

INTEGRATING GiD WITH IMPACT

Jonas Forssell

SimTool

Kallhedsängen 7

423 63, Torslanda, Sweden

e-mail: jonasforssell@yahoo.se , web page: <http://impact.sourceforge.net/>

Key words: Explicit Finite Element, Dynamic, Integration, Interface, Impact, GiD

Abstract. *Impact is an Explicit Finite Element Program which solves dynamic problems often involving large displacements. Even though the structure and object orientation of the program allows it to be flexible in terms of interfacing with different pre- and post processors, Impact was from the start written to take advantage of GiD. This paper describes the preprocessor interface configuration of GiD together with how Impact reads and parses the indata. The printing of the results and how GiD can present them to the user is also discussed. Integration of GiD and Impact has been proven to work well on several different platforms despite the fact that Impact still is under heavy development. The paper concludes with proposals for further work on the interface as well as Impact itself.*

1 INTRODUCTION

There is a great variety of traditional small displacement implicit finite element programs available on the Internet. In the area of explicit dynamic solvers however, the offering is limited if one excludes the commercial programs. With this background, the author initiated work on Impact and with the help of sourceforge.net, the project was placed on the Internet under the Gnu Public License (similar to Linux operating system).

The first version 0.0.52 was released in March 2002 and with the help of contributors the program quickly grew to the current state of version 0.4.1. The first pre-processor interface was written by Ruediger Heim and was later extended by the author to support more recent features of Impact.

The goal for Impact is to be a viable alternative to the commercial offerings but also to be simple in design and easy to use. Therefore, a new input format called Fembic was developed and the data format used by GiD was chosen for the result data.

Impact is still in Alpha stage which means that the results are not verified in all areas and new features are still being added to the program.

2 DESCRIPTION OF IMPACT

Impact can solve most dynamic problems such as crash deformations, metal sheet forming or spring/mass problems. At its present state though, the program is only suitable for component models due to speed and memory issues have not been optimised yet.

2.1 Theory

In a simplified way, Impact is build around solving a dynamic force equilibrium. Equation 1 shows the equation for an system without damping as described by Cook et al¹. The left term is the displacement to be solved for, with the first right term being the external forces minus the internal forces generated from the elements itself via the element stiffness matrix. Finally, the dynamic contribution from the node displacement and velocities are added. All the mass is lumped in the nodes.

$$\frac{1}{\Delta t^2} \{M\} \{D\}_{n+1} = \{F\}_n^{\text{ext}} - \{K\} \{D\}_n + \frac{1}{\Delta t^2} \{M\} \left[\{D\}_n + \Delta t \dot{D}_{n-\frac{1}{2}} \right] \quad (1)$$

An explicit code relies on the velocities being calculated a half timestep shifted from the displacements and accelerations. In other words, the code calculates the next step by using the values of previous steps. As long as the steps are smaller than the critical timestep, the solution converges. The critical timestep for an element is determined by the element size and material properties. The global timestep is set by the smallest critical timestep of all the elements in the model.

2.2 Features

Impact includes a range of features as shown in fig 1. The contact algorithm is based on micro meshes where the contact elements are included into the other elements in various ways to allow contact detection.

Boundary conditions can be applied in local directions and also include rigid bodies. Time step size can be defined by the user or can be automatically optimized by the program. Output formats are selectable from within the indata file itself.

<u>Elements</u>	1d: Rod element; 6-deg spring element with damping 2d: C0 Triangular shell; BLT Quad shell; 1-5 integration points through thickness 3d: Solid isoparametric hexa with 1 or 8 integration points
<u>Contact</u>	Contact line and triangle elements Friction
<u>Load and BC</u>	Variable pressure and node load Variable boundary condition on nodes Local axis defineable for load and boundary conditions Rigid body between one or more nodes
<u>Material</u>	Elastic and Elastoplastic isoparametric material model Strain rate effect
<u>Input</u>	Fembic indata format with flexible syntax
<u>Output</u>	Default output to .flavia.res and .msh Trackers to read and plot curve data
<u>Control</u>	Automatic or manual time step Large strain thickness reduction on shell elements and rod Hourglass control for shell elements

Figure 1: Impact features

3 THE PRE-PROCESS USER INTERFACE

Impact comes with a set of configuration files for GiD. These are just to be copied to the GiD directory for setup and GiD then needs to be restarted. The menu is now available under the Data -> Problem type menu.

3.1 Defining the materials

The material dialog currently features only elastic and elastoplastic materials. A default of five different types are available from start which can be assigned to the geometry.

3.2 Defining the conditions

Conditions are used for defining loads and boundary conditions including rigid bodies. By specifying several values within brackets according to Fembic syntax, the values can be made to vary over time. To aid the user, helping text is shown next to the fields. This text is actually a picture. Using conditions allows both loads and boundary conditions to be applied to a single node. It also fits well with the Fembic syntax. Unfortunately, conditions cannot be saved in GiD version 7, forcing them currently to be redefined at each change of value.

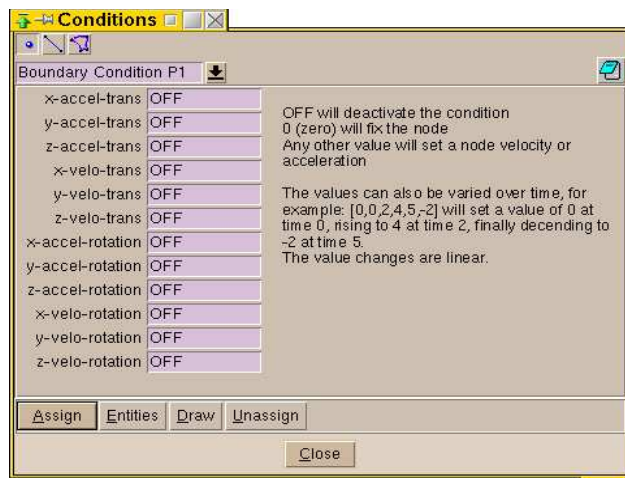


Figure 2: condition menu example

3.3 Defining general parameters

Time control, step size, print intervals and hourglass control parameters are set in separate general menus. The hourglass control values are applied to all shell elements.

4 DESIGN OF THE INTERFACE

The syntax of a typical Fembic² file is shown in figure 3. The file is divided in blocks, each starting with a key word. The blocks may be repeated and come in any order. Some cross references between the blocks are shown and the data related to an element or node is defined together with that node for

```
Nodes
1 x = 1.1 y = 0.0 z = 0.0 load = ld1
2 x = 2.1 y = 0.0 z = 0.0 constraint = fix

Elements of type rod_2
1 nodes = [1,2] D = 5.0 material = steel

Materials of type elastic
steel rho = 7.8E-6 E = 210 nu = 0.3

Constraints of type boundary_condition
fix ax=0 ay=0 az=0 vx=0 vy=0 vz=0

Loads
ld1 Fx = 0.1

Trackers of type rodforce
1 element = [1] filename = rod_force.trk

Controls
run from 0 to 20 step 0.0001
print every 0.1 step
print tracker every 0.01 step
```

Figure 3: Fembic example file

maximum simplicity. Property cards, common in other codes are not used here.

4.1 The .cnd file

As mentioned earlier, the conditions are used for both loads and boundary conditions. To be able to separate them, hidden id fields (ID ,ID2) are used in the .cnd file. These are pre-set with values depending on what type of condition or load is applied. An example is shown in figure 4, for a rigid body boundary condition.

```

NUMBER: 52 CONDITION: Rigid_Body_RB2
CONDTYPE: over points
CONDMESHTYPE: over nodes
IMAGE: rb.gif
QUESTION: ID
VALUE: 52
STATE: HIDDEN
QUESTION: ID2
VALUE: 0
STATE: HIDDEN
QUESTION: Master-Node-number
VALUE: 0
END CONDITION

```

Figure 4: .cnd file extract

4.2 The .bas file

This file defines the main part of the Fembic file. We will now look at how the definition of the node with attached parameters are defined as shown in figure 5.

```

NODES
*Set cond Boundary_Condition_P1 *nodes *or
(2,int)
....
*Add cond Boundary_Condition_L1 *nodes *or
(2,int)
....
*Add cond Boundary_Condition_S1 *nodes *or
(2,int)
....
*Add cond Rigid_Body_RB1 *nodes *or(2,int)
....
*Add cond Force_Load_P1 *nodes *or(1,int)
....
*Add cond Dummy *nodes *or(1,int) *or(2,int)
*loop nodes
*format "%8i x = %15.8f y = %15.8f z = %
15.8f"
*NodesNum*NodesCoord(1)*NodesCoord(2)
*NodesCoord(3) *\
*if(cond(1,int) > 0)
*format "%1i"
constraint = Constr_*cond(1,int) *\
*endif
*if(cond(2,int) > 0)
*format "%1i"
load = Load_*cond(2,int)
*else
*endif
*endif
*end nodes

```

Nodes
1 x = 1.1 y = 0.0 z = 0.0 load = Load_1
2 x = 2.1 y = 0.0 z = 0.0 constraint = Constr_1
3 x = 1.1 y = 1.2 z = 0.0 constraint = Constr_1

Elements of type shell_C0_3
1 nodes = [1,2,3] t = 5.0 material = mat_1

Materials of type elastic
mat_1 E = 210 rho = 7.8E-6 nu = 0.3

Constraints of type boundary_condition
Constr_1 ax=0 ay=0 az=0 vx=0 vy=0 vz=0

Loads
Load_1 Fx = 0.1 Fy = 0 *abbreviated*

Trackers of type rodforce
1 element = [1] filename = rod_force.trk

Controls
run from 0 to 20 step 0.0001
print every 0.1 step
print tracker every 0.01 step

Figure 5: Definition of nodes

All conditions are included in the database. Note the including of a dummy condition which is needed for the scan to work properly. Next, all nodes are looped and the coordinate for each node is printed by default. First hidden ID field is checked and if there is a condition defined, it is printed with the ID as the variable. Second hidden ID is also checked for the loads and printed along the same principle.

4 PARSING AND SOLVING

The Impact process of parsing and solving is summarised in figure 6.

All data is stored within the element and node objects which in turn are stored in large arrays within the smack object.

4.1 Structure

To keep data as isolated as possible in good object orientation practice, the parsing of the indata is mainly done by the elements themselves. The Fembic object (created initially), reads through the fembic file, creates new element objects and feeds them their indata line just like it is written in the indata file. The element object then parses that line itself and creates all the variables it needs for private storage.

Since Impact should be able to handle different indata file formats, each element must be prepared to parse different indata strings. This is handled by separate methods, one for each file format as shown in figure 7.

4.2 Writing

The same process is used for writing the results into GiD .flavia result format. The GiD writer object handles the writing process, calling each node or element object in turn, asking them to write their data. The method `print_Gid` is used with a passed flag so that the element knows what to print. The resulting string given back by the element is then inserted properly into the .flavia.res file by the GiD writer object.

- Init
 - Create a Fembic instance
- Parse
 - Ask F for number of elements etc.
 - Set up arrays
 - Ask F to generate nodes, elements, writer, Controlset, Constraints etc.
- Setup Mass
 - Calculate mass matrix etc.
- Set initial conditions
- Solve
 - Loop through time
 - Loop through all elements and calculate forces
 - Remove failed elements
 - Loop constraints and update
 - Loop nodes and calculate new positions
 - Loop trackers and let them print if needed
 - Update timestep
 - Print result data if needed by ordering Writer
 - Loop nodes and clear forces
 - End loop time
- Post

Figure 6: Parsing and solve process

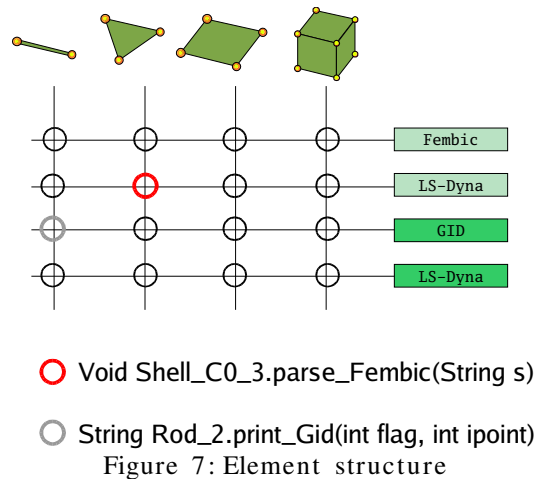


Figure 7: Element structure

5 POST-PROCESSING

The process of viewing the results is straightforward. The .flavia.res file is read by the user and in order to get an animation as an end result, the result menu must first be set up followed by the mesh deformation. Most of the time, the mesh deformation requires a factor of 1 which means a change of the default value.

5.2 Results

Impact currently supports strains and stresses for color plots. Both in global and local coordinate system on those elements that supports it. GiD currently only supports one result set which means that some shell elements which have several integration points through the thickness must preselect which one to print. This is done in the Fembic file already before the solution starts.

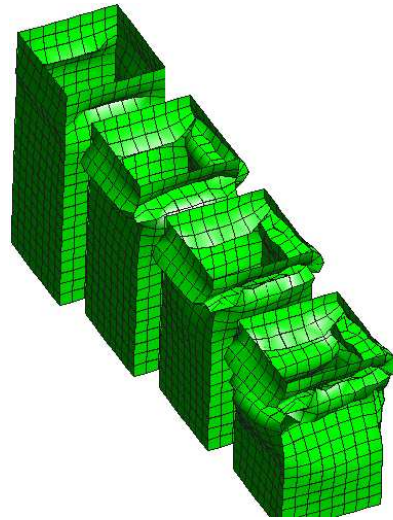


Figure 8: Beam model

5.3 Animation

Animation works very well with the possible exception that rotation and zooming cannot be performed during animation. The export possibility of the animation to different file formats is very powerful for presentation in other forums.

5.4 Trackers

Trackers is a feature that extends the result viewing possibility with GiD. It is a feature which specifically tracks a model node or element and plots the result as a function plot. The filename is decided by the user together with the rest of the tracker parameters in the Fembic file. This file can then be read by GiD and plotted as a curve as shown in figure 9. In this specific case, it is the total reaction load of the beam in figure 8, as a function of time.

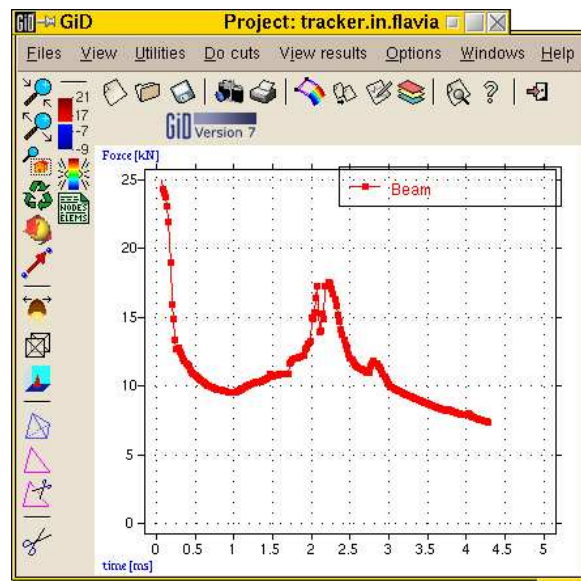


Figure 9: Tracker data plotted in GiD

6 FURTHER WORK

Impact is under heavy development up to release 1.0. This means there is much improvement to be done on both the program and the interface with GiD.

6.1 Pre- process interface

The interface is not optimal at this time. The constraints cannot be saved which means the user must redefine them when switching to another condition. There is also no feedback to the user on which geometry which condition is defined respectively.

All of the supported elements in Impact must be available as well as all eight tracker types which are now missing completely. Some more general options need also to be available.

6.2 Impact

Impact itself is being developed along a masterplan. Current development is focused on distributed solution capability on LAN and Internet. Some more elements will be added such as tetrahedron, beam and wedge elements. Optimisation in speed and memory usage and detailed verification and debugging of the program will complete the development for version 1.0 which will be considered the first stable release.

7 CONCLUSIONS

- GiD has proven to be a well performing choice of pre- and post processor to the explicit dynamic finite element code Impact.
- The configuraton of Impact has been straightforward with a heavy reliance on the definition of conditions for both loads and boundary conditions.
- Impacts modularity allows various input and output formats but GiD was chosen as the processor of choice due to the academic license, simple file format and multi platform support.
- Both Impact and interface required further development. Proposals for improvement of GiD was made.

REFERENCES

[1] Concepts and Applications of Finite Element Analysis, Robert D Cook. David S. Malkus, Michael E. Plesha, 3:rd edition, ISBN 0- 471- 84788- 7

[2] Impact users and programmers manual, Version 0.4.1,
<http://Impact.sourceforge.net>