

NLS++

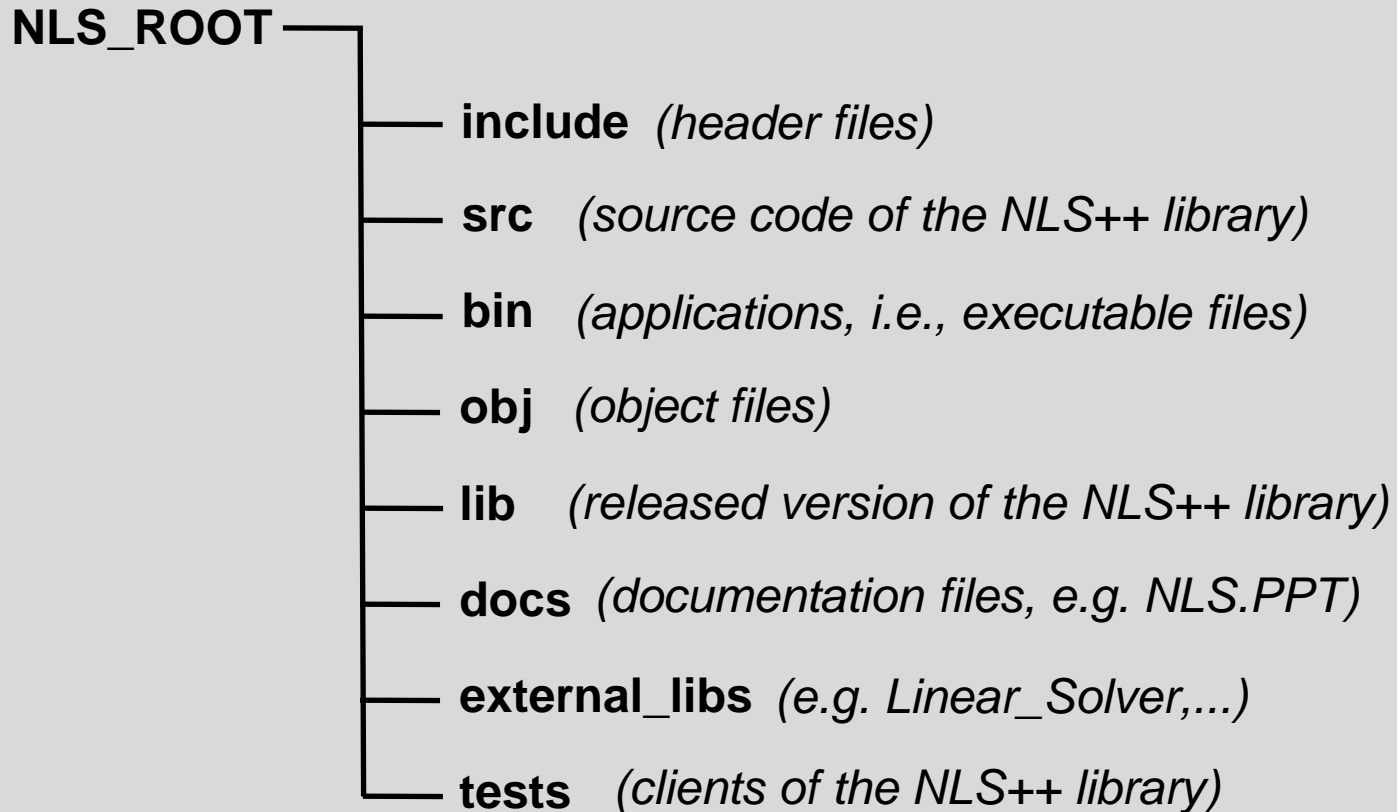
A Computational Program for Solving Non Linear Systems of Equations

Developed by:

- Anderson Pereira
- Ivan Menezes
- Glaucio Paulino



Directory Structure



Creating (coding) a New Application

```
#include <nls.h>

class cMyModel : public cModel
{
    ...
};

int main()
{
    cMyModel *pcModel = new cMyModel();
    cLynSys *pcLinSys = new cCroutProfile( );
    ...
    cCtrl *pcCtrl = new cNewtonRaphson( pcModel, &sCtrl, pcLinSys );
    ...
    pcCtrl->Solver( );
}
```

File: myapp.cpp

Generating a New Application

Compilation Step:

```
[~]g++ -c myapp.cpp -I<NLS_ROOT>/include  
      -I<NLS_ROOT>/external_libs/LinSys>/include
```

Linking Step:

```
[~]g++ -o myapp myapp.o  
      <NLS_ROOT>/lib/<System>/libnls.a  
      <NLS_ROOT>/external_libs/LinSys>/lib/<System>/liblinsys.a
```

Notes:

1. <NLS_ROOT> is the start directory of the NLS++ (as shown in slide 2)
2. <System> is the user operational system (e.g., Linux24g3, vc9, gcc3, ...)

General Comments

- Once the NLS++ library is generated for a particular operational system, it does not need to be changed anymore.
- The users can derive their own models or use the existing ones (see directory: “<NLS_ROOT>/tests”).
- The libraries for the solution of LINEAR systems of equations are provided for several operational systems (see directory: “<NLS_ROOT>/external_libs”). The corresponding source code is NOT available here.

Testing the NLS++ Lib

Main Steps:

- <1> Generate the NLS++ Library: Type “make” in the directory “NLS_ROOT/src” (Note: make sure that the directories “NLS_ROOT/lib” and “NLS_ROOT/obj” have already been created).
- <2> Copy all include files (from all “NLS_ROOT/src” subdirectories) to the “NLS_ROOT/include” directory (Note: you can use the script file “copy_includes.sh” which is provided in “NLS_ROOT/src” directory).
- <3> Generate your application (see slides 3 and 4).
- <4> Run your application by typing:

```
[NLS_ROOT/tests] ../bin/nls model.inp algorithm.inp
```

where: model.inp = model file name*
 algorithm.inp = algorithm file name*

**model and algorithm files are described in slides 9-12*

Available Algorithms in the NLS++ Lib

- Newton-Raphson
- Displacement Control
- Work Control
- Arc-Length
- Generalized Displacement Control
- Orthogonal Residual Procedure

Main Papers:

- W.F. **Lam** and C.T. **Morley**, “*Arc-length method for passing limit points in structural calculations*”, Journal of Structural Engineering (ASCE), 118(1), 169-185, **1992**.
- M.A. **Crisfield**, “*A fast incremental/iterative solution procedure that handles snap-through*”, Computers & Structures, 13, 55-62, **1981**.
- S. **Krenk**, “*An orthogonal residual procedure for nonlinear finite element equations*”, IJNME, 38, 823-839, **1995**.
- Y.B. **Yang** and M.S. **Shieh**, “*Solution method for nonlinear problems with multiple critical points*”, AIAA Journal, 28(12), 2110-2116, **1990**.

Main Books:

- J.N. **Reddy**, “*An Introduction to Nonlinear Finite Element Analysis*”, Oxford, **2004**.
- K.J. **Bathe**, “*Finite Element Procedures*”, Prentice-Hall, **1996**.
- M.A. **Crisfield**, “*Non-Linear Finite Element Analysis of Solids and Structures*”, John Wiley and Sons, Volume 1, **1991**.

Available Models for Testing NLS++ Lib

- **Function Defined by an Equation**

Class: cModelFunction

File: modfunc.cpp

- **Space Truss (Plane Truss is a particular case)**

Class: cModelTruss

File: modst.cpp

- **Plane Frame**

Class: cModelBeam2D

File: modbeam2.cpp

- **Space Frame**

Class: cModelBeam3D

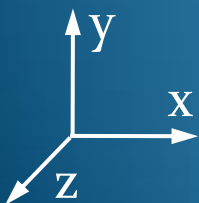
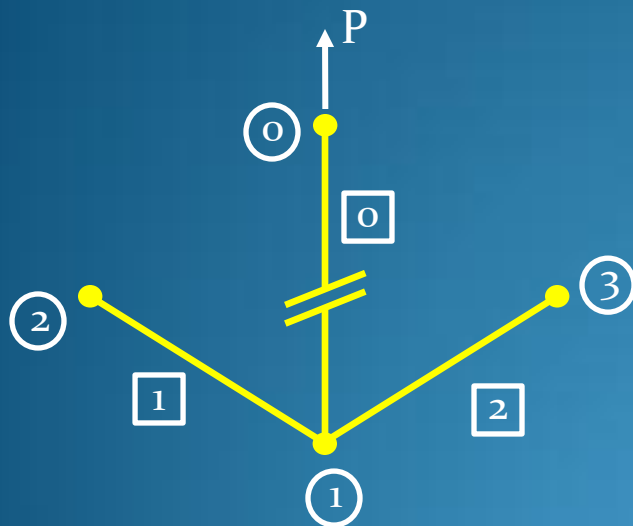
File: modbeam3.cpp

Note: The above “cpp” files can be found in “<NLS_ROOT>/tests” directory

"Model" File (Example 1)

von Mises Truss

(modst.cpp)



○ Node
□ Element

model type → 'space_truss'

of nodes

c_x c_y c_z

4
0.0 1000.0 0.0
0.0 0.0 0.0
-8.660254037 5.0 0.0
8.660254037 5.0 0.0

of bound. cond.

node d_x d_y d_z

4
0 1 0 1
1 1 0 1
2 1 1 1
3 1 1 1

of loads

node f_x f_y f_z

1
0 0.0 1.0 0.0

of elements

node₁ node₂ EA, σ_o

3
0 1 50.0 0.0
1 2 1.0 0.0
1 3 1.0 0.0

of disp. curves

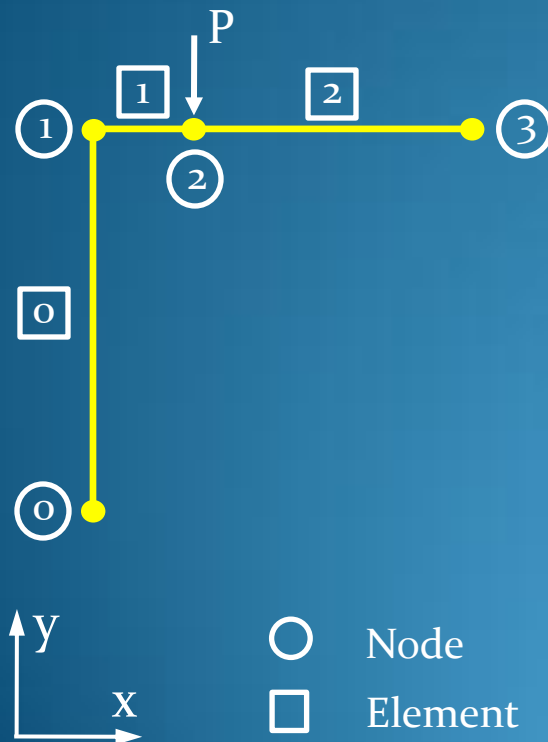
node d.o.f.

2
0 1
1 1

Initial axial stresses

“Model” File (Example 2)

Lee Frame
(modbeam2.cpp)



model type → 'plane_frame'

of nodes

c_x c_y

4

0 0
0 120
24 120
120 120

of bound. cond.

node d_x d_y r_z

2

0 1 1 1
3 1 1 1

of loads

node f_x f_y m_z

1

2 0 1 0

of elements

node₁ node₂ EA GA EI

3

0 1 4320 2160 1440
1 2 4320 2160 1440
2 3 4320 2160 1440

of disp. curves

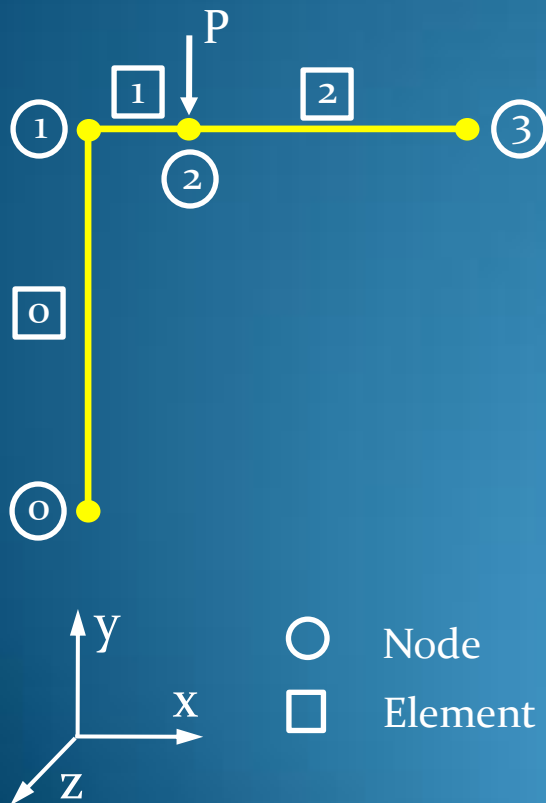
node d.o.f.

2

2 0
2 1

“Model” File (Example 3)

Lee Frame
(modbeam3.cpp)



model type \longrightarrow 'space_frame'

of nodes

$c_x \ c_y \ c_z$

4

0 0 0
0 120 0
24 120 0
120 120 0

of bound. cond.

node $d_x \ d_y \ d_z \ r_x \ r_y \ r_z$

2

0 1 1 1 1 1 1
3 1 1 1 1 1 1

of loads

node $f_x \ f_y \ f_z \ m_x \ m_y \ m_z$

1

2 0 1 0 0 0 0

of elements

node₁ node₂ EA EI_x EI_y EI_z ...
... GA_y GA_z v_x v_y v_z

3

0 1 4320 1000 1440 1440 2160 2160 0 0 1
1 2 4320 1000 1440 1440 2160 2160 0 0 1
2 3 4320 1000 1440 1440 2160 2160 0 0 1

of disp. curves

node d.o.f.

2

2 0
2 1

unit vectors to define
member orientation

“Algorithm” File

Line 1	Linear solver type	Parameter 1	Parameter 2
Crout-Profile	0	none	none
PCG-Profile *	1	Max. number of iterations	Tolerance for convergence
PCG-CSR *	2	Max. number of iterations	Tolerance for convergence

Line 2	Alg. type	Update type **	Initial load factor	Parameter 1	Parameter 2
Newton-Raphson	0	none	none	none	
Displacement control	1	D.o.f. of the controlled equation		Type (Constant <0>, Variable <1>)	
Arc-length	2	Type (Constant <0>, Variable <1>)		none	
Work control	3	none		none	
Gen. disp. control	4	none		none	
Ort. residual proc.(unified)	5	Initial inc. scale factor		none	
Strain ratio control	6	Load increment for the first step		none	
Strain control	7	D.o.f. of the controlled deformation		Type (Constant <0>, Variable <1>)	
Ort. residual proc. (original)	8	Initial inc. scale factor		none	

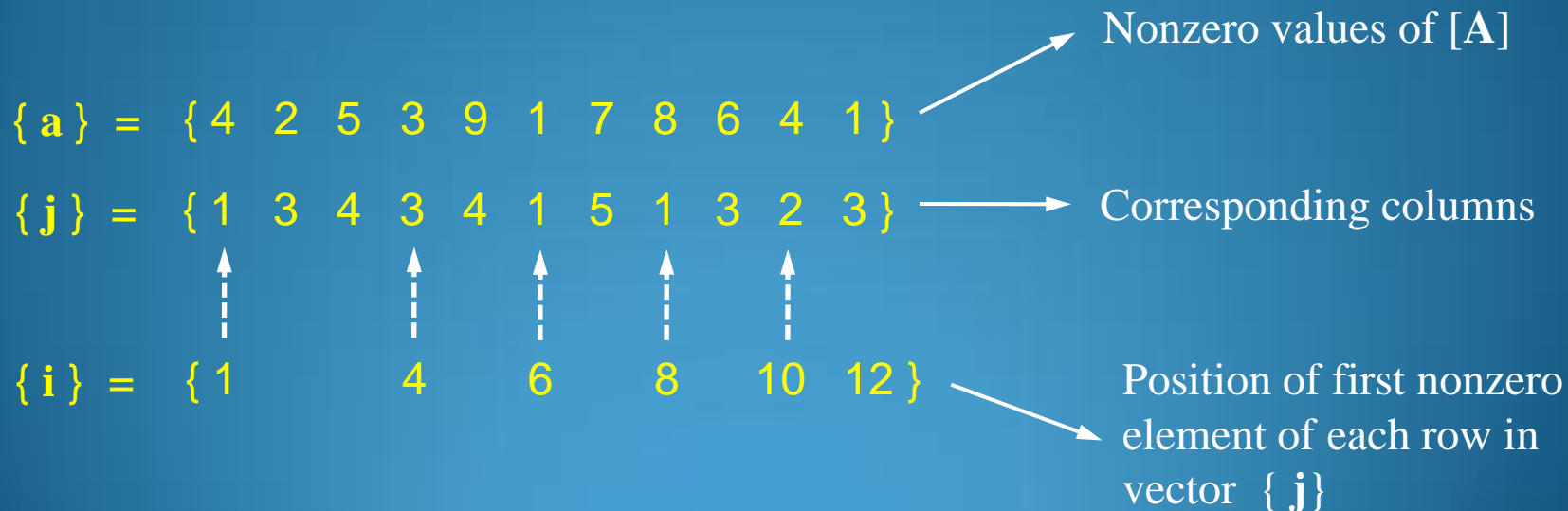
Line 3	Max. # of steps	Max # of iterations	Tolerance for convergence

*Profile = Skyline Storage; CSR = Compressed Sparse Row
(see slide 13 for more details about CSR scheme)

**Stiffness Matrix Update : <0>Standard <1>Modified

CSR Storage Scheme

$$[A] = \begin{pmatrix} 4 & 0 & 2 & 5 & 0 \\ 0 & 0 & 3 & 9 & 0 \\ 1 & 0 & 0 & 0 & 7 \\ 8 & 0 & 6 & 0 & 0 \\ 0 & 4 & 1 & 0 & 0 \end{pmatrix}$$



Example (Lee Frame)

'plane_frame'

11

0.000000E+000	0.000000E+000
0.000000E+000	2.400000E+001
0.000000E+000	4.800000E+001
0.000000E+000	7.200000E+001
0.000000E+000	9.600000E+001
0.000000E+000	1.200000E+002
2.400000E+001	1.200000E+002
4.800000E+001	1.200000E+002
7.200000E+001	1.200000E+002
9.600000E+001	1.200000E+002
1.200000E+002	1.200000E+002

2

0	1	1	0
10	1	1	0
1			
6	0	-1	0

10

0	1	4.320000E+003	2.160000E+003	1.440000E+003
1	2	4.320000E+003	2.160000E+003	1.440000E+003
2	3	4.320000E+003	2.160000E+003	1.440000E+003
3	4	4.320000E+003	2.160000E+003	1.440000E+003
4	5	4.320000E+003	2.160000E+003	1.440000E+003
5	6	4.320000E+003	2.160000E+003	1.440000E+003
6	7	4.320000E+003	2.160000E+003	1.440000E+003
7	8	4.320000E+003	2.160000E+003	1.440000E+003
8	9	4.320000E+003	2.160000E+003	1.440000E+003
9	10	4.320000E+003	2.160000E+003	1.440000E+003

2

6 0

6 1

0

4 0 0.10

250 40 0.0001

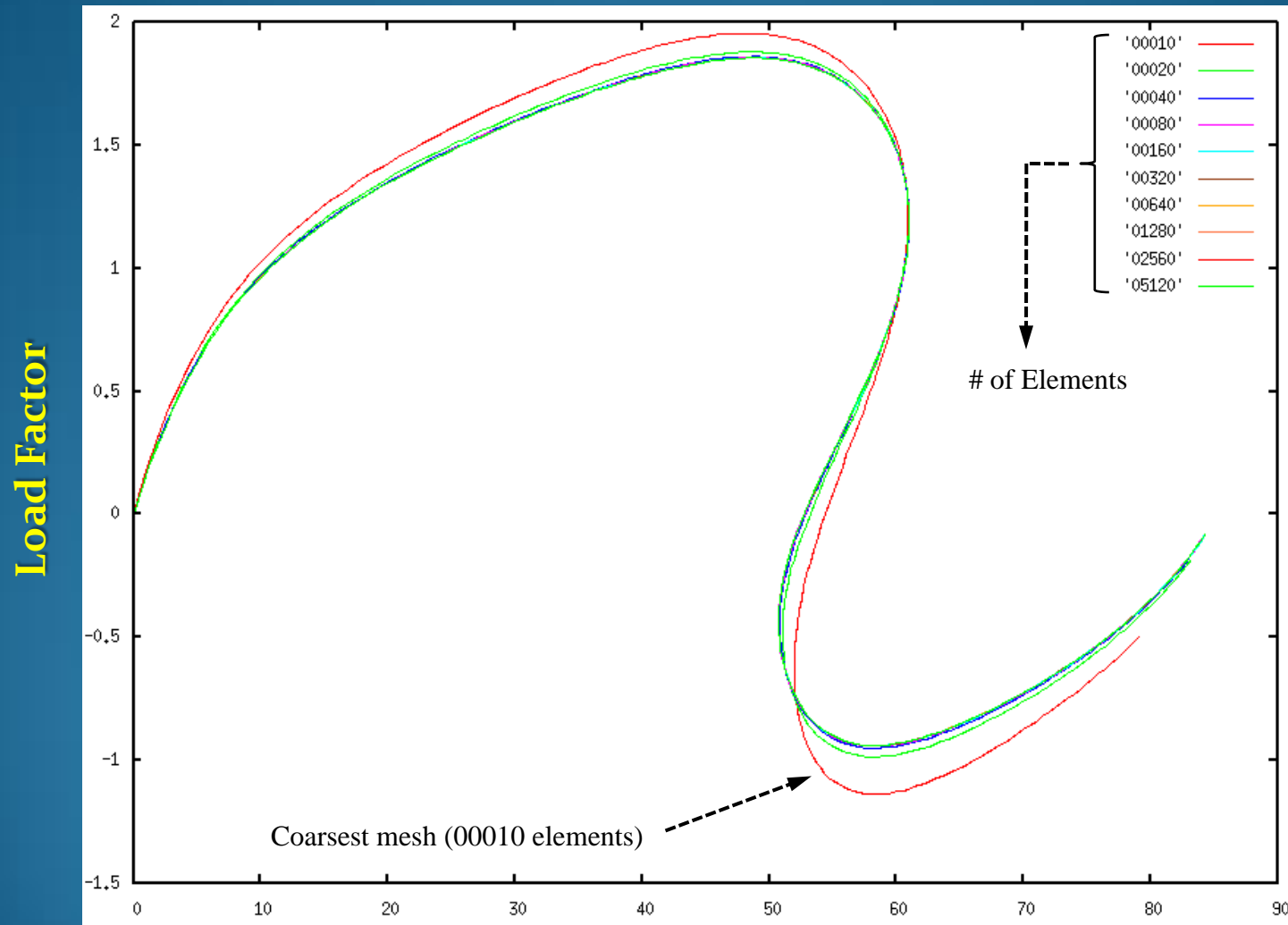
Algorithm File
(Crout Solver)

Model File
(Plane Frame)

Example (Lee Frame)

Total # of Elements	CPU Time (s)
10	0.374
20	0.639
40	0.983
80	1.888
160	3.448
320	6.318
640	11.684
1280	22.543
2560	44.772
5120	87.985

Example (Lee Frame)



Vertical Displacements (where the load is applied)

Ongoing Work

- Implementing NLS++ in the context of TopFEM program
- Deriving new models for testing the NLS++

*Anderson Pereira (anderson@tecgraf.puc-rio.br),
Ivan Menezes (ivan@tecgraf.puc-rio.br), and
Glaucio Paulino (paulino@illinois.edu)*

