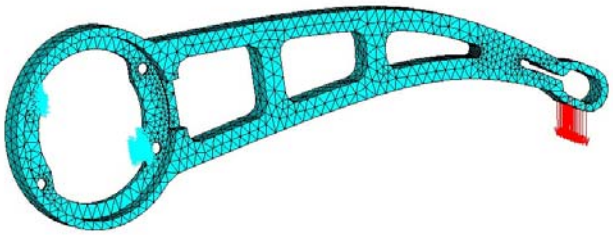
FELyX
Finite Element Library eXperiment

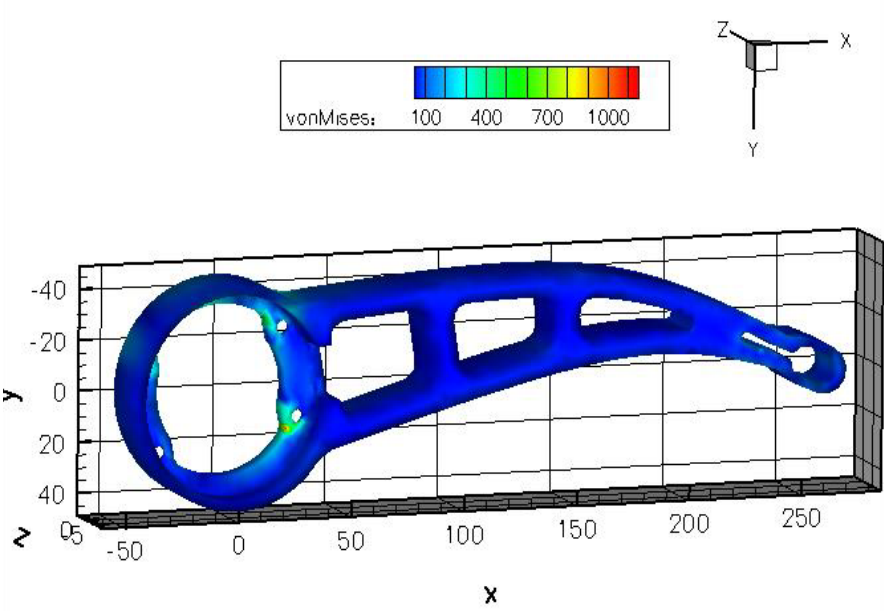
# *Winschhebel Manual:*

*Import ANSYS database*

*Solve*

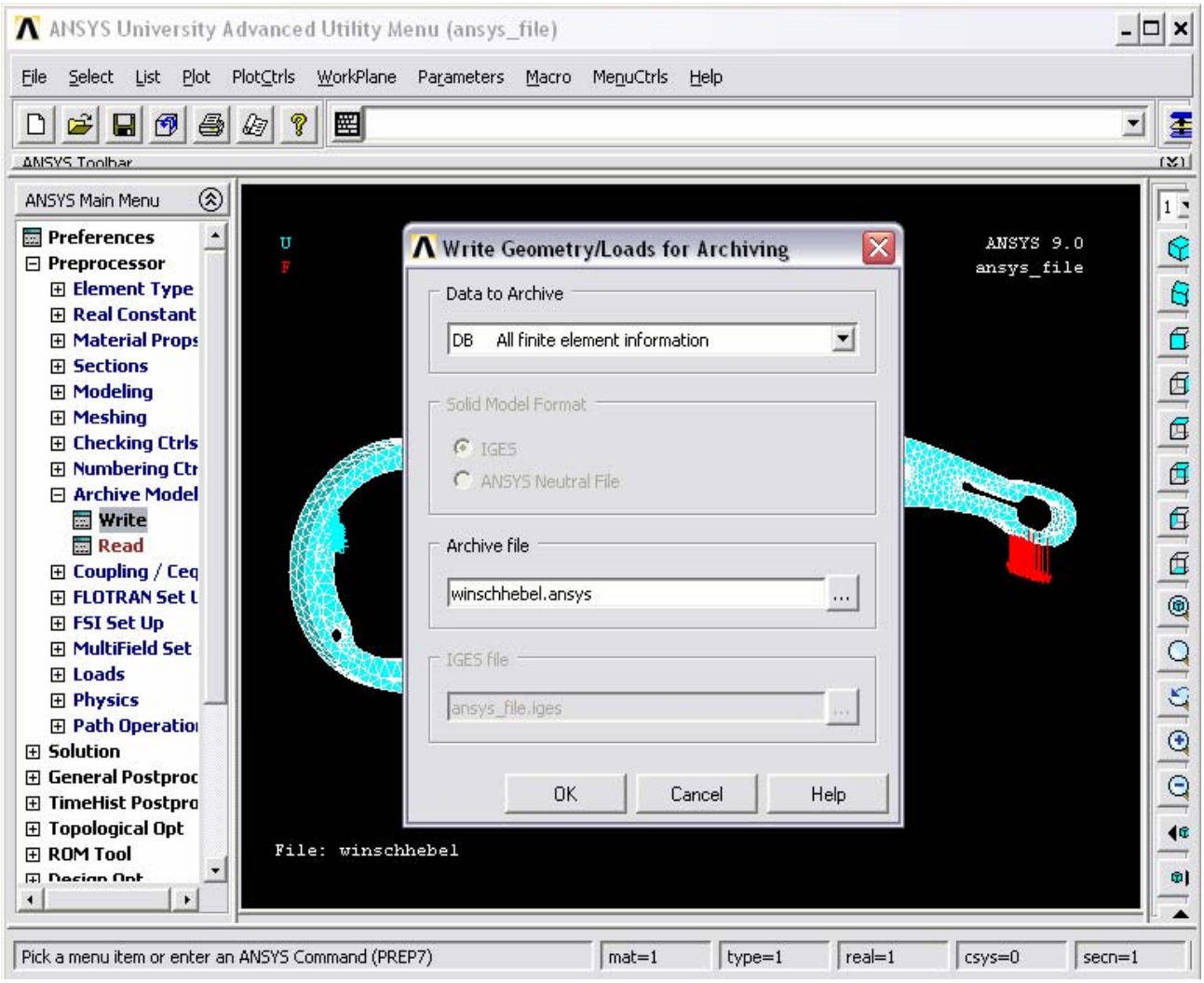
*Postprocessing*





# *Export database from ANSYS*

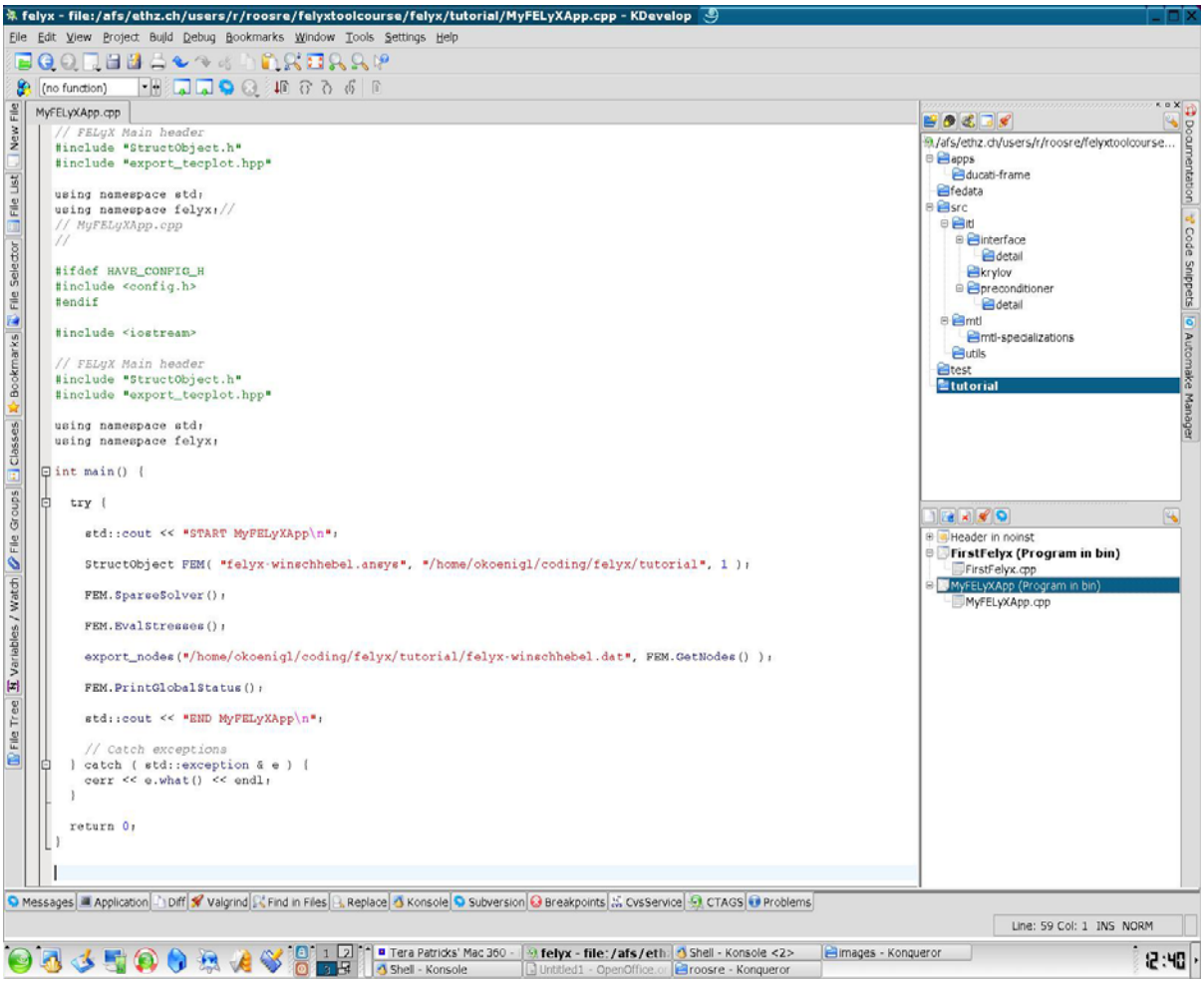
- Preprocessing
  - ANSYS offers a tool to export the FE Model
  - FELyX needs all FE information
  - Path: Preprocessor – Archive Model – Write and then select in “Data to archive” “DB all finite element information”
  - Or with this command: CDWRITE,DB,filename,ansys,,,



# New Executable in FELyX

- Download FELyX from the CVS Repository
  - Type this command in Linux shell

  cvs –z3 –d:pserver:anonymous@cvs.sourceforge.net:/cvsroot/felyx co –P felyx
- Our developer tool is KDevelop
  - Shell command: kdevelop felyx.kdevelop
  - Have a look into the *Automake Manager*

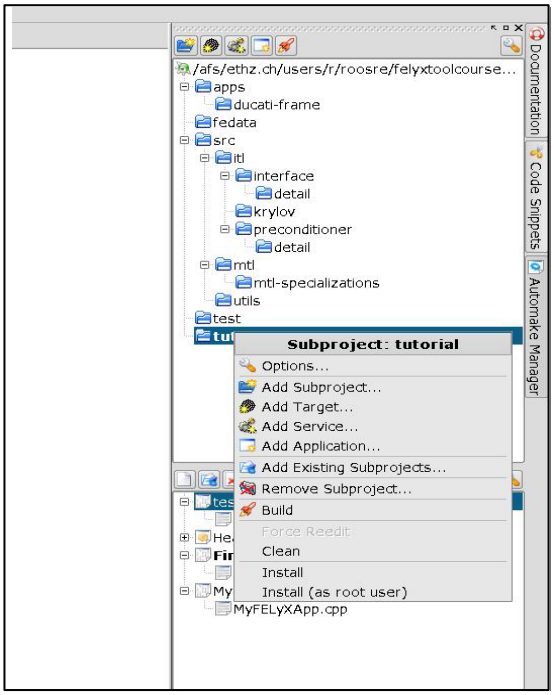


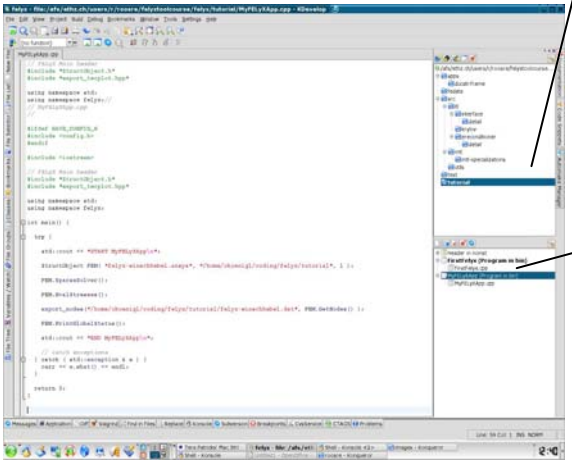
- **Add in FELyX project a new executable**
  - Open *Automake Manager* and choose *Add Target* in the tutorial folder (right mouse button)
  - In this case the new Target is a program

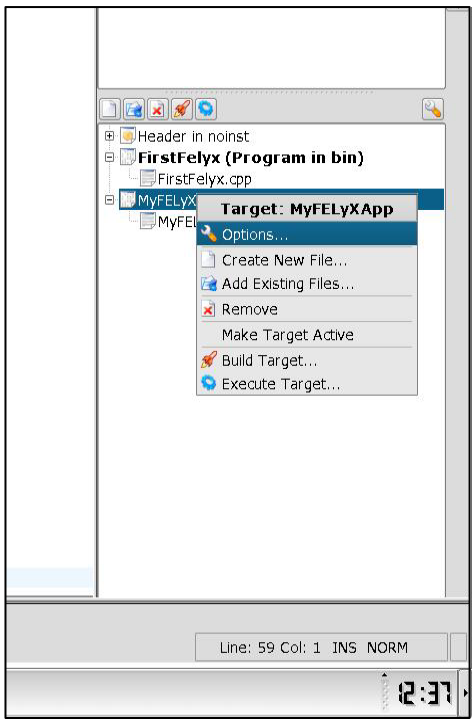
  *(first illustration)*

  - In this target you can *Creat New File* and link this file with this target
  - The new file is C++ source code
  - *Make Target Active*

  *(second illustration)*





- **StructObject, Solve, Postprocessing and Save Results**
  - Include FEM Object in your new executable

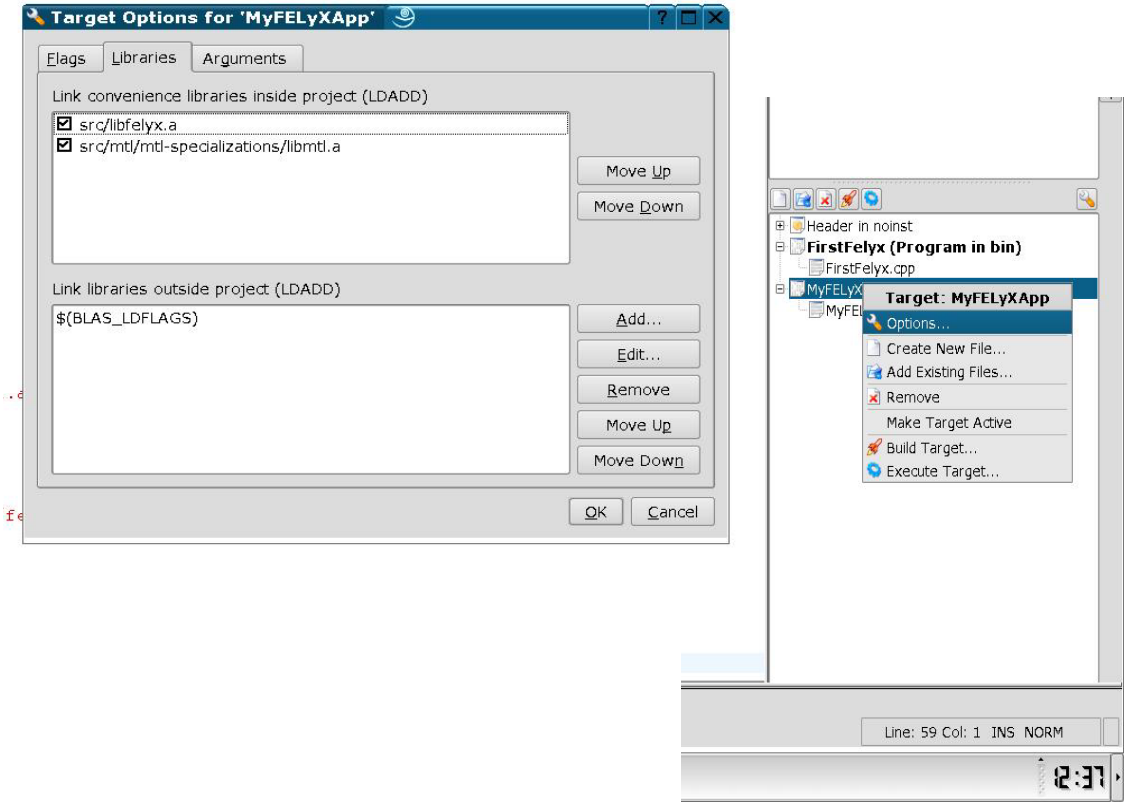
```
// MyFELyXApp.cpp
//

#ifdef HAVE_CONFIG_H
#include <config.h>
#endif

#include <iostream>
// FELyX Main header
#include "StructObject.h"
#include "export_tecplot.hpp"

using namespace std;
using namespace felyx;

int main() {

    std::cout << "START MyFELyXApp\n";

    StructObject FEM( "winschhebel.ansys", "path", 1 );

    FEM.SparseSolver();

    FEM.EvalStresses();

    export_nodes("/path/winschhebel.dat", FEM.GetNodes() );

    FEM.PrintGlobalStatus();

    std::cout << "END MyFELyXApp\n";

  return 0;
}
```

```
// comments

Include header files

Include used FELyX header files

Define namespaces

Import FE database winschhebel.ansys from
location e.g. /home/felyx/totorial
Solve FE Model by using a sparse solver

Postprocessing

Export solution in tecplot-format

Writes main results

End
```

  - The same example is already attached in /felyx/tutorial MyFELyXApp.cpp
  - For a better understanding you can use the doxygen documentation
    - Choose in kdevelop: Build – Build API Documentation
  - This generates a html – documentation in the felyx/doc/html folder
  - This documentation describes the classes, functions and dependencies of the different objects

# Run the new Executable

- Build the active target
  - Build and execute the active target
  - The main results are printed
- Errors
  - Sometimes you have to add some libraries to the active target or put them into the right order. Most suitable is to compare with the other executables and make it in the same manner
  - Choose *Options* in the active target and *add, edit or move the libraries*



# *Postprocessing*

- Stresses and Deformations
  - In this case we write the deformations and stresses into a dat-file
    export_nodes("/path/winschhebel.dat", FEM.GetNodes() );
  - You can use *tecplot* to illustrate this deformations and stresses
  - Plots for the deformation in y-Direction calculated by ANSYS and FELyX