`gmd` User's Guide

Tim Fuller

September 3, 2013

Chapter 1

Introduction

`gmd` is a Generalized Model Driver.

Chapter 2

Building gmd

gmd's code base is largely written in Python and requires no additional compiling. However, the exodusII third party library and material models written in fortran must be built.

2.1 Setting Up

Set up and build the third party libraries.

```
% cd GMD/toolset
% python setup.py
```

Add GMD/toolset to PATH.

2.2 Building

Build the material libraries

```
% buildmtls
```

Chapter 3

Running

Make sure `GMD/toolset` is on your `PATH`.

```
% gmd runid[.xml]
```

The following files will be produced

```
% ls runid.*
runid.exo       runid.log       runid.xml
```

`runid.exo` is the exodusII output database, `runid.log` the log file, and `runid.xml` the input file.

Chapter 4

User Input

User input is via xml control files. In general, tags use CamelCase and attributes lower case. Attributes are described in this document as

`attr="type[default]{choices}"`

where `default` is the default value (if any) and {`choices`} are valid choices (if any). Any attribute not having a default value are required. Types are `str`, `int`, `real`, `list`. Lists are given as space separated lists (i.e., "1 2 3").

4.1 GMDSpec

`<GMDSpec>`

All input files must have as their root element `<GMDSpec>`. Recognized subelements of `<GMDSpec>` are

- `<Physics>`
- `<Permutation>`
- `<Optimization>`

Additionally, the following elements are read from anywhere in the input file

- `<Include>`
- `<Function>`

4.2 Preprocessing

Preprocessing allows specifying variables in the input inside of comment tags for use in other parts of the input. Syntax mirrors that of `aprepro`.

4.2.1 Example

Specify the `<Material>` parameter `K` and `<Path>` parameter `estar` as variables

```
<GMDSpec>
  <!-- {K = 23e9}
       {estar = -.05}
  -->
  <Physics>
    <Material model="elastic">
      <K> {K} </K>
      <G> 54e9 </G>
    </Material>
    <Path type="prdef" estar="{estar}">
      ...
    </Path>
  </Physics>
</GMDSpec>
```

4.3 Include

```
<Include href="str"/>
```

Path to file to be included as if its contents were inplace in the input file

4.3.1 Example

```
<Include href="/path/to/some/file.ext"/>
```

4.4 Function

```
<Function id="int"
          type="str{analytic expression, piecewise linear}"
          var="str[x]" href="str" cols="list[1 2]">
```

Define functions to be used elsewhere in input. `id=0` and `id=1` are reserved for the constant 0 and 1 functions, respectively.

4.4.1 Examples

Analytic expression

```
<Function id="2" type="analytic expression" var="t">
  sin(t)
</Function>
```

Piecewise linear table

```
<Function id="2" type="piecewise linear">
  1 2
  2 3
  3 5
</Function>
```

Read a piecewise linear table from an external file using columns 1 and 3

```
<Function id="2" type="piecewise linear" href="./file.dat" cols="1 3"/>
```

```
% cat file.dat
# Column1 Column2 Column3
1 1 4
2 3 7
.
.
.
100 4.2 1.43
```

4.5 Physics

```
<Physics driver="str[solid]{solid, eos}" termination_time="real[]">
```

Define the physics of the simulation. Recognized subelements of `<Physics>` are

- `<Path>`
- `<Surface>`

- `<Material>`
- `<Extract>`

If specified, `termination_time` defines the termination time for the simulation. If not specified, termination time is taken as final time in `<Path>`.

4.5.1 Path

Supporting Drivers: `solid`

```
<Path type="str{prdef}"
      format="str[default]{default, table, fcnspec}"
      cols="list[1, ..., n]" cfmt="str" tfmt="time"
      nfac="int[1]" kappa="real[0]"
      tstar="real[1]" estar="real[1]" sstar="real[1]"
      amplitude="real[1]" ratfac="real[1]" href="str">
```

Define deformation paths. The jth leg of `<Path>` is sent to the driver in form `[tf, n, cfmt, Cij]`, where `tf`, `n`, `cfmt`, and `Cij` are the termination time, number of steps, control format, and control values.

A note on `cfmt` and `Cij`. `cfmt[i]` instructs the driver as to the type of deformation represented by `Cij[i]`. Supported `cfmt` are described in Table 4.1. For example, the following `cfmt` instructs the driver that the components of `Cij` represent [stress, strain, stress rate, strain rate, strain, strain], respectively: `cfmt = 423122`. Mixed modes are allowed only for components of strain rate, strain, stress rate, and stress. Electric field components can be included with any deformation type.

The components `Cij` take the following order

Vectors: [X, Y, Z]

Symmetric tensors: [XX, YY, ZZ, XY, YZ, XZ]

Tensors: [XX, XY, XZ, YX, YY, YZ ZX, ZY, ZZ]

If `len(Cij)` $\neq$ `6` (or 9 for deformation gradient), the missing components are assumed to be zero strain.

Examples

The following examples will help clarify the `<Path>` input syntax

format: default Uniaxial strain, all six components of strain prescribed

cfmt	Deformation type
1	Strain rate
2	Strain
3	Stress rate
4	Stress
5	Deformation gradient
6	Electric field

Table 4.1: Supported deformation types and `cfmt` code

```
<Path type="prdef" kappa="0" tstar="1" estar="-.5" amplitude="1" ratfac="1">
  <!-- termination time, number of steps, cfmt, Cij -->
  0   0 222222 0 0 0 0 0 0
  1 100 222222 1 0 0 0 0 0
  2 100 222222 2 0 0 0 0 0
  3 100 222222 1 0 0 0 0 0
  4 100 222222 0 0 0 0 0 0
</Path>
```

format: default Uniaxial strain, stress controlled

```
<Path type="prdef" nfac="100">
  0 0 444 0 0 0
  1 1 444 -7490645504 -3739707392 -3739707392
  2 1 444 -14981291008 -7479414784 -7479414784
  3 1 444 -7490645504 -3739707392 -3739707392
  4 1 444 0 0 0
</Path>
```

format: default Uniaxial stress, mixed mode

```
<Path type="prdef" nfac="100">
  0 0 222 0 0 0
  1 1 244 {epsmax} 0 0
  4 1 244 0 0 0
</Path>
```

format: table Read legs from table. Control type is uniform for all legs. Specify control type as `cfmt` attribute of `<Path>`. Optionally, specify the time format as `tfmt` and number of steps for each leg as `nfac`.

```
<Path type="prdef" format="table" cols="1:4" cfmt="222" tfmt="time">
  0 0 0 0
  1 1 0 0
    ...
  n 2 0 0
</Path>
```

format: table Read the table from a file, first by the `<Include>` element and then the `href` attribute.

```
<Path type="prdef" format="table" cols="1 3:8" cfmt="222222" tfmt="time">
  <include href="exmpls.tbl"/>
</Path>

<Path type="prdef" format="table" cols="1 3:8" cfmt="222222" tfmt="time"
      href="exmpls.tbl"/>
```

format: fcnspec Create legs from functions. Functions are specified as `function id[:scale]`. Syntax is otherwise similar to table format. Only a single leg can be specified.

```
<Path type="prdef" kappa="0" tstar="1" amplitude="1" format="fcnspec"
      cfmt="222" nfac="200">
  {2 * pi} 2:1.e-1 1:0 1:0
</Path>
```

4.5.2 Surface

Supporting Drivers: `eos`

```
<Surface format="str[default]{default, table}"
         cols="list[1, ..., n]" cfmt="str"
         nfac="int[1]" tstar="real[1]" rstar="real[1]"
         amplitude="real[1]" href="str">
```

Define equaiton of state surface boundaries. Input is similar to the `<Path>` specification, but leg termination time is not specified. Control parameters also differ, as shown in Table 4.2.

Examples

The following examples demonstrate the `<Surface>` input.

cfmt	Variable type
1	Density
2	Temperature

Table 4.2: Supported surface variable types and `cfmt` code

format: default

```
<Surface>
  <!-- nsteps, control, Cij -->
  <!-- control goes as 1 -> density
                       2 -> temperature -->
    0 12 1 100
  100 12 5 300
</Surface>
```

format: table

```
<Surface format="table" cfmt="12" nfac="100">
  <!-- Cij -->
  1 100
  5 300
</Surface>
```

4.5.3 Material

```
<Material model="str">
```

Specify the material model and parameters. Subelements of `<Material>` are

- `<Matlabel>`
- `<Key>`

Where `<Key>` is a valid material parameter name.

Matlabel

```
<Matlabel href="str[F_MTL_PARAM_DB]">
```

Insert model parameters from a database file. The default file `F_MTL_PARAM_DB` is in `/path/to/gmd/materials/material_properties.db`.

Key

```
<Key> float </Key>
```

Examples

```
<Material model="elastic">
  <G>  54E+09 </G>
  <K> 124E+09 </K>
</Material>
```

```
<Material model="elastic">
  <Matlabel href="./materials.xml"> aluminum </Matlabel>
  <K> 124E+09 </K>
</Material>
```

4.5.4 Extract

```
<Extract format="str[ascii]{ascii, mathematica}" step="int[1]" ffmt="str[.18f]">
```

Extract variables and paths from exodus output and (optionally) write to different formats. Recognized subelements of `<Extract>` are

- `<Path>`
- `<Variables>`

Variables

```
<Variables> VAR_1, ..., VAR_N </Variables>
```

Variables to extract from the exodus output database. Variables are specified children of the `<Variables>` element. All components of vector and tensor variables will be extracted if only the basename is specified. Time is always extracted as the first entry of the output file. Extracted variables are in `runid.out` or `runid.math` depending if the format is ascii or mathematica.

Path

Supporting Drivers: `eos`

```
<Path type="str{isotherm, hugoniot}" increments="int[100]"
      density_range="list" initial_temperature="real">
```

Extract a specified path from the equation of state surface through the specified density range starting at the initial temperature.

Examples

Extract all components of stress and strain

```
<Extract format="ascii">
  <variables>
    STRESS STRAIN
  </variables>
</Extract>
```

Extract only the XX, YY, and ZZ components of stress

```
<Extract format="ascii">
  <variables>
    STRESS_XX STRESS_YY STRESS_ZZ
  </variables>
</Extract>
```

Extract all variables

```
<Extract format="ascii">
  <variables>
    ALL
  </variables>
</Extract>
```

Extract Hugoniot and Isotherm paths

```
<Extract>
  <Path type="isotherm" increments="200"
        density_range="1 3" initial_temperature="225"/>
  <Path type="hugoniot" increments="100"
        density_range="1 3" initial_temperature="100"/>
</Extract>
```

4.6 Permutation

```
<Permutation method="str[zip]{zip, combine, shotgun}" seed="real[12]"
                     correlation="list[none]{plot, table, none}">
```

Permutate model input parameters, running jobs with different realization of parameters. Good for investigating model sensitivities. Recognized subelements of `<Permutation>` are

- `<Permutate>`
- `<ResponseFunction>`

The `method` attribute describes which method to use to determine parameter combinations to run. The `zip` method runs one job for each set of parameters (and, thus, the number of realizations for each parameter must be identical), the `combine` method runs every combination of parameters, finally, the `shotgun` method zips a uniform distribution for each parameter.

The `correlation` attribute is only meaningful if a `<ResponseFunction>` is specified. Also, note that issues relating to reading the exodusII database prevent gmd from running simultaneous permutation jobs that define a `<ResponseFunction>`.

4.6.1 Permutate

```
<Permutate var="str"
           values="str{range, list, weibull, uniform, normal, percentage}"
```

Specify the paramaters to permutate. Variable names should occur elsewhere in the input file in preprocessing braces.

4.6.2 Example

Permutate the `K` and `G` parameters

```
<Permutation method="zip" seed="12">
  <Permutate var="K" values="weibull(125.e9, 14, 3)"/>
  <Permutate var="G" values="percentage(45.e9, 10, 3)"/>
</Permutation>
```

In the `<Material>` element, the `K` and `G` parameters are specified as

```
<Material model="elastic">
  <K> {K} </K>
  <G> {G} </G>
</Material>
```

4.6.3 ResponseFunction

```
<ResponseFunction href="str" descriptor="str[]"/>
```

Name of response function that returns the response from permutation or optimization jobs. `href` must either be the path to a file containing the response function, or the name of a builtin `gmd` response function.

Built in response functions are

- `gmd.max`
- `gmd.min`
- `gmd.mean`
- `gmd.ave`
- `gmd.absmax`
- `gmd.absmin`

Built in response functions operate only on variabes in the simulation output file.

If `href` is a user defined script, the script is called from the command line as

```
% ./scriptname simulation_output.exo [auxiliary_file_1 [... auxiliary_file_n]]
```

Examples

```
<ResponseFunction href="./scriptname" descriptor="PRES"/>
```

```
<ResponseFunction href="gmd.max(PRESSURE)" descriptor="PRES"/>
```

4.7 Optimization

```
<Optimization method="str[simplex]{simplex, powell, cobyla}"
              maxiter="int[25]" tolerance="real[1e-6]">
```

Optimize specified parameters against user specified objective function. Recognized subelements of `<Optimization>`

- `<Optimize>`
- `<AuxiliaryFile>`
- `<ResponseFunction>`

4.7.1 Optimize

```
<Optimize var="str" initial_value="real" bounds="list[]"/>
```

Specify the variable to be optimized, giving initial value and, optionally, bounds. Only the `cobyla` method accepts bounds. Variable names should occur elsewhere in the input file in preprocessing braces.

4.7.2 ResponseFunction

Same as for `<Permutation>`. The value returned from the response function is interpreted as the error to be minimized.

4.7.3 AuxiliaryFile

```
<AuxiliaryFile href="str"/>
```

Path to any auxiliary file needed by the optimization objective function.

4.7.4 Example

Optimize the `K` and `G` parameters

```
<Optimization method="simplex" maxiter="25" tolerance="1e-4" disp="0">
  <ResponseFunction href="opt-sig-v-time" descriptor="SIG_V_TIME"/>
  <AuxiliaryFile href="opt-baseline.dat"/>
  <Optimize var="opt_k" initial_value="129.e9"/>
  <Optimize var="opt_g" initial_value="54.e9"/>
</Optimization>
```

In the `<Material>` element, the `K` and `G` parameters are specified as

```
<Material model="elastic">
  <K> {opt_k} </K>
  <G> {opt_g} </G>
</Material>
```