PyFEM 1.0

User manual

Joris J.C. Remmers, Clemens V. Verhoosel and René de Borst

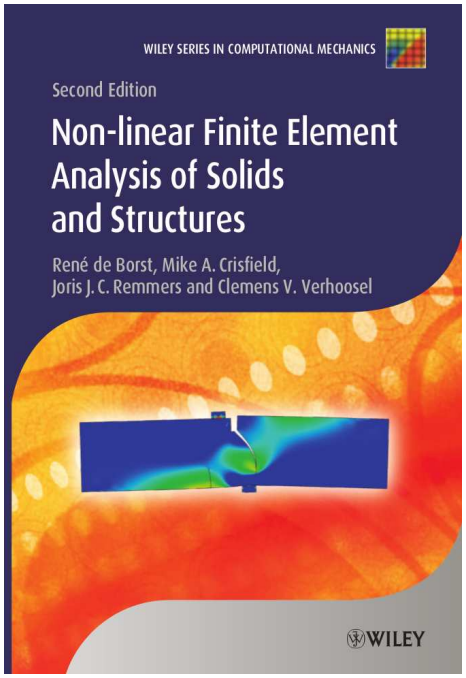
August 29, 2012

Contents

1 About the code

This is the user manual for PyFEM version 1.0. This `python`-based finite element code accompanies the book:

'Non-Linear Finite Element Analysis of Solids and Structures' by R. de Borst, M.A. Crisfield, J.J.C. Remmers and C.V. Verhoosel John Wiley and Sons, 2012, ISBN 978-0470666449



The code is open source and intended for educational and scientific purposes only. If you use PyFEM in your research, the developers would be grateful if you could cite the book in your work. Comments and suggestions are welcome at:

`PyFEM-support@tue.nl`

Disclaimer

The authors reserve all rights but do not guarantee that the code is free from errors. Furthermore, the authors shall not be liable in any event caused by the use of the program.

2 Installation

The code can be downloaded from the website that accompanies the book.

```
http://www.wiley.com/go/deborst
```

On this website, both the current version 1.0 as well as all previous major releases of the code can be found. The code is packed as a `zip` file and can be unzipped in a directory of choice.

This version of the PyFEM is written to work properly in combination with `python` version 2.7. In addition, the code uses the modules `numpy`, `scipy` and `matplotlib`. Installation guidelines are given for various operating systems.

2.1 Windows OS (Windows XP and Windows 7)

1. Since precompiled versions of `numpy`, `scipy` and `matplotlib` are available in 32-bit versions only, it is advised to install the 32-bit version of `python`. This code is available at:

   ```
   http://www.python.org/getit
   ```

   It is recommended to install the latest 32-bit version, which is 2.7.3.

2. Download and install `numpy`. This module is available at:

   ```
   http://sourceforge.net/projects/numpy/files/NumPy
   ```

   It is recommended to install the latest 32-bit version, which is 1.6.2.

3. Download and install `scipy`. This module is available at:

   ```
   http://sourceforge.net/projects/scipy/files/scipy/
   ```

   It is recommended to install the latest 32-bit version, which is 0.11.01b.

4. Download and install `matplotlib`. This module is available at:

   ```
   http://sourceforge.net/projects/matplotlib/files/matplotlib/
   ```

   It is recommended to install the latest 32-bit version, which is version 1.1.0.

5. Run the python file `install.py` in the root directory `pyfem-1.0` by double-clicking it. It creates the required executables and returns the total path in which PyFEM is installed. This path must be added to the environment variables `PYTHONPATH` and `PATH`.

   Windows has a built-in dialog for changing environment variables. In the case of Windows XP in the classical view:

   - Click your machine (usually located on your Desktop and called My Computer) and choose Properties there.
   - Then, open the Advanced tab and click the Environment Variables button.

   In short, your path is:

   ```
   My Computer->Properties->Advanced->Environment Variables
   ```

   In this dialog, you can add or modify user and system variables. To change system variables, you need non-restricted access to your machine (i.e. Administrator rights).

   The main program `pyfem` can be run from the command prompt. For example, in order to run the file `StressWave20x20.pro` in the directory `examples\ch05`, simply type:

   ```
   pyfem StressWave20x20.pro
   ```

   or by clicking any `.pro` file with the right mouse button and selecting the batch file `pyfem.bat` to execute it with.

2.2 Linux OS

The `python` program and the modules `numpy`, `scipy` and `matplotlib` are included in most common distributions of Linux and can be installed without any problems. In many cases, different versions of `python` are offered. Please make sure that `python` version 2.6 or 2.7 is installed (version 2.7 is preferred).
Run de `python` file `install.py` in the root directory `pyfem-1.0`. In a terminal, one can type:

```
python install.py
```

This script returns the total path in which PyFEM is installed. This path must be added to the environment variables `PYTHONPATH` and `PATH`. When using a bash shell, the following lines have to be added to the file `.bashrc` in your root directory:

```
export PYTHONPATH=<pyfemdir>
alias  pyfem="python <pyfemdir>/PyFEM.py"
```

When using csh or tcsh add the following lines to `.cshrc` or `.tcshrc`:

```
setenv PYTHONPATH <pyfemdir>
alias  pyfem "python <pyfemdir>/PyFEM.py"
```

It goes without saying that in the case of multiple `PYTHONPATH` settings, the path to PyFEM should be added to existing paths. For example, in the case of a bash shell, this will look like:

```
export PYTHONPATH=<pyfemdir>:$PYTHONPATH
```

The main program `pyfem` can be run from the command prompt. For example, in order to run the file `StressWave20x20.pro` in the directory `examples/ch05`, simply type:

```
pyfem StressWave20x20.pro
```

2.3 Mac OS

1. The most recent versions of Apple Mac-OS ship with their own version of `python`. However, it is strongly recommended to install the official `python` 2.7 at:

   ```
   http://www.python.org/getit
   ```

   The latest version is 2.7.3.

2. Download and install the latest version of `numpy`. This module is available at:

   ```
   http://sourceforge.net/projects/numpy/files/NumPy
   ```

   It is recommended to install the latest version, which is 1.6.2.

3. Download and install `scipy`. This module is available at:

   ```
   http://sourceforge.net/projects/scipy/files/scipy/
   ```

   It is recommended to install the latest version, which is 0.11.01b.

4. Download and install `matplotlib`. This module is available at:

   ```
   http://sourceforge.net/projects/matplotlib/files/matplotlib/
   ```

   It is recommended to install the latest version, which is version 1.1.0.

5. When all programs and packages mentioned above are installed, open a terminal and run the `python` file `install.py` in the root directory `pyfem-1.0`, by typing:

   ```
   python install.py
   ```

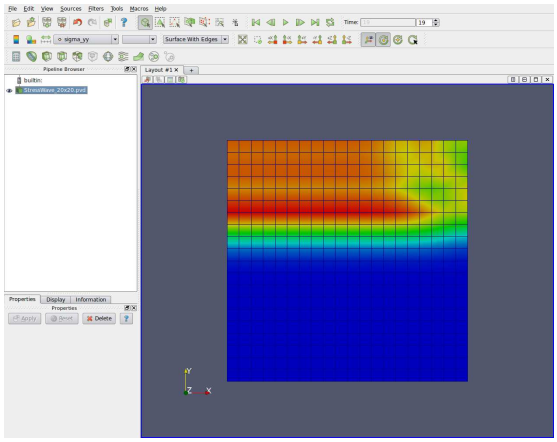


Figure 1: Screen shot of the results of the simulation `StressWave20x20.pro` shown in PARAVIEW.

This script returns the total path in which PYFEM is installed. This path must be added to the environment variables `PYTHONPATH` and `PATH`. The following lines have to be added to the file `.bashrc` in your root directory:

```
export PYTHONPATH="your_pyfem_path"
alias  pyfem     ="python your_pyfem_path/PyFEM.py"
```

2.4 Additional software

PYFEM can store the solution of a simulation in the `vtk`-format, which can be viewed with the program PARAVIEW. This program is available for free for academic use from the following website [1]:

`http://www.paraview.org`

The results are stored as a single `.pvd` file, which refers to a number of `.vtu` files. By opening the `.pvd` file in PARAVIEW one can see the deformed mesh and stress contours, as shown in Figure 1. A more detailed description how to create these output files is given in paragraph 5.1.

3 Quick start

In order to test whether everything is installed properly, the following two simulations can be run.

[1] Please read the terms on their website in case of non-academic use.

Simple example

In the directory `examples/ch02` the script `PatchTest.py` can be executed from a terminal (or DOS-shell) by typing:

```
python PatchTest.py
```

In Windows, this script can also be executed by double-clicking the icon.

PyFEM example

The full finite element code PyFEM can be run by typing `pyfem` in the terminal. In directory `examples/ch04` for example, the input file `ShallowTrussRiks.pro` is processed by typing:

```
pyfem ShallowTrussRiks.pro
```

Here, `ShallowTrussRiks.pro` is the input file, which by definition ends with `.pro`. When it is opened in a text editor, it looks as follows:

```
input = "ShallowTrussRiks.dat";

TrussElem  =
{
  ....
};

SpringElem =
{
  ....
};

solver =
{
  ....
};

outputModules = ["graph"];

graph =
{
  ....
};
```

The dots indicate lines that have been omitted in this example. The first argument in the `.pro`-file specifies the input file, which contains the positions of the nodes, the element connectivity and the boundary conditions. The structure of this file, which normally has the extension `.dat`, is as follows:

```
<Nodes>
  1   0.0 0.0 ;
  2 -10.0 0.0 ;
  3  10.0 0.0 ;
  4   0.0 0.5 ;
</Nodes>

<Elements>
  1 'TrussElem' 2 4 ;
  2 'TrussElem' 3 4 ;
  3 'SpringElem' 1 4 ;
</Elements>

<NodeConstraints>
  u[1] = 0.0;
  u[2] = 0.0;
  u[3] = 0.0;

  v[1] = 0.0;
  v[2] = 0.0;
  v[3] = 0.0;
</NodeConstraints>

<ExternalForces>
  v[4] = -100.0 ;
</ExternalForces>
```

The nodes are defined between the labels `<Nodes>` and `</Nodes>`. The first number indicates the node identification number. The remaining numbers denote the coordinates in x-, y-, and in the case of a three dimensional simulation, the z-direction. For example, node 2 has the x and y coordinates $(-10, 0)$.

The element connectivity is given after the tag `<Elements>`. The first number indicates the element ID number. The string refers to the name of the element model this element belongs to. The remaining numbers are the nodes that are used to construct the element. In this example, the first element is of the type `'TrussElem'` and is supported by nodes 2 and 4.

The boundary conditions and applied loads are specified next. The node constraints are given after the label `<NodeConstraints>`. In this example, the displacement components `u` and `v` of nodes 1,2 and 3 have a prescribed value of 0.0. The external forces are specified in a similar manner in the field `<ExternalForces>`. Here, a unit external force with magnitude -100.0 is added to node number 4 in the direction that corresponds to the `'v'` displacement. Hence, this force is acting in the negative y-direction.

The parameters of the finite element model are specified in the `.pro` file, in the fields `TrussElem` and `SpringElem`, which refer to the labels used in the

element connectivity description.

```
TrussElem  =
{
  type = "Truss";
  E    = 5e6;
  Area = 1.0;
};

SpringElem =
{
  type = "Spring";
  k    = 100.0;
};
```

The elements denoted by the label `TrussElem` are of the type `'Truss'`. This model requires two additional parameters, the Young's modulus of the material `E` and the area of the cross-section `Area`. The label `SpringElem` denote elements of the type `'Spring'`. Here, one additional parameter is required: the spring stiffness `k`. A detailed overview of the element types and the corresponding parameters can be found in Section 6 of this manual.

The parameters of the solver are defined next:

```
solver =
{
  type = 'RiksSolver';

  fixedStep = true;
  maxLam    = 10.0;
};
```

The solver is of the type `'RikSolver'`. The two additional parameters specify that the magnitude of the path-parameter is constant (`fixedStep = true`) and that the simulation is stopped when the load parameter λ reaches a value of 10.0. A detailed overview of available solver types and their parameters is given in Section 4.

Finally, the results of the simulation can be stored and visualised in several ways. To this end, a chain of output modules can be specified. In this example, the results are stored in a load-displacement curve in the module `GraphWriter`.

```
outputModules = ["graph"];

graph =
{
  type     = "GraphWriter";
  onScreen = true;

  columns = [ "disp" , "load" ];

  disp =
  {
    type   = "state";
    node   = 4;
    dof    = 'v';
    factor = -1.0;
  };

  load =
  {
    type = "fint";
    node = 4;
    dof  = 'v';
  };
};
```

In this example, two colums are stored: `'disp'`, the displacement (`'state'`) of node 4 in the vertical direction and `'load'`, the corresponding internal force. The parameter `onScreen = true` is used to show the load-displacement curve on the screen during the simulation. By default, the results will be stored in a file called `ShallowTrussRiks.out`. A description of all available output modules can be found in Section 5.

4 Solvers

In this section, a concise overview of the solvers that are available in PyFEM is given.

4.1 Linear solver

The linear solver is discussed in detail in Section 2.6 of the book.

Name: `LinearSolver`

Source: `pyfem/solver/LinearSolver.py`

Mandatory parameters:

None

Optional parameters:

None

Examples:

`ch02`: `PatchTest4.pro`

`ch02`: `PatchTest8.pro`

4.2 Non-linear (Newton-Raphson) solver

The Newton-Raphson solver can be used to solve non-linear systems with a monotonously increasing external load or prescribed displacement. The exact procedure is discussed in detail in Sections 2.4 and 2.5 of the book.

Name: `NonlinearSolver`

Source: `pyfem/solver/NonlinearSolver.py`

Mandatory parameters:

None

Optional parameters:

`maxLam`	The maximum load parameter λ for which the ‘ simulation will be terminated.
`maxCycle`	The number of load cycles (loading steps) after which the simulation will be terminated.
`tol`	The precision that is used to determine whether a solution is converged. The default value is set to 10^{-3}.

Examples:

`ch03`: `cantilever8.pro`

`ch06`: `ContDamExample.pro`

4.3 Riks’ arc-length solver

Riks’ arc-length method allows to solve problems in which the load parameter is not monotonously increasing. The solver is discussed in detail in Section 4.2 of the book. The source code is explained in detail in Section 4.3.

Name: `RiksSolver`

Source: `pyfem/solver/RiksSolver.py`

Mandatory parameters:

None

Optional parameters:

`maxFactor`	The maximum for which the path-parameter may increase with respect to the magnitude of the path-parameter in the first step.
`fixedStep`	If set to `true` a constant step size is used. This is identical to `maxFactor=1`. The default value is `false`.
`opt`	Optimal number of iterations, see Section 4.5 of the book for further details.
`tol`	The precision that is used to determine whether a solution is converged. The default value is set to 10^{-3}.
`maxLam`	The maximum load parameter λ for which the ' simulation will be terminated.

Examples:

ch04:	`ShallowTrussRiks.pro`
ch09:	`FrameKirchhoff.pro`
ch09:	`FrameTimoshenko.pro`
ch09:	`KirchhoffEuler_01.pro`
ch09:	`KirchhoffEuler_1.pro`
ch09:	`KirchhoffEuler.pro`

4.4 Dissipated energy solver

This is the dissipated energy based arc-length solver as described in Section 4.2, page 123 of the book.

Name: `DissipatedEnergySolver`

Source: `pyfem/solver/DissipatedEnergySolver.py`

Mandatory parameters:

`switchEnergy`	Amount of dissipated energy in a single step for which the solution technique will switch from force controlled to energy dissipation controlled.

Optional parameters:

`maxCycle`	Number of cycles after which the simulation will be terminated.
`maxdTau`	Maximum amount of energy that may be dissipated in a single step.
`tol`	The precision that is used to determine whether a solution is converged. The default value is set to 10^{-3}.
`maxLam`	The maximum load parameter λ for which the ‘ simulation will be terminated.

Examples:

`ch13`:	`PeelTest.pro`

4.5 Explicit time integration solver

The explicit time integration solver is discussed in detail in Section 5.2 of the book. The source code is explained in detail in Section 5.3 of the book.

Name: `ExplicitSolver`

Source: `pyfem/solver/ExplicitSolver.py`

Mandatory parameters:

`dtime`	Magnitude of time step
`lam`	Load factor λ as a function of time. This can be written as a string. For example, `'4.0*sin(3.0*t'` represents a sinusoidal load, with period 3.0 and amplitude 4.0.

Optional parameters:

`maxCycle`	Number of cycles after which the simulation will be terminated.
`maxTime`	Time after which the simulation will be terminated.

Examples:

`ch05`:	`StressWave20x20.pro`

5 Output modules

5.1 Mesh output writer

The mesh output writer saves all data during a simulation to the disk. The data is organised as follows: during a simulation, a single output file `filename.pvd` will be created which refers to the output of single steps, which are stored in the file `filename-xx.vtu`, where `xx` indicates the step number. This data can be visualised by opening the file `filename.pvd` in the external program ParaView.

Name: `MeshWriter`

Source: `pyfem/io/MeshWriter.py`

Mandatory parameters:

None

Optional parameters:

`prefix`	The prefix of the output filename that will be used. By default, the prefix of the input filename is used.
`interval`	The interval (number of cycles) for which output is stored. By default, every step is stored.
`elementgroup`	When specified, only the elements in this group will be stored. By default, all elements will be stored.

Examples:

ch03:	`cantilever8.pro`
ch05:	`StressWave20x20.pro`
ch06:	`ContDamExample.pro`
ch13:	`PeelTest.pro`

5.2 Graph output writer

The output is stored in a multi column file by this writer. The first two columns can be shown on the screen as a curve during the simuluation.

Name: `GraphWriter`

Source: `pyfem/io/GraphWriter.py`

Mandatory parameters:

`columns`	Array of strings indicating the column that will be stored. For each column, the type of data, and if needed, the node, degree of freedom and scaling factor needs to be specified.
`type`	Type of data. This can be either `state`, `velo`, `fint`, `stress`, etc.
`node`	Node ID.
`dof`	Degree of freedom. This is most likely `'u'` or `'v'`

Optional parameters:

`factor`	The scaling factor for the output. The default value is 1.0.
`onScreen`	When set to `true` the first two columns will be shown on the screen. The default value is `false`.

Examples:

ch04:	`ShallowTrussRiks.pro`
ch06:	`ContDamExample.pro`
ch09:	`FrameKirchhoff.pro`
ch09:	`FrameTimoshenko.pro`
ch09:	`KirchhoffEuler_01.pro`
ch09:	`KirchhoffEuler_1.pro`
ch09:	`KirchhoffEuler.pro`
ch13:	`PeelTest.pro`

6 Elements

In this section, a list of elements available in PyFEM is given.

6.1 Finite strain continuum

The finite strain continuum element is discussed in detail in Section 3.6 of the book. In the code, the two dimensional version is implemented. It can be used as a 3,4,6,8 and 9 node element.

Name: `FiniteStrainContinuum`

Source: `pyfem/materials/FiniteStrainContinuum.py`

Mandatory parameters:

`material`	The material model that is used in this element, see Section 7 for more details.

Optional parameters:

None

Examples:

`ch03`:	`cantilever8.pro`
`ch05`:	`StressWave20x20.pro`

6.2 Kirchhoff non-linear beam

The Kirchhoff beam element is discussed in Section 9.2.

Name: `KirchhoffBeam`

Source: `pyfem/elements/KirchhoffBeam.py`

Mandatory parameters:

`E`	Young's modulus
`A`	Cross-section of the truss
`I`	Moment of inertia

Optional parameters:

None

Examples:

`ch09`:	`FrameKirchhoff.pro`

6.3 Small strain continuum

The small strain continuum element is discussed in detail in Section 2.6 of the book. In the code, the two dimensional version is implemented. It can be used as a 3,4,6,8 and 9 node element.

Name: `SmallStrainContinuum`

Source: `pyfem/materials/SmallStrainContinuum.py`

Mandatory parameters:

`material` Material Model, see Section 7

Optional parameters:

None

Examples:

`ch02`: `PatchTest4.pro`

`ch02`: `PatchTest8.pro`

`ch06`: `ContDamExample.pro`

`ch13`: `PeelTest.pro`

6.4 Linear spring

The linear spring is used in the Shallow Truss examples in the first chapters of the book.

Name: `Spring`

Source: `pyfem/elements/Spring.py`

Mandatory parameters:

`k` Spring stiffness

Optional parameters:

None

Examples:

`ch04`: `ShallowTrussRiks.pro`

6.5 Timoshenko non-linear beam

The Timoshenko beam element is discussed in Section 9.2 of the book.

Name: `TimoshenkoBeam`

Source: `pyfem/elements/TimoshenkoBeam.py`

Mandatory parameters:

`E`	Young's modulus
`A`	Cross-section of the truss
`I`	Moment of intertia
`G`	Shear modulus

Optional parameters:

None

Examples:

`ch09`: `FrameTimoshenko.pro`

6.6 Non-linear truss

The non-linear truss element is discussed in Sections 3.1 and 3.2 of the book.

Name: `Truss`

Source: `pyfem/elements/Truss.py`

Mandatory parameters:

`E`	Young's modulus
`A`	Cross-section of the truss

Optional parameters:

None

Examples:

`ch04`: `ShallowTrussRiks.pro`

6.7 Cohesive zone interface

The Cohesive zone interface element is discussed in Section 13.2 of the book.

Name: `Interface`

Source: `pyfem/materials/Interface.py`

Mandatory parameters:

`material`	Material Model, see Section 7

Optional parameters:

`intmethod`	Integration method, this can be either `Gauss`, `Newton-Cotes` or `Lobatto`. The default is `Newton-Cotes`
`intorder`	Integration order. The level of over- or underintegration is specified here as an integer (e.g. `+2` or `-1`). Default value is `0`.

Examples:

`ch13`:	`TractionOscillation.pro`
`ch13`:	`PeelTest.pro`

7 Material models

In this section, the input parameters for the different material models that are available in PyFEM are given.

7.1 Plane strain linear elastic model

A plane strain, linear elastic constitutive relation as presented on page 109-110 of the book.

Name: `PlaneStrain`

Source: `pyfem/materials/PlaneStrain.py`

Mandatory parameters:

`E`	Young's modulus
`nu`	Poisson's ratio

Optional parameters:

None

Examples:

`ch02:`	`PatchTest4.pro`
`ch02:`	`PatchTest8.pro`
`ch05:`	`StressWave20x20.pro`
`ch13:`	`TractionOscillation.pro`
`ch13:`	`PeelTest.pro`

7.2 Plane strain damage

See Section 6.2 in the book for a detailed description.

Name: `PlaneStrainDamage`

Source: `pyfem/materials/PlaneStrainDamage.py`

Mandatory parameters:

`E`	Young's modulus
`nu`	Poisson's ratio
`kappa0`	Equivalent strain at which damage intiates.
`kappac`	Equivalent strain at which damage is 1.0.

Optional parameters:

None

Examples:

`ch06:`	`ContDamExample.pro`

7.3 Plane stress linear elastic

Name: `PlaneStrain`

Source: `pyfem/materials/PlaneStrain.py`

Mandatory parameters:

`E`	Young's modulus
`nu`	Poisson's ratio

Optional parameters:

None

Examples:

7.4 Power Law cohesive model

Name: `PowerLawModeI`

Source: `pyfem/materials/PowerLawModeI.py`

Mandatory parameters:

`Tult`	Ultimate traction
`Gc`	Fracture toughness

Optional parameters:

None

Examples:

`ch13`: `PeelTest.pro`

8 Version history

1.0	August 29, 2012	• First major release.