Master's thesis in Mechanical Engineering:

Development of the Finite Element Method in CHRONO::ENGINE

UNIVERSITA’ DEGLI STUDI DI PARMA

via Università, 12 - I 43121 Parma
www.unipr.it

Presentation for the Applied Computing Summer School. University of Wisconsin - Madison. 08/08/2013
Chrono::Engine FEM classes:

**Elements.**

- **ChElementBase**
  - Base class for all finite elements, that can be used in the ChMesh physics item.

- **ChElementGeneric**
  - Class for all elements whose stiffness matrix can be seen as a NxN block-matrix to be splitted between N nodes. Helps reducing the complexity of inherited FEM elements because it implements some bookkeeping for the interface with LCP solver.

- **ChElement3D**
  - Class for all 3 dimensional elements.

  - **ChTetrahedron**
    - **ChElementTetra_4**
    - **ChElementTetra_10**

  - **ChHexahedron**
    - **ChElementHexa_8**
    - **ChElementHexa_20**
Chrono::Engine FEM classes:

Nodes.

ChNodeBase

ChNodeXYZ

ChNodeFEMbase

ChNodeFEMxyz

ChNode...

Class for a generic node, that has some degrees of freedom and that contain a proxy to the solver.

Class for a single 'point' node, that has 3 degrees of freedom and a mass.

Base class for a generic finite element node that can be stored in ChMesh containers. Children classes must implement specialized versions.

Class for a generic finite element node in 3D space, with x,y,z displacement.

- 3 DOF
- ID
- Reference position
- Actual position
- Speed
- Acceleration
Chrono::Engine FEM classes:

**Mesh.**

- **ChShared**: Base class for shared objects, i.e. objects which must be managed with smart pointers of intrusive type (where the reference counting method is used to understand when to delete the object).
- **ChObj**: Base class for items which can be named, deleted, copied, etc. as in the editor of a 3d modeler.
- **ChPhysicsItem**: Base class for items that can contain objects of ChLcpVariables or ChLcpConstraints, such as rigid bodies, mechanical joints, etc.
- **ChIndexedNodes**: Interface class for clusters of points that can be accessed with an index. Must be inherited by children classes.
- **ChMesh**: Class which defines a mesh of finite elements.

```cpp
// std::vector<ChNodeFEMbase*> vnodes; // nodes
// std::vector<ChElementBase*> velements; // elements
LCP SYSTEM FUNCTIONS for interfacing all elements with LCP solver
```
Chrono::Engine FEM classes:

Integration.

Stiffness Matrix

\[ [K] = \int_V [b]^T [E] [b] dV \]

System

\[ \{f\} = [K] \{s\} \]

Strain

\[ \{\varepsilon\} = [b] \{s\} \]

Stress

\[ \{\sigma\} = [E] \{\varepsilon\} \]

ChGaussPoint

Class for a gauss point, that has a position (1D-3D) and a weight. It also contains the strain and the stress tensor.

ChIntegrationRule

General class for integration rules.

ChGaussIntegrationRule

Class for the management of the Gauss Quadrature in 1D, 2D or 3D space. Integration is done over the canonical interval \((-1, ..., +1\)), so the position of each gauss point is expressed in terms of natural coordinates.
Chrono::Engine FEM classes:

The importance of integration.
#include "physics/CHapidll.h"
#include "physics/CHsystem.h"
#include "physics/CHnodeBody.h"
#include "lcp/CHlcpIterativePMINRES.h"
#include "fem/CHelementHexa_20.h"
#include "fem/CHmesh.h"

GetLog() << "TEST: QUADRATIC hexahedral element FEM \n\n";

// The physical system: it contains all physical objects.
ChSystem my_system;

// Create a mesh, that is a container for groups
// of elements and their referenced nodes.
ChSharedPtr<ChMesh> my_mesh(new ChMesh);

// Create a material, that must be assigned to each element,
// and set its parameters
ChSharedPtr<ChContinuumElastic> mmaterial(new ChContinuumElastic);
mmaterial->Set_E(207e6);
mmaterial->Set_v(0.3);

// Create some nodes. These are the classical point-like
// nodes with x,y,z degrees of freedom, that can be used
// for many types of FEM elements in space.
ChNodeFEMxyz mnode1(ChVector<>(0,0,0));
ChNodeFEMxyz mnode2(ChVector<>(0,0,0.001));

mnode1.SetMass(0.001);
mnode2.SetMass(0.001);

// Remember to add nodes and elements to the mesh!
my_mesh->AddNode(mnode1);
my_mesh->AddNode(mnode2);

// For example, set an applied force to a node:
ChVector<> mnode7.SetForce(0, -1000, 0);

// Create the tetrahedron element, and assign
// it nodes and material
ChElementHexa_20 melement1;
melement1.SetNodes(&mnode1, &mnode2, &mnode3, &mnode4, &mnode5, &mnode6,
 &mnode7, &mnode8, &mnode9, &mnode10, &mnode11, &mnode12, &mnode13,
 &mnode14, &mnode15, &mnode16, &mnode17, &mnode18, &mnode19, &mnode20);
melement1.SetMaterial(mmaterial);

// Use this statement to use the reduced integration
// Default number of gauss point: 27. Reduced integration -> 8 Gp.
melement1.SetReducedIntegrationRule();

// Remember to add elements to the mesh!
my_mesh->AddElement(melement1);

Now your mesh contains a list of all nodes and elements created.
Chrono::Engine FEM:
How to use it.

The mesh must be added to the system

// This is necessary in order to precompute the
// stiffness matrices for all inserted elements in mesh
my_mesh->SetupInitial();

// Remember to add the mesh to the system!
my_system.Add(my_mesh);

Now you have to create the constraints

// Create a truss
ChSharedPtr<ChBody> truss(new ChBody);
my_system.Add(truss);
truss->SetBodyFixed(true);

// Create a constraint between a node and the truss
ChSharedPtr<ChNodeBody> constraint1(new ChNodeBody);
constraint1->Initialize(my_mesh, // node container
0, // index of node in node container
truss); // body to be connected to

ChSharedPtr<ChNodeBody> constraint2(new ChNodeBody);
constraint2->Initialize(my_mesh, // node container
1, // index of node in node container
truss); // body to be connected to

..........

Constraints must be added to the system

my_system.Add(constraint1);
my_system.Add(constraint2);
..........

Your analysis is ready to be executed

// Perform a linear static analysis
my_system.SetLcpSolverType(ChSystem::LCP_ITERATIVE_PMINRES);
// ^ NEEDED ^ because other solvers can't handle stiffness matrices

chrono::ChLcpIterativePMINRES* msolver =
(chrono::ChLcpIterativePMINRES*)my_system.GetLcpSolverSpeed();

msolver->SetDiagonalPreconditioning(true);
msolver->SetVerbose(true);
my_system.SetIterLCPPmaxItersSpeed(100);
my_system.SetTolSpeeds(1e-12);

my_system.DoStaticLinear();

To compute strain and stress

melement1.GetStrain();
melement1.GetStress();
Example: the FEM technology for finite elements

TEST: QUADRATIC hexahedral element FEM

----- Projected MINRES -supporting stiffness-, n.vars nx=72 max.iters=100 -----

-1.23804e-010
-8.45919e-011
-1.42096e-010
1.31079e-011
-1.20064e-010
0.001
0.001
-1.36752e-010
0.001
0.001
-9.37750e-011
-5.78011e-011
Node7 displ: 
14.4218
-32.5601
14.4218

Strain at Integration Point 3:
Matrix 6 rows, 1 columns.
2143.36
-3799.73
-489.605
-4380.91
2733.81

Stress at Integration Point 3:
Matrix 6 rows, 1 columns.
2.15581e+009
-2.96655e+008
-7.38738e+009
-3.89801e+008
-3.07388e+009
2.17654e+009

Premere un tasto per continuare . . .
void ChMesh::LoadMesh(char *file_path,
   ChSharedPtr<ChContinuumElastic> mmaterial);

This function allow you to read mesh from ASCII files, such as .vol (Netgen) or .geo (Gmsh) files, using the Chrono built-in interface for Input-Output (Chstream.h).

...............std::list<ChNodeFEMxyz>* NodeList;

for(int nodecounter=0; nodecounter<NodeNum; nodecounter++){
   *myfilei >> pos1 >> pos2 >> pos3;
   ChNodeFEMxyz* node = new ChNodeFEMxyz();
   node->SetPos(ChVector<>(pos1/1000, pos2/1000, pos3/1000)); //mm->m
   node->SetMass(0.01);
   node->ID = nodecounter+1;
   vnodes.push_back(node);
   this->AddNode(*node);
}...............
Chrono::Engine FEM: Performance.

- Read mesh
- Ste up the mesh:
  - Stiffness matrices of elements
  - Ndof of nodes
- Add constraints
- Execute solver

--- LINEAR ---
- nodes: 443
- elements: 1301
- 0.09s
- ~ 0.03%

--- QUADRATIC ---
- nodes: 2571
- elements: 1301
- 1.93s
- ~ 0.11%

--- LINEAR ---
- 263s

--- QUADRATIC ---
- 1691s
Questions?