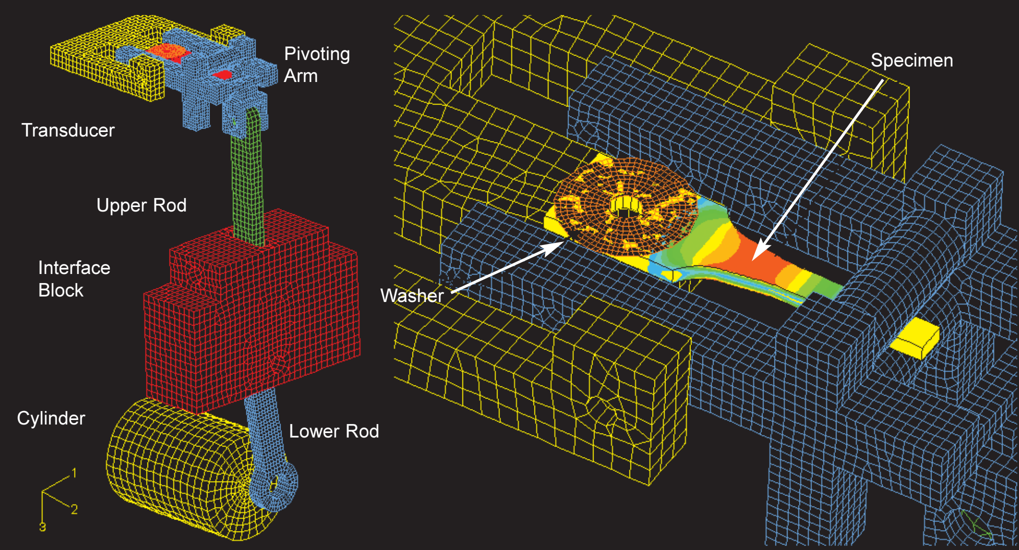
ABAQUS/*Answers*

ANSWERS TO COMMON ABAQUS QUESTIONS **FALL 2001**

CONTENTS



Three-dimensional model of mechanism and specimen.

Courtesy of Bayer Plastics.

ABAQUS

Hibbitt, Karlsson & Sorensen, Inc.

1080 Main Street
Pawtucket, RI
02860-4847 USA

PHONE 401.727.4200
FAX 401.727.4208
E-MAIL sales@abaqus.com
www.abaqus.com

Application of Connector Elements in ABAQUS/Standard

Introduction

This article discusses the use of connector elements in ABAQUS/Standard to simulate a machine mechanism. The analysis was performed by Bayer Plastics in Pittsburgh, PA, in the U.S. and determines the stress distribution in a material specimen subjected to cyclic loading. Figure 1 shows the laboratory testing equipment configuration.

Simulation Issues

The loading scenario of interest can be idealized as an end-loaded cantilever. The load is imparted to the specimen through rollers mounted on a crank mechanism. Initial assessments were based on a simple cantilever model that used oscillating concentrated forces to define the load. This method was found to yield inaccurate predictions when compared to the actual test data.

The main issue was identified as being the interaction of the rollers with the specimen during the loading cycle:

- The line of action of the load in the test procedure stays at a constant distance from the anchor point of the pivoting arm as the specimen bends. In contrast, the use of concentrated forces in the initial simulation requires the load to be defined at nodal locations that move with the specimen, thus changing the line of action of the load.
- The applied load switches between the upper and lower roller during the load cycle. This "switching" depends on the fit of the rollers with the specimen and the "springback" of the specimen during the load cycle.

Considering these two conditions, an accurate definition of the direct load was made impractical by the difficulty in fully understanding the load transfer. In addition, any changes in specimen material and size or alterations in the eccentricity of the crank mechanism would result in a different loading scenario.

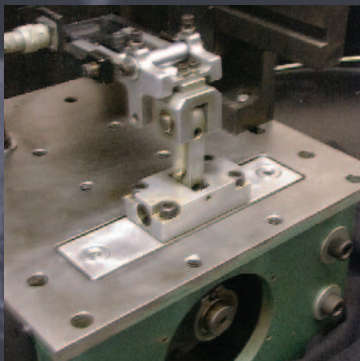


Figure 1: Fatigue testing machine.

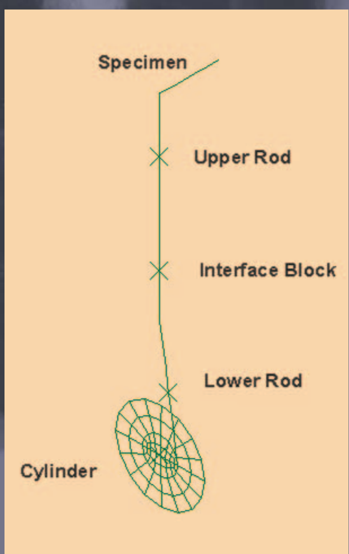


Figure 2: Simple mechanism model.

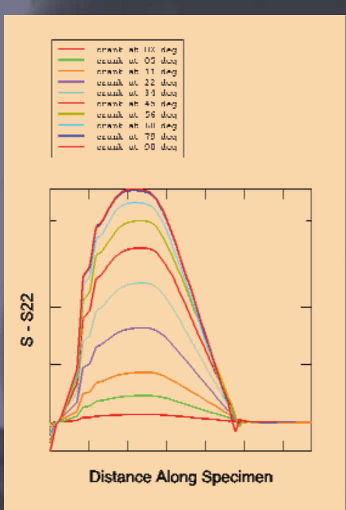


Figure 3: Path plot of S22 from 0 to full load.

Model Generation

It was decided to address this simulation by modeling not only the deformable specimen but also the machine geometry and connectivity. The connector elements in ABAQUS allow the use of this approach.

To develop the analysis methodology, a simplified representation of the system was generated (Figure 2). This simple model allowed the general response of the system to be understood and allowed the model to be debugged without a complex three-dimensional geometry. Representations of the cylinder, lower rod, interface block, upper rod, and deformable specimen were modeled using a simplified geometry. This file is available on the HKS web site (**`www.abaqus.com/answers_fall_2001/simple.inp`**).

The machine is constructed using rigid bodies; the connectivity of the mechanism is generated with connector elements. The specimen is modeled using hexahedral finite elements (C3D8I).

The machine mechanism is modeled in such a way that multiple specimens can be "tested." Physical adjustments to the machine setup (for example, crank eccentricity) can be incorporated using a parametric definition, which allows the base model to be used in a range of testing scenarios. These scenarios represent the machine adjustments made by the laboratory technicians.

The outer surface of the CAD geometry is meshed using rigid shell elements (R3D4, R3D3); connector elements (CONN3D2) of type HINGE are used to model the rigid body connections. The fatigue machine consists of seven rigid bodies: the cylinder, lower rod, interface block, upper rod, pivoting arm (including the rollers that are built into the arm), transducer, and washer. The motion is defined as a rotation of the cylinder, which drives the lower rod. The specimen itself is clamped into the main body of the machine, between a washer and a transducer, and interacts with rollers through contact conditions. Direct boundary conditions and constraints imposed by the connector elements define the following motion of the rigid parts: the cylinder can rotate only about the y-axis, the lower rod can move only in the x-z plane, the interface block can move only along the z-axis, the upper rod and pivoting arm can move only in the x-z plane, and the transducer and washer are fully constrained.

Results

The results of the simulation can be viewed dynamically as an animation (**`www.abaqus.com/answers_fall_2001/mechanism.avi`**). The animation shows the oscillatory nature of the stress magnitudes and their varying distribution. The location of the peak stress changes as the load is transferred from the rollers to the specimen. Path plots of maximum stress along the center of the specimen are shown in Figure 3. Using a simple end-loaded cantilever, the predictions of maximum stress and the stress distribution show significant variation from the full machine model.

Figure 4 shows the oscillatory motion of the interface block and a point on the bottom roller as well as the trajectory of the roller node. The first plot was produced by combining two X-Y data objects in ABAQUS/Viewer.

The horizontal and vertical motion of the rollers is responsible for the load switching between the lower and upper surfaces of the specimen. Figure 5 shows the contact pressure at the bottom face of the specimen at a representative time and its distribution along the path defined on the lower surface of the specimen as the crank rotates. Both the magnitude and the location of the peak stress change during the loading cycle, which illustrates the load transfer characteristics of the fatigue machine model. Using concentrated loads, the location of the applied load would remain constant.

The Future

The objective of this study was to incorporate the results from the ABAQUS/Standard simulation in the assessment of fatigue life and calculation of damage for a range of polymer specimens. The accurate assessment of the principal stress fields is critical for fatigue life assessment and damage predictions. Any variance in the stress predictions is exaggerated once an assessment of the fatigue life is made.

This work is ongoing but will allow the development and calibration of fatigue properties for a range of proprietary materials. The application of life calculations for plastics depends on capturing the behavior and calibrating the material properties. The test procedures discussed in this article are an integral part of the development of a knowledge base. Combining this knowledge with advanced numerical analysis techniques will increase the understanding of these concepts.

Scripting in ABAQUS: Reducing the Size of an Output Database

This article is the third in a series on the capabilities of the ABAQUS Scripting Interface. Refer to the online ABAQUS Scripting Manual for detailed information on how to use the ABAQUS Scripting Interface to create and analyze ABAQUS/CAE models, to view the results of an analysis, and to automate repetitive tasks. The manual also describes the syntax of each command used in this example.

An output database consists of model data and results data. Model data include the parts, assembly, part instances, sections, regions, and step definitions. Results data include the frames and field output that describe the results of the analysis. Results data can also include history data.

The analysis of a large, complex model can generate a large output database, especially if you accepted the default output requests. After you use ABAQUS/Viewer or ABAQUS/CAE to look at the contents of the output database, you might decide that you need only a fraction of the number of frames of results data and would like a technique to decrease the size of the output database.

This example shows you how to create a smaller output database that contains the model data along with only the frames of results data that you would like to archive. Since the contents of the ABAQUS output database cannot be deleted, you cannot simply remove unwanted frames of results data. However, you can take advantage of the fact that you can append data to the output database. You first create a new output database that contains only the model data by running a **datacheck** analysis on the original input file. You then extract selected frames of results data from the original output database and append them to the new output database.

In this example **`large`** is the name of the input file used to create the original output database (**`large.odb`**), and **`small`** is the name of the new output database that you will create. You first create a version of the **`small`** output database that contains only model data by running a **datacheck** analysis on the original input file as follows:

```
abaqus job=small datacheck input=large
```

You then run the script on Page 4 (**`copyOdbStep.py`**), which reads the **`large`** output database and appends results data from a selected step to the **`small`** output database, as follows:

```
abaqus python copyOdbStep.py small.odb large.odb
```

To minimize the data that are written to the output database, the script skips over frames during the step at a selected frequency. You can run the script again to append frames from a different step to the output database. After you append data to **`small.odb`**, you can open the output database with ABAQUS/Viewer or ABAQUS/CAE and verify the steps and frames that it contains.

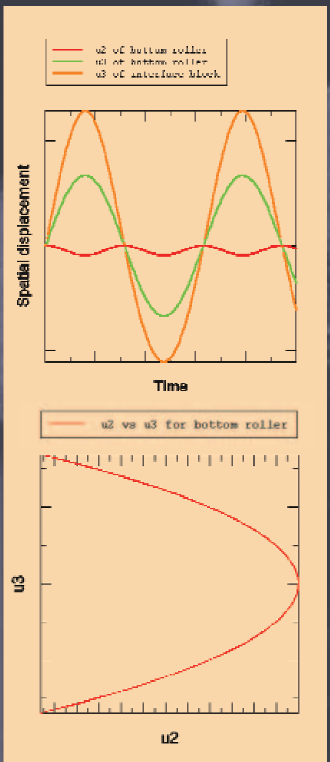


Figure 4: History plot of displacements of the interface block and the bottom roller and trajectory of one of the points of the top face of the bottom roller.

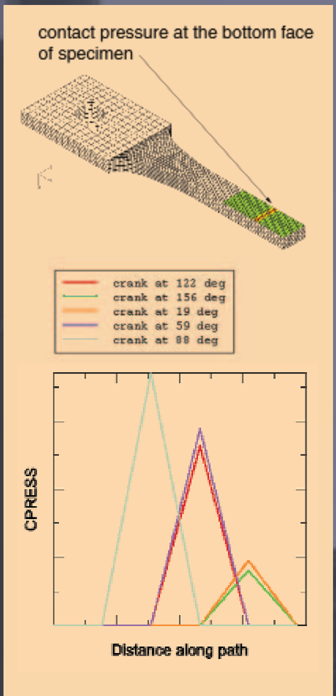


Figure 5: Path plots of contact pressure along the bottom face of the specimen during the first half period.

Reducing the Size of an Ouput Database Using the ABAQUS Scripting Interface

```
"""
usage: abaqus python copyOdbStep.py small.odb large.odb
where: small.odb is the output database to which you will
append data, and large.odb is the output database from which
you will read data.
"""

import sys, string
from odbAccess import openOdb

if len(sys.argv) != 3:
    print __doc__
    sys.exit()

odbPathSmall = sys.argv[1]
odbPathLarge = sys.argv[2]

# Open both output database files.

print 'Opening small output database: ', odbPathSmall
odbSmall = openOdb(path=odbPathSmall)

print 'Opening large output database: ', odbPathLarge
odbLarge = openOdb(path=odbPathLarge)

# Ask user for a step name since the step number can be ambiguous.

stepNames = odbLarge.step.keys()
print 'Step Names:', string.join(stepNames, ', ')

stepName = raw_input('Name of step to be copied: ')
stepName = string.strip(stepName)

if not stepName in stepNames:
    raise ValueError, 'Step name not found'
stepIndex = stepNames.index(stepName)

# Assign the step to a temporary variable.

stepLarge = odbLarge.step[stepName]
stepSmall = odbSmall.step[stepName]

# Ask user how many frames should be skipped.

print 'Number of frames in %s: %d'%(stepName, len(stepLarge.frame))
f = raw_input('Skip frequency (default=0 includes all frames): ')
if f == '':
    f = 0
else:
    f = int(f)
frameIncrement = f + 1

# Copy every specified frame to odbSmall.
# Make sure that the last frame is also copied.

framesToAdd = range(0, len(stepLarge.frame), frameIncrement)

lastFrameIndex = len(stepLarge.frame) - 1
if not lastFrameIndex in framesToAdd:
    framesToAdd.append(lastFrameIndex)

for i in framesToAdd:
    if stepIndex==0 and i==0:

        # For the first step, the base frame is written by datacheck.

        continue

    print 'Copying frame %d...' % i
    currentFrame = stepLarge.frame[i]
    newFrame = stepSmall.Frame(i, currentFrame.frameValue,
        currentFrame.description);
    for currentField in currentFrame.fieldOutput.values():
        newFrame.FieldOutput(currentField)

odbSmall.save();
odbSmall.close();
odbLarge.close();
```