Finite Element Modeling of Contact and Impact Problems Using Python

Ryan Krauss (rkrauss@siue.edu) – *Southern Illinois University Edwardsville,* USA

This paper discusses an on going project to improve the accuracy of automotive crash simulations. Two likely causes for discrepancy between simulations and the results of real, physical tests are discussed. An existing Python package for finite element analysis, SfePy, is presented along with plans to contribute additional features in support of this work, including nonlinear material modeling and contact between two bodies.

Background and Motivation

Introduction

Automobile crashes kill 30,000 Americans in an average year [Kahane]. Motor vehicle crashes are the leading cause of death in the U.S. for the age group 4-34 [Subramanian]. Some of these deaths might be preventable. Engineers can save lives through the design of safer automobiles.

Crashworthiness design is a complicated process [Bellora] in which many important safety-related decisions must be made before realistic tests can be run. As a result, reliable virtual tests or simulations are essential, so that early design decisions can be data driven and safety countermeasures can be correctly designed.

Problem

Unfortunately, simulations are often not as accurate as they need to be. This can lead to failure of physical tests late in the design of a vehicle. Fixing such failures can be very expensive and extremely challenging. If the causes of the differences between simulation and experiment can be identified and removed, simulations could provide a true virtual test environment and these late and expensive failures could be avoided.

Two likely sources of the discrepancies between simulation and experiment include determining high speed material properties and correctly modeling contact between two bodies during simulation.

Long Term Goal

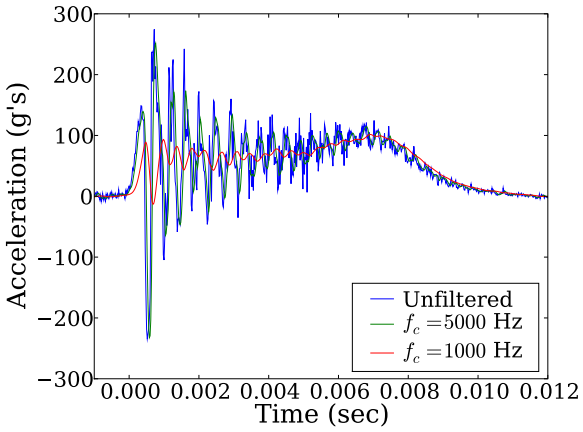
The long term goal of this research project is to remove these two obstacles to more reliable virtual tests. This will include designing a test device for impact testing of material samples. This device must be very stiff so that if its natural frequencies are excited during the impact test, the ringing can be filtered out of the data without contaminating the data itself.

The second step in achieving the long term goal of a reliable virtual test environment is creating Python based simulation software for this work, based on the SfePy [SfePy] finite element analysis package. A method for estimating material properties and generating deformation models will be developed based on the combination of simulation and experiment.

Designing a Test Device

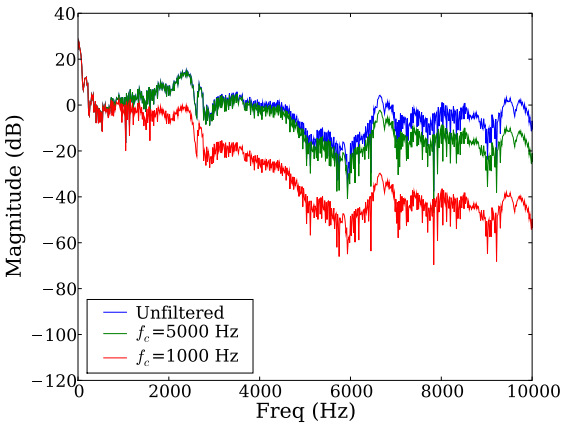
Many existing devices for high-speed materials testing contaminate the data with ringing or vibrations when used for impact testing. It is difficult to design a device with sufficiently high natural frequencies so that any ringing can be filter out without significantly altering the material response data [Maier].

The next graph shows time domain data from a device with significant ringing problems.



Impact tests results from a device with significant ringing problems

The following graph shows the fast Fourier transform (FFT) of this same data.



FFT of the data in the preceding figure

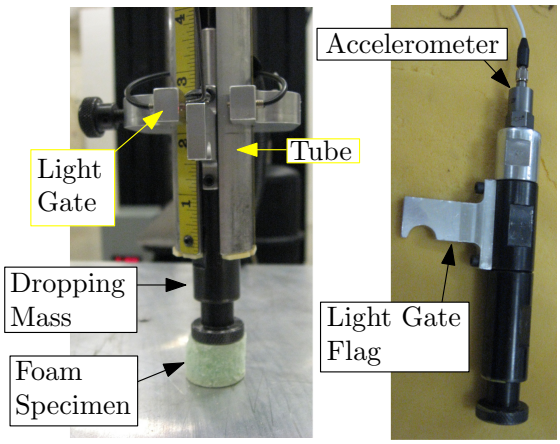
The ringing in the data shown above is not well separated in frequency from the material response data and it is difficult to filter out the ringing without altering the slope on the leading edge of the impulse.

This slope is directly related to estimates of material properties such as Young's Modulus.

Small Drop Tester

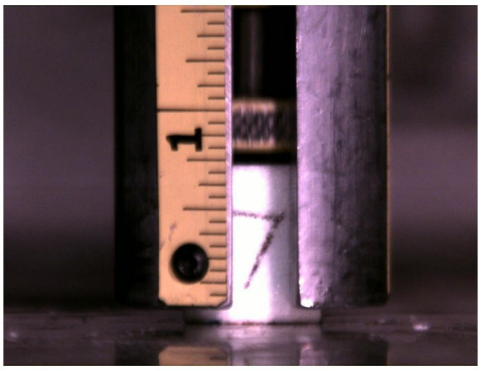
Several different impact test devices were investigated as part of this research project in the summer of 2007. The amplitude of ringing excited in a typical impact test was compared along with the separation in frequency between the ringing and the rest of the data. One device stood out: its natural frequencies were significantly higher than the rest, because it depends on a fairly small mass being dropped onto the material sample.

The device is shown in the next figure. A test consists of a small mass being dropped onto polyurethane foam samples. The mass is dropped inside a tube that is used to guide it while it is falling and to keep it vertical. The dropping mass has an accelerometer on the back of it to measure the force being exerted on the mass by the foam sample. There is also a metal flag on the side of the mass used to measure the impact velocity with a light gate.



Small mass drop tester

A picture from just before the dropping mass impacts a foam sample is shown in the next picture.

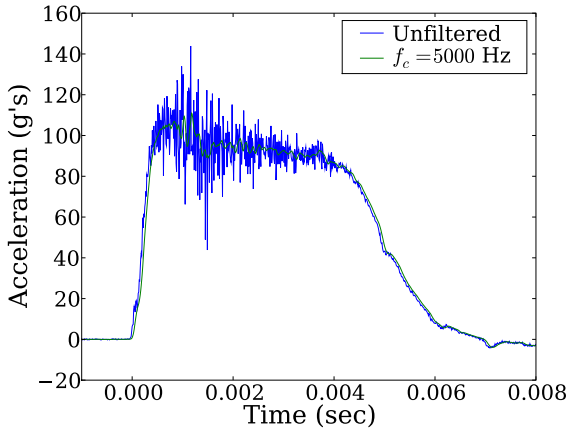


The small mass drop tester just before impact

The edges of the dropping mass were painted yellow to increase visibility in the high speed video. The white cube with the number 7 on it is the foam sample. A video clip can be seen here: http://www.siue.edu/~rkrauss/sfepy/output.avi

Example Data

An example of data from the small mass drop tester is shown below. Note that the ringing in this data is at a very high frequency and it has been largely filtered out without significantly altering the slope of the leading edge of the impulse.



Impact tests results from a device whose ringing is of a high enough frequency to be filtered without contaminating the data

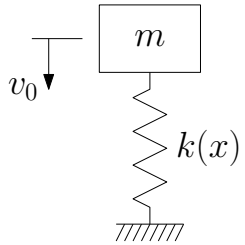
Python Based Simulations

Along with designing an impact test device that does not contaminate the data with ringing, the other goal of this project is to develop Python software modules for simulating impact tests.

The initial modeling goal is to be able to accurately simulate the small mass drop tester impacting polyurethane foam samples. This will be pursued for two reasons. First, it is necessary in order to extract the parameters for a model of the impact response of the foam. Second, it is a good candidate for an initial stepping stone toward modeling more complicated scenarios, eventually moving toward simulating in-vehicle impacts with a dummy head.

A Simple Model

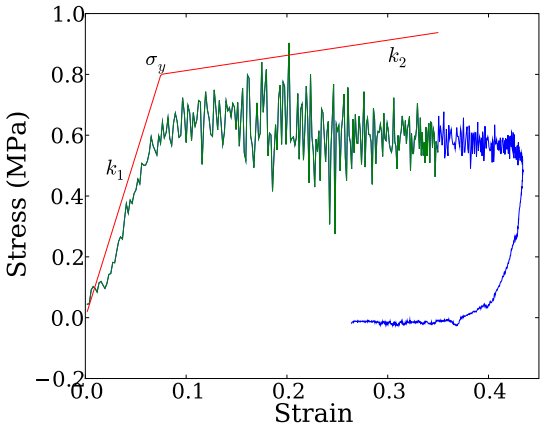
The first step in Python based simulations of the foam impact tests was to use a really simple model: a mass with an initial velocity compressing a nonlinear spring as shown below:



A simple model of the foam impact test

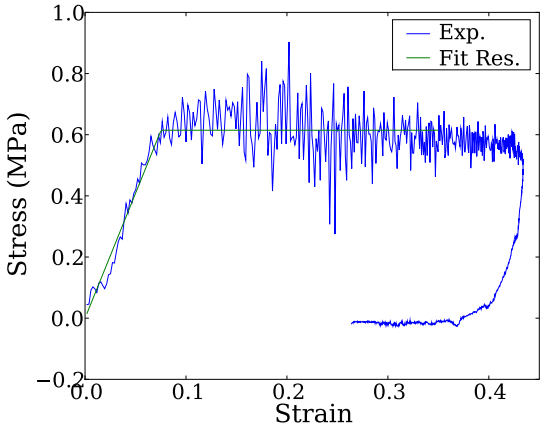
The spring was modeled as bi-linear: its force/deflection curve has a linear elastic region near

the origin and then a knee representing yielding. Above the knee, the force deflection curve is still linear, but with a different slope. The corresponding stress/strain curve can be found by dividing the force by the cross-sectional area of the foam sample and the displacement by the initial height. An example stress/strain curve is shown below. The data was curve-fit to find k_1, k_2, and σ_y using the Nelder-Mead simplex algorithm of Scipy's optimize.fmin.



Bi-linear curve fit setup

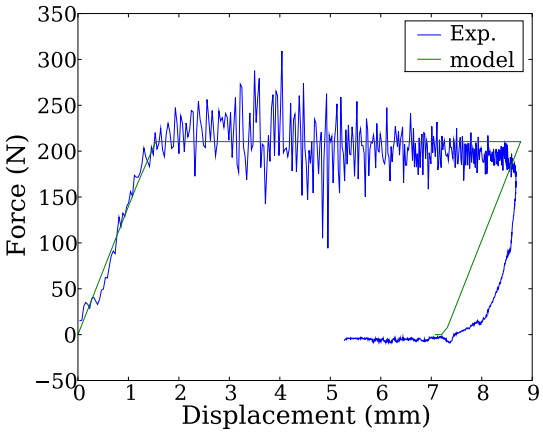
The results of the curve-fit are shown in the next graph.



Curve fit results

Model Predictions

The estimates for k_1, k_2, and σ_y were then used in an ODE model of the simple system. The resulting force vs. deflection curve is shown in the next graph. Note that the model is a fairly good fit, in spite of it being such a simple model. The primary deficiency of the model is that it does not fit the back of the force/deflection curve very well (during the rebound portion of the curve when the mass is bouncing back up).



Model predictions vs. experiment

Finite Element Modeling

While the simple model does a decent job, the accuracy of the simulation needs to be improved, especially in the rebound portion. A model is needed that accounts for the fact that the foam does not all displace uniformly. The foam is continuous and the displacement within it can vary with position (i.e. the x, y, and z coordinates of a point within the foam) as well as with time. This takes the model into the realm of partial differential equations and finite element analysis (FEA).

FEA is an approach to modeling a real, physical problem with a collection of small elements. These small elements are used to discretize a continuous problem. Ultimately, a partial differential equation model is converted to a matrix expression such as

$$[\mathbf{K}]\{\mathbf{D}\} - \{\mathbf{R}\} = 0 \qquad (1)$$

where $[\mathbf{K}]$ is the stiffness matrix, $\{\mathbf{R}\}$ is the forcing vector, and $\{\mathbf{D}\}$ is the vector of nodal displacements.

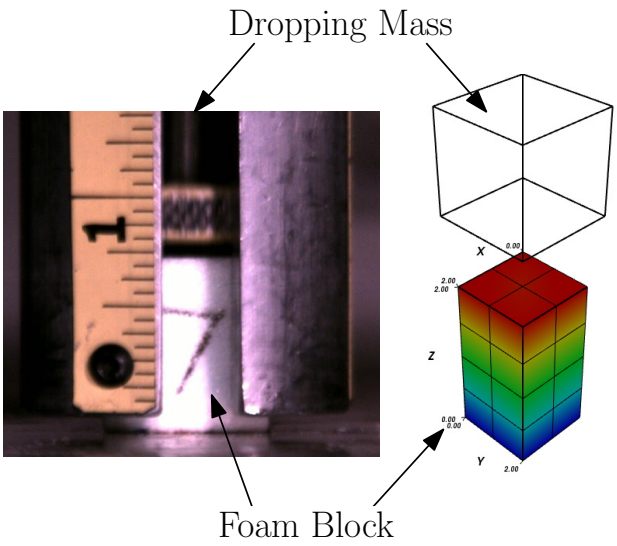
Problems with commercial FEA software

While the FEA portion of this project may be considered solved by some, the correlation between FEA simulations and the results of real, physical tests is often not as good as it should be. Most researchers are satisfied as long as key measures between simulation and experiment agree, even if the curves do not overlay closely [Rathi]. Closed-source software impedes research into the causes of these discrepancies. As such, this project seeks to add features to an existing Python FEA tool called SfePy [SfePy] so that it can fully solve this problem.

Applying FEA to this Problem

In order to apply FEA to this problem, the foam block will be discretized into many elements using a mesh. The dropping mass can be represented by one large, rigid element. The relationship between the FEA

model and the physical system is shown in the next figure.



FEA setup for this problem

Introduction to SfePy

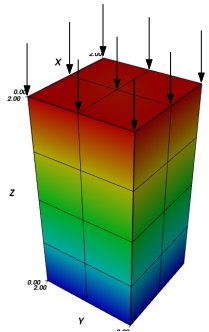
SfePy stands for Simple Finite Elements for Python. It is an FEA solver written primarily by Robert Cimrman. Along with the solver it provides a fairly high-level syntax for problem description that serves as a sort of pre-processor. SfePy also includes mesh generation capabilities for simple geometries. Post-processing must be done by some other software such as Paraview or Mayavi2.

SfePy can handle nonlinear and time varying problems and it provides solvers for linear elasticity, acoustic band gaps, Poisson's equation, and simple Navier-Stokes problems.

This project aims to contribute to SfePy capabilities for modeling nonlinear materials, multi-body dynamics, and contact between colliding bodies.

Initial FEA Model

The initial FEA model of this problem included only the foam with the interaction force between the foam and the dropping mass modeled as a traction load spread out over the top surface of the foam as shown in the next figure.



Initial FEA setup with a traction load across the top surface of the foam

Initial Input Script

Highlights of the input to SfePy are shown below. First, define the mesh file

```
filename_mesh = '3D_mesh.vtk'
```

Then label important regions of the mesh:

```
region_1 = {
    'name' : 'Bottom',
    'select' : 'nodes in (z < 0.001)'
}
region_2 = {
    'name' : 'Top',
    'select' : 'nodes in (z > 1.999)',
}
```

and use those defined regions to specify boundary conditions:

```
ebc_1 = {
    'name' : 'fixed_u',
    'region' : 'Bottom',
    'dofs' : {'u.all' : 0.0},
}
```

Material properties can be defined like this:

```
material_1 = {
    'name' : 'solid',
    'mode' : 'here',
    'region' : 'Omega',
    # Lame coefficients:
    'lame' : {'lambda' : lamb, 'mu' : mu},
}
```

Understanding FEA/SfePy

FEA is a mathematical tool for solving partial differential equations (PDEs). In the context of this problem, that means that the stress (force per unit area) and strain (change in height divided by initial height) in the foam sample are functions of more than one variable: $\sigma(x, y, z, t)$ and $\varepsilon(x, y, z, t)$.
SfePy uses the weak formulation for finite element problems. This means that the problem is stated in terms of an integral expression that is valid over a volume. For the case of solid mechanics, this integral is the potential energy functional:

$$\Pi_p = \int \{\sigma\}^T \{\mathbf{d}\varepsilon\} - \int \{\mathbf{u}\}^T \{\mathbf{\Phi}\}\, dS \qquad (2)$$

where $\{\sigma\}$ is stress, $\{\varepsilon\}$ is strain, $\{\mathbf{u}\}$ is displacement, and $\{\mathbf{\Phi}\}$ is a traction vector representing force distributed over a surface S.
The derivation in this section is based on [Cook], especially chapters 3 and 4. Discretizing and replacing the integral over the entire volume with the sum of integrals over the finite elements gives:

$$\Pi_p = \sum_{i=1}^{N_{els}} \int \int \{\sigma\}^T \{\mathbf{d}\varepsilon\}\, dV - \sum_{i=1}^{N_{els}} \int \{\mathbf{u}\}^T \{\mathbf{\Phi}\}\, dS \qquad (3)$$

Substituting a linear elastic material model

$$\{\sigma\} = [\mathbf{E}]\{\varepsilon\} \qquad (4)$$

(where $[\mathbf{E}]$ is a matrix expressing Hooke's law in three dimensions) results in

$$\int \{\sigma\}^T \{\mathbf{d}\varepsilon\} = \frac{1}{2} \{\varepsilon\}^T [\mathbf{E}] \{\varepsilon\} \tag{5}$$

and

$$\Pi_p = \sum_{i=1}^{N_{els}} \int \frac{1}{2} \{\varepsilon\}^T [\mathbf{E}] \{\varepsilon\} \, dV - \sum_{i=1}^{N_{els}} \int \{\mathbf{u}\}^T \{\mathbf{\Phi}\} \, dS \tag{6}$$

Defining the matrix $[\partial]$ for the relationship between strain ε and displacement

$$\{\varepsilon\} = [\partial] \{\mathbf{u}\} \tag{7}$$

and the matrix $[\mathbf{N}]$ for interpolation from the displacement of the corners of a brick element $\{\mathbf{d}\}$, to any point within it

$$\{\mathbf{u}\} = [\mathbf{N}] \{\mathbf{d}\} \tag{8}$$

allows the strain tensor to be written as

$$\{\varepsilon\} = [\partial] [\mathbf{N}] \{\mathbf{d}\} \text{ or } \{\varepsilon\} = [\mathbf{B}] \{\mathbf{d}\} \tag{9}$$

where $[\mathbf{B}] = [\partial] [\mathbf{N}]$.
Substituting this expression for $\{\varepsilon\}$ into equation 6 and integrating over each element produces

$$\Pi_p = \frac{1}{2} \sum_{i=1}^{N_{els}} \{\mathbf{d}\}^T [\mathbf{k}]_i \{\mathbf{d}\} - \sum_{i=1}^{N_{els}} \{\mathbf{d}\}^T \{\mathbf{r_e}\}_i \tag{10}$$

where

$$[\mathbf{k}]_i = \int [\mathbf{B}]^T [\mathbf{E}] [\mathbf{B}] \, dV \text{ and } \{\mathbf{r_e}\}_i = \int [\mathbf{N}]^T \{\mathbf{\Phi}\} \, dS \tag{11}$$

Defining a matrix $[\mathbf{L}]_i$ for each element that selects the element degrees of freedom from the global displacement vector $\{\mathbf{D}\}$

$$\{\mathbf{d}\}_i = [\mathbf{L}]_i \{\mathbf{D}\} \tag{12}$$

allows equation 10 to be rewritten as

$$\Pi_p = \frac{1}{2} \{\mathbf{D}\}^T [\mathbf{K}] \{\mathbf{D}\} - \{\mathbf{D}\}^T \{\mathbf{R}\} \tag{13}$$

where

$$[\mathbf{K}] = \sum_{i=1}^{N_{els}} [\mathbf{L}]_i^T [\mathbf{k}]_i [\mathbf{L}]_i \text{ and } \{\mathbf{R}\} = \sum_{i=1}^{N_{els}} [\mathbf{L}]_i^T \{\mathbf{r_e}\}_i \tag{14}$$

Rendering equation 13 stationary requires that

$$d\Pi_p = [\mathbf{K}] \{\mathbf{D}\} - \{\mathbf{R}\} = 0 \tag{15}$$

which is the final FEA matrix formulation that will be solved for the nodal displacement vector $\{\mathbf{D}\}$.
The interested reader is referred to [Cook] for a more thorough explanation.

References

[Kahane] Kahane, C. J., "Lives Saved by the Federal Motor Vehicle Safety Standards and Other Vehicle Safety Technologies, 1960-2002," Tech. rep., National Highway Traffic Safety Administration, Oct. 2004.

[Subramanian] Subramanian, R., "Motor Vehicle Traffic Crashes as a Leading Cause of Death in the United States, 2003," *Traffic Safety Facts, Research Note*, March 2006.

[Bellora] Bellora, V., Krauss, R., and Van Poolen, L., "Meeting interior head impact requirements: A basic scientific approach," *SAE 2001 World Congress & Exhibition, Session: Safety Test Methodology (Part C & D)*, Society of Automotive Engineers, Detroit, MI, March 2001.

[SfePy] Cimrman, R., 2008. SfePy Website. http://sfepy.kme.zcu.cz.

[Rathi] Rathi, K., Lin, T., and Mazur, D., "Evaluation of Different Countermeasures and Packaging Limits for the FMVSS201U," *SAE 2003 World Congress & Exhibition Session: Modeling of Plastic Foam and Cellular Materials for Crash Applications (Part 2 of 4)*, Society of Automotive Engineers, Detroit, MI, March 2003.

[Maier] Maier, M., Huber, U., Mkrtchyan, L., and Fremgen, C., "Recent improvements in experimental investigation and parameter fitting for cellular materials subjected to crash loads," *Composites Science and Technology*, Vol. 63, No. 14, 2003, pp. 2007-2012.

[Cook] Cook, R. D., Malkus, D. S., Plesha, M. E., and Witt, R. J., *Concepts and Applications of Finite Element Analysis*, 2002 John Wiley & Sons.