`gmd` User's Guide

Tim Fuller

August 31, 2013

Chapter 1

Introduction

`gmd` is a Generalized Model Driver.

Chapter 2

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").

2.1 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>`
- `<TerminationTime>`

2.2 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`.

2.2.1 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>
```

2.3 Include

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

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

2.3.1 Example

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

2.4 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.

2.4.1 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
```

2.5 TerminationTime

```
<TerminationTime> float </TerminationTime>
```

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

2.5.1 Example

```
<TerminationTime> 1.e-6 </TerminationTime>
```

2.6 Physics

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

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

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

2.6.1 Path

```
<Path type="str{prdef, prstate, isotherm, hugoniot}"
      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 2.1. 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

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

Table 2.1: Supported deformation types and `cfmt` code

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

```
<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>
```

2.6.2 Material

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

Specify the material model and parameters. Material parameters are specified as individual elements.

Material database

```
<Matlabel db="str[MTL_PARAM_DB_FILE]">
```

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

Examples

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

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

2.6.3 Extract

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

Extract variables from exodus output to different formats. Variables to be extracted are specified children of the `<Extract>` 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.

Examples

Extract all components of stress and strain

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

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

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

Extract everything

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

2.7 Permutation

```
<Permutation method="str[zip]{zip, combine}" seed="real[12]">
```

Permutate model input parameters to investigate sensitivities. Recognized subelements of `<Permutation>`

- `<Permutate>`

2.7.1 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.

2.7.2 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>
```

2.8 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>`
- `<ObjectiveFunction>`

2.8.1 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.

2.8.2 AuxiliaryFile

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

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

2.8.3 ObjectiveFunction

Path to a user defined executable script that returns the error to the optimization routine.

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

An `ObjectiveFunction` given by `href` is called from the command line as

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

2.8.4 Example

Optimize the `K` and `G` parameters

```
<Optimization method="simplex" maxiter="25" tolerance="1e-4" disp="0">
  <ObjectiveFunction href="opt-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>
```