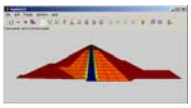
Applications



Heart simulation
INRIA, Zapadoceska Univerzita

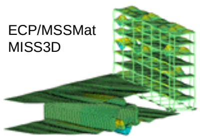


Internal ear modeling
University Hospital Zuerich



ECP/MSSMat Gefdyn

ECP/MSSMat
MISS3D



OpenFEM history

Start in 2001 from

- *Structural dynamics toolbox* .m file elements limited library
- *MODULEF* large library but no longer a convenient prototyping environment

Phase I -> OpenFEM 1.0 & 2.0

- Port of MODULEF elements (2D and 3D volumes, MITC4)
- Translation to SCILAB (Claire Delforge)

Phase II (current)

- Efficient non-linear operation (generic compiled elements, geometric non-linear mechanics, …)

Design criteria

- Be a toolbox (easy to develop, debug, optimization only should take time)
- Optimize ability to be extended by users
- Performance identical to good fully compiled code
- Solve very general multi-physics FE problems
- Be suitable for application deployment

A Matlab/Scilab Toolbox, why ?

- Development is easier (interactive mode, debugger)
- Students (non experts) understand it and can rapidly prototype variations from standard code
- Performance is not worse (can be better than poor compiled code)
- One can easily link into most external libraries

Easy user extensions

- Object oriented concepts (user object provides its methods)
- But non typed data structures (avoid need to declare inheritance properties)

Example user element

- Element name of .m file (beam1.m)
- Must provide basic methods (node, DOF, face, parent, …)
- Self provide calling format. Eg : beam1('call')
  [k1,m1]=beam1(nodeE, elt(cEGI(jElt),:), pointers(:,jElt), integ, constit, elmap, node);

Other self extensions : property functions

OpenFEM architecture

Preprocessing
- Mesh manipulations
- Structured meshing
- Property/boundary condition setting

Import
- Modulef, GMSH, GID
- Nastran, IDEAS, ANSYS, PERMAS, SAMCEF, MISS, GEFDYN

FEM core
- Shape function utilities
- Element functions
- Matrix and load assembly
- Factored matrix object (dynamic selection of sparse library)
- Linear static and time response (linear and non linear)
- Real eigenvalues
- Optimized solvers for large problems, superelements, and system dynamics, model reduction and optimization
- Drive other software (NASTRAN, MISS)

Postprocessing
- Stress computations
- Signal processing
- 3D visualization (major extension , optimized, object based)

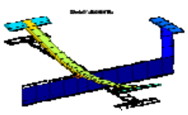
Export
- MEDIT
- Nastran, IDEAS, SAMCEF
- Ensight, MISS3D, Gefdyn

OpenFEM, SDTools, MSSMat

Meshing 1 : example

```
femesh

FEelt=[];
FEnode = [1 0 0 0  0 0 0;2 0 0 0  0 0 .15;
        3 0 0 0 0.4 1.0 .176;4 0 0 0 0.4 0.9 0.176];
% fuselage
femesh('objectbeamline 1 2');
femesh('extrude 0  1.0 0.0 0.0', …
  [linspace(0,.55,5) linspace(.65,1.4,6) 1.5]);
femesh('addsel;');

% vertical tail
femesh('objectbeamline',femesh('findnode z==.15 & x>=1.4'));
femesh(';extrude 3 0 0 .1;addsel;');
% vertical horizontal tail
femesh('objectbeamline',femesh('findnode z==.45'));
femesh('extrude 0  0.0 0.2 0.0',[-1 -.5 0 .5 1]);
femesh('addsel;');

% right drum
femesh(';objectbeamline 3 4;extrude 1 .4 0 0');
femesh('divide',[0 2/40 15/40 25/40 1],[0 .7 1]);
femesh('addsel;');

% left drum
femesh(';symsel 1 0 1 0;addsel;');
```

- Structured meshing
- Mapped divisions
- Objects (beam, circle, tube, …)



```
 Node: [144x7 double]
  Elt: [100x9 double]
   pl: [2x6 double]
   il: [4x6 double]
  bas: []
Stack: {}
```

Meshing 2 : femesh/feutil

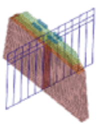
- Add FEeli FEelj, AddSel
- AddNode [,New] [, From i]
- AddTest [,NodeShift,Merge]
- Divide div1 div2 div3
- DivideInGroups
- DivideGroup i ElementSelectors
- EltId
- Extrude nRep tx ty tz
- FindDof ElementSelectors
- GetDof
- Find [Elt,El0] ElementSelectors
- FindNode Selectors
- GetEdge[Line,Patch]
- GetElemF
- GetLine
- GetNode Selectors
- GetNormal[Elt,Node][,Map]
- GetPatch
- Info [,FEeli, Nodei]
- Join [,el0] [group i, EName]
- model [,0]
- Matid,ProId,MPID
- ObjectBeamLine i, ObjectMass i
- ObjectHoleInPlate
- Object[Quad,Beam,Hexa] MatId ProId
- Object[Circle,Cylinder,Disk]
- Optim [Model, NodeNum, EltCheck]
- Orient, Orient i [, n nx ny nz] [,-neg]
- Plot [Elt, El0]
- Quad2Tria, quad42quadb, etc.
- RefineBeam
- Remove[Elt,El0] ElementSelectors
- Renumber
- RepeatSel nITE tx ty tz
- Rev nDiv OrigID Ang nx ny nz
- RotateSel OrigID Ang nx ny nz
- Sel [Elt,El0] ElementSelectors
- SelGroup i, SelNode i
- SetGroup [i,name] [Mat j, Pro k, EGID e, Name s]
- StringDOF
- SymSel OrigID nx ny nz
- TransSel tx ty tz
- UnJoin Gp1 Gp2

Generation, Selection, …

Meshing 3: unstructured

Rationale : meshing is a serious business that needs to be integrated in a CAD environment. OpenFEM is a computing environment.

- IMPORT (MODULEF, GMSH, GID, NASTRAN, ANSYS, SAMCEF, PERMAS, IDEAS)
- Run meshing software : GMSH Driver, MODULEF
- 2D quad meshing, 2D Delaunay

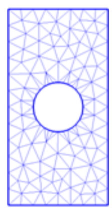


OpenFEM, SDTools

Meshing 4: fe_gmsh

```
FEnode = [1 0 0 0  0 0 0; 2 0 0 0  1 0 0; 3 0 0 0  0 2 0];
femesh('objectholeinplate 1 2 3 .5 .5 3 4 4');
FEelt=FEel0;femesh('selelt seledge');model=femesh('model0');
model.Node=feutil('getnode groupall',model);

model=fe_gmsh('addline',model,'groupall');
mo1=fe_gmsh('write temp.msh -lc .3 -run -2 -v 0',model);
feplot(mo1); delete('temp.msh')
```

Default element size
GMSH options



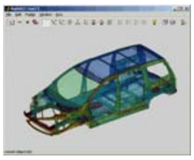
- Good functionality for 2D and 3D
- Limited handling of complex surfaces

Meshing 5: selection

Recursive node and element selections

GID ~=i
Group ~=i
Groupa i
InElt{sel}
NodeId > i
NotIn{sel}
Plane == i nx ny nz
rad <=r x y z
Setname name
x >a
x y z

EltId i
EltInd i
EltName ~=s
EGID == i
Facing > cos x y z
Group i
InNode i
MatId i
ProId i
SelEdge type
SelFace type
WithNode i
WithoutNode i



Element library

m-file functions
- 3D lines/points : Bar, Beam, Pre-stressed beam, Spring, bush, viscoelastic spring, mass
- Shells 3/4 nodes
- Multilayer shell element

mex from MODULEF
- 2D, plane stress/strain, axi, linear, quadratic
- 3D, linear and quadratic, geometric-linear, orthotropy
- Shells (MITC4)

mex (generic compiled elements)
- 2D, plane stress/strain, linear, quadratic
- 3D, linear and quadratic, geometric non-linear mechanics, full anisotropy, mechanical or thermal pre-stress
- Acoustic fluids
- *INRIA : hyperelasticity, follower pressure*
- SDTools : piezo volumes and shells with composite support, poroelasticity

Recent, in development

The OpenFEM specification is designed for multiphysics applications

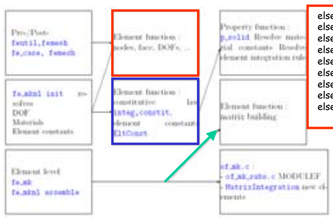
99 DOF/node
999 internal DOF/element

Shape function utilities (integrules)

Supported topologies are
- bar1 (1D linear)
- beam1 (1D cubic)
- quad4 (2D bi-linear), quadb (2d quadratic)
- tria3 (2D affine), tria6 (2D quadratic)
- tetra4, tetra10
- penta6, penta15
- hexa8, hexa20, hexa27

```
» integrules('hexa8',3)

         N: [27x8 double]
        Nr: [27x8 double]
        Ns: [27x8 double]
        Nt: [27x8 double]
        Nw: 27
       NDN: [8x108 double]
 NDNLabels: {'' ',x' ',y' ',z'
      jdet: [27x1 double]
         w: [27x4 double]
     Nnode: 8
        xi: [8x3 double]
      type: 'hexa8'
```

User elements



```
elseif  comstr(Cam,'node');  out = [1 2];
elseif  comstr(Cam,'prop');  out = [3 4 5];
elseif  comstr(Cam,'dof');   out=[1.01 1.02 1.03 2.01 2.02 2.03]';
elseif  comstr(Cam,'line');   out = [1 2];
elseif  comstr(Cam,'face');   out =[];
elseif  comstr(Cam,'sci_face'); out = [1 2 2];
elseif  comstr(Cam,'edge');   out =[1 2];
elseif  comstr(Cam,'patch');  out = [1 2];
elseif  comstr(Cam,'parent');   out = 'beam1';
```

[k1,m1]=beam1(nodeE, elt(cEGI(jElt),:), pointers(:,jElt), integ, constit, elmap, node);

```
[ID,pl,il]=deal(varargin);
pe=pe(find(pe(:,1)==ID(1),3:end); % material properties
ie=ie(find(ie(:,1)==ID(2),3:end);

%          E*A nu       eta  rho*A        A lump
constit = [pe(1)*ie(4)  0    pe(3)*ie(4) ie(4) ie(7)];
integ=ID;%matid proid
Elmap=[];

out=constit(:); out1=integ(:); out2=ElMap;
```

Generic compiled elements

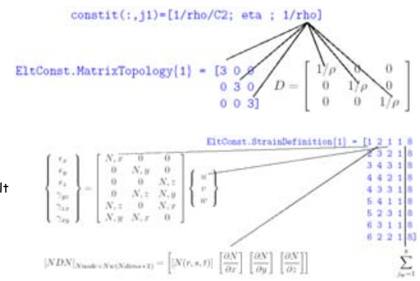
Objective ease implementation of
- arbitrary multi-physic
- linear element families
- Good compiled speed
- provisions for non linear extensions

Assumptions
- Strain ε=[B]{q} linear function of N and ∇N
- Element matrix quadratic function of strain

$$k^{(e)} = \sum_{ji,jj} \sum_{jw} \left[\{B_{ji}\} D_{ji\ jk}(w(jw)) \{B_{jj}\}^T \right] J(w(jw)) W((jw))$$

Generic compiled elements

During assembly init define
- D ↔ constit
- ε ↔ EltConst.NDN
- K_e ↔ D,e (EltConst. MatrixIntegrationRule built in integrules MatrixRule)



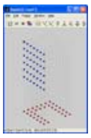
EltConst=p_solid('constsolid','hexa8',[],[])
p_solid('constsolid','hexa8',[],[])
p_solid('constfluid','hexa8',[],[])

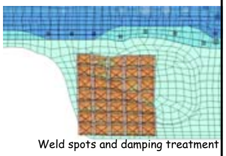
Boundary conditions

Cases define :

boundary conditions, point and distributed loads, physical parameters, ...

```
data=struct('sel','x==-.5', ...
          'eltsel','withnode {z>1.25}', ...
          'def',1,'DOF',.19);
model = fe_case(femesh('testubeam'),...
     'FSurf','Pressure load',data, ...
     'FixDof','Fixed boundary condition','x==0');
```





Weld spots and damping treatment

Supported boundary conditions

- KeepDOF, FixDOF
- Rigid
- MPC, Un=0

Handling by elimination : solve

$$[Ms^2 + Cs + K]\{q(s)\} = [b]\{u(s)\}$$
$$\{y(s)\} = [c]\{q(s)\}$$
$$[c_{int}]\{q(s)\} = 0$$

$$\left[T^T M T s^2 + T^T C T s + T^T K T\right]\{q_R(s)\} = \left[T^T b\right]\{u(s)\}$$
$$\{y(s)\} = [cT]\{q_R(s)\}$$

$$\mathrm{range}([T]_{N\times(N-NC)}) = \ker([c_{int}]_{NS\times N})$$

ofact : gateway to sparse libraries

Kq=F is central to most FEM problems. Optimal is case/machine dependent. ofact object allows library independent code.

- Method : dynamic selection of method (OpenFEM, SDTools)

```
      lu : MATLAB sparse LU solver
    chol : MATLAB sparse Cholesky solver
 pardiso : PARDISO sparse solver
*umfpack : UMFPACK solver (NOT AVAILABLE ON THIS MACHINE)
->  spfmex : SDT sparse LDLt solver
  mtaucs : TAUCS sparse solver
 sp_util : SDT skyline solver
 *psldlt : SGI sparse solver (NOT AVAILABLE ON THIS MACHINE)
```

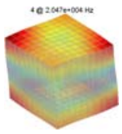
- Symfact : symbolic factorization (renumbering, allocation)
- Fact : numeric factorization (possibly multiple for single symfact)
- Solve : forward backward solve (possibly multiple for single fact)
- Clear : free memory

- Not tried : MUMPS, BCS-Lib, …

ofact : performance test

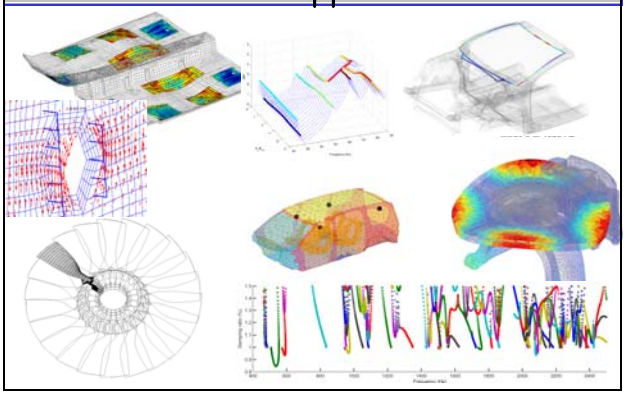
10x10x100 elt 36 663 DOF	10x20x100 elt 69 993 DOF	10x40x100 elt 136 653 DOF	
83 (0.8)	363 (2.6)	1706 (5.8)	SPOOLES, PIII 1 Ghz, Linux
10 (0.2)	90 (2.3)	262 (6.0)	TAUCS snll + metis, PIII 1GHz Linux
	39 (2.6)		SPOOLES, AMD 64 4000+ Linux
28 (0.17)	99 (0.4)		SPOOLES, Xeon 2.6 GHz, Windows
6.8 (0.48)	16 (1.1)		MKL-Pardiso, Xeon 2.6 GHz, Windows
32 (0.64)			CHOL Matlab 7.1 (R13SP3) Xeon 2.6 GHz, Windows
56 (0.69)			LU Matlab 7.1 (R13SP3) Xeon 2.6 GHz, Windows

Fact (solve) CPU seconds



- *All libraries can be accessible (OpenFEM, SDTools), best is application/machine dependent.*
- *Memory usage and fragmentation is another issue that may drive library selection*

SDTools applications



General info

- OpenFEM 3.0 (cvs) Matlab (6.1 and higher) www.sdtools.com/openfem
- OpenFEM 2.0 Matlab & Scilab (3.0) www.openfem.net
- "GNU Lesser Public License" (LGPL)
- Supported on : Windows, Linux 32 & 64, Sun, MacOS X
- Also works on SGI, HP, IBM
- Deployable with MATLAB Compiler

Current activities

- User extendability for distributed loads and non-linear constitutive laws (hyperelasticity)
- Follower pressure and inertial load, thermal, gyroscopic, multilayer shell
- Improve stress processing

SDTools activities that impact OpenFEM

- Composite+piezo shell, piezo volumes
- Advanced constraints (weld, non conform mesh, …)

Current needs

- People to attend various issues
  - Keep SCILAB version up to date
  - Keep OpenFEM/feplot alive and/or MEDIT interface
  - Systematic testing (OpenFEM alone, manual conformity, element validations, ...)
- Constructive feedback (alpha testers)
  - Thermal and thermoelastic elements, …