Partial Grading
Readme

Aron Pasieka
aron@aron.ca

September 3, 2012

1 About

The PartialGrading library was written for use in Maple TA. In developing content for a first-year university-level electricity and magnestism course, I found that I wanted more flexibility in how Maple TA assigned grades. The numeric grading options therein allowed for marks to be given if a student was correct within sig figs *or* within uncertainty *or* within uncertainty in the N^{th} digit – but these rules could not be in effect concurrently. More importantly, on a question with units, there was no way to assign part marks to the value and units if one also wanted flexibility for the students to enter $3cm$ or $0.03m$.

After building Maple code to handle these scenarios, I decided that I could build upon the code to handle vectors responses as well – providing part marks if students not only got a single component incorrect, but if they permuted components, provided a scalar instead of a vector, etc.

With this code in place, it was a simple matter to extend the output to include a detailed list of how a particular grade was calculated for the comments section.

Finally, I added partial grading for algebraic responses.

2 Structure

Calls should be routed through the module `PartialGrading[Grade]`. It will then pass options along to either `GradeVectors[GVectors]` or `GradeScalars[GScalars]`. These two procedures can handle either numeric or algebraic input. As a fail-safe, `GradeVectors[GVectors]` will pass back to `GradeScalars[GScalars]` if both answer and response are scalar quantities.

The module `SigFigs` does not need to be addressed directly, but provides sig fig checking and rounding support.

3 Usage

3.1 General

The partial grading can be used in any Maple-Graded question (including Question Designer). The files,

```
PartialGrading.lib
PartialGrading.ind
```

must be uploaded via the Web Site Editor. In the question itself, under *Repository*, link the .lib file. In the *Grading Code* section, enter the appropriate call to `PartialGrading[Grade]` as described in the following sections.

3.2 Grading Comments

Currently, the grading comments are only available in Maple-Graded questions (not Question Designer). This is due to the fact that `$ANSWER` and `$RESPONSE` are not addressable in Question Designer questions. Furthermore, the input method must be set to Text Entry, not Equation Editor. If the syntax option is set to Maple, Text Entry can be forced, but if it is set to Formula, the student cannot be forced into Text Entry mode.

Example code for the comment section (replace `_options` with grading options as outlined in the following sections and `_PATH` with the path to the library):

```
${if(eq($RESPONSE,No answer),"No answer entered.",maple("PartialGrading[Grade]($ANSWER,‘$RES
```

3.3 Scalars

Numeric scalars can be graded with the following call:

```
PartialGrading[Grade]($ANSWER,$RESPONSE,subPackage=numericScalar,_options);
```

while algebraic scalars can be graded with the following:

```
PartialGrading[Grade]($ANSWER,$RESPONSE,subPackage=algebraicScalar,_options);
```

Both of these calls will be routed to the `GradeScalars[GScalars]` procedure. They will, by default, output a float between 0.0 and 1.0.

There are a number of keyword style options that can be used. Most keywords have a long-form and short-form for ease of use, which are indicated by square brackets, a type indicated by colons and a default value indicated by assignment, as follows:

```
[longForm,shFrm]::type:=defaultValue
```

3.3.1 Global Flags

The options in this section affect the way the entire procedure functions. They apply to `GradeScalars[GScalars]`.

1. `[algebraicScalar,algScal]::boolean:=false`

   Controls whether numeric or algebraic processing and checking is performed on `$ANSWER` and `$RESPONSE`. This is set by `PartialGrading[Grade]` as noted above.

Table 1: Scalar Grading Categories

Category	Name	Description
1	Base Grades	The property that defines a fully correct answer
2	Secondary Grades	Properties that define almost fully correct answers
3	Multiplicative If Non-zero	If the grade is non-zero at this point and they do not satisfy this property, multiply the grade.
4	Multiplicative Factors I	Properties that warrant a scaling of the grade
5	Additive Factors	Properties that warrant an addition to the grade
6	Subtractive Factors	Properties that warrant an subtraction from the grade
7	Additive and Subtractive	If the grade at this point is 0.0 and they satisfy this property, add to the grade. If the grade is non-zero and they do not satisfy this property, subtract from the grade
8	Multiplicative Factors II	Properties that warrant a final scaling of the grade

2. `[giveComments]::boolean:=false`

   Controls whether the procedure outputs a grade or MathML comments.

3. `[internalComments]::boolean:=false`

   Instead of fully formed MathML, outputs just MathML table rows for insertion into other commands.

3.3.2 Grading Options

These options affect how the grading is performed in `GradeScalars[GScalars]`. In order to make the marking as flexible as possible, each grading option can be assigned to one of eight categories, which can be seen in Table 1 in the order that they are applied.

1. `[useExactMatch,uEM]::boolean:=algebraicScalar`
   `[marksExactMatch,mEM]::float:=1.0`
   `[catExactMatch,cEM]::integer:=1`

   Checks to see if `$ANSWER` and `$RESPONSE` are equal without concern for sig figs. The default is false (my preference for sig figs marking).

2. `[useSigFigs,uSF]::boolean:=true`
   `[numSigFigs,nSF]::integer:=3`
   `[marksSigFigs,mSF]::float:=1.0`
   `[catSigFigs,cSF]::integer:=1`

   Checks to see if `$ANSWER` and `$RESPONSE` are equal to the specified sig figs. No effect in algebraic mode.

3. `[useAlgMultMarks,uAMM]::boolean:=true`
   `[marksAlgMultMarks,mAMM]::float:=1.0`
   `[catAlgMultMarks,cAMM]::integer:=1`

   Checks to see if `$ANSWER` and `$RESPONSE` are equal. For every algebraic term in `$ANSWER/$RESPONSE`, reduce the grade by the inverse of the number of terms in the correct answer. No effect in numeric mode.

4. `[usePercentError,uPE]::boolean:=false`
   `[numPercentError,nPE]:=1.0`
   `[marksPercentError,mPE]::float:=0.8`
   `[catPercentError,cPE]::integer:=2`

   Checks to see if `$ANSWER` and `$RESPONSE` are equal to the specified percent error (numPercentError). No effect in algebraic mode.

5. `[useUncertainty,uU]::boolean:=false`
   `[numUncertainty,nU]:=0.01`
   `[marksUncertainty,mU]::float:=0.8`
   `[catUncertainty,cU]::integer:=2`

   Checks to see if `$ANSWER` and `$RESPONSE` are equal to the specified uncertainty (numUncertainty). No effect in algebraic mode.

6. `[useUncertaintyNth,uUN]::boolean:=false`
   `[numUncertaintyNth,nUN]::integer:=1`
   `[digUncertaintyNth,dUN]::integer:=3`
   `[marksUncertaintyNth,mUN]::float:=0.66`
   `[catUncertaintyNth,cUN]::integer:=2`

   Checks to see if `$ANSWER` and `$RESPONSE` are equal to the specified uncertainty (numUncertaintyNth) in the N^{th} digit (digUncertaintyNth). No effect in algebraic mode.

7. `[useDimensionsMarks,uDM]::boolean:=true`
   `[marksDimensionsMarks,mDM]::float:=0.25`
   `[catDimensionsMarks,cDM]::integer:=7`

   Checks to see if `$ANSWER` and `$RESPONSE` are equal dimensioanlly. No effect in algebraic mode.

8. `[useSignMarks,uSM]::boolean:=true`
   `[marksSignMarks,mSM]::float:=0.75`
   `[catSignMarks,cSM]::integer:=4`

   Checks to see if `$ANSWER` and `$RESPONSE` are equal in sign.

9. `[useOrderMarks,uOM]::boolean:=true`
   `[marksOrderMarks,mOM]::float:=0.75`
   `[catOrderMarks,cOM]::integer:=3`

   Checks to see if the mantissa of `$ANSWER` and `$RESPONSE` are equal. No effect in algebraic mode.

3.3.3 Examples

A few example scenarios:

1. Numeric. Full marks if numbers match to 3 sig figs. Give 0.9 if values differ by 1 in the third sig fig. Subtract 0.25 if they get the sign incorrect. Ignore any dimensionality.

   ```
   PartialGrading[Grade]($ANSWER,$RESPONSE,subPackage=numericScalar,uSF,cSF=1,
   nSF=3,mSF=1.0,uUN,cUN=2,nUN=1,dUN=3,mUN=0.9,uSM,cSM=5,mSM=0.25,uDM=false);
   ```

2. Numeric. Full marks if number match without sig figs. Must get dimensions exactly correct. Must get sign correct.

   ```
   PartialGrading[Grade]($ANSWER,$RESPONSE,subPackage=numericScalar,uSF=false,uEM,
   cEM=1,mEM=1.0,uSM,cSM=7,mSM=0.0,uDM,cDM=7,mDM=0.0);
   ```

3. Numeric. Full marks if answer equals response dimensionally, with no regard to value.

   ```
   PartialGrading[Grade]($ANSWER,$RESPONSE,subPackage=numericScalar,uSF=false,
   uDM,cDM=1,mDM=1.0,uSM=false);
   ```

3.4 Vectors

Vectors with numeric components can be graded with the following call:

```
PartialGrading[Grade]($ANSWER,$RESPONSE,subpackage=vectorNumericComponents,_options);
```

while vectors with algebraic components can be graded with the following:

```
PartialGrading[Grade]($ANSWER,$RESPONSE,subPackage=vectorAlgebraicComponents,_options);
```

Both of these calls will be routed to the `GradeVectors[GVectors]` procedure. They will, by default, output a float between 0.0 and 1.0. `GradeVectors[GVectors]` will pass the question along to `GradeScalars[GScalars]` if both `$ANSWER` and `$RESPONSE` are scalars. For this reason, `subpackage=vectorNumericComponents` is the default value and may be omitted.

There are a number of keyword style options that can be used. Most keywords have a long-form and short-form for ease of use, which are indicated by square brackets, a type indicated by colons and a default value indicated by assignment, as follows:

```
[longForm,shFrm]::type:=defaultValue
```

Table 2: Vector Grading Categories

Category	Name	Description
11	Component Multiplicative Factors	Properties that warrant a scaling of the component grade
12	Component Subtractive Factors	Properties that warrant a subtraction from the component grade
21	Global Multiplicative Factors	Properties that warrant a scaling of the total grade
22	Global Subtractive Factors	Properties that warrant a subtraction from the total grade

3.4.1 Global Flags

The options in this section affect the way the entire procedure functions. They apply to `GradeVectors[GVectors]`.

1. `[algebraicComponents,algComp]::boolean:=false`

   Controls whether numeric or algebraic processing and checking is performed on `$ANSWER` and `$RESPONSE`.

2. `[giveComments]::boolean:=false`

   Controls whether the procedure outputs a grade or MathML comments.

3.4.2 Grading Options

These options affect how the grading is performed in `GradeVectors[GVectors]`. In order to make the marking as flexible as possible, each vector grading option can be assigned to one of four categories, which can be seen in Table 2 in the order that they are applied. These options define the permutations of the components in the response that are valid. The permutation and penalty combination that yields the highest total grade will be used. Each component is individually marked by the `GradeScalars[GScalars]` procedure as outlined above, thus one can include the full selection of scalar grading options along with the vector grading options when using this procedure.

1. `[useComponentMatch,uCM]::boolean:=true`
   `[marksComponentMatch,mCM]::float:=1.0`
   `[catComponentMatch,cCM]::integer:=21`

   Defines the default permutation, no components are moved.

2. `[useSwitch12,uS12]::boolean:=false`
   `[marksSwitch12,mS12]::float:=0.75`
   `[catSwitch12,cS12]::integer:=11`

   Defines a permutation where components one and two are interchanged.

3. `[useSwitch23,uS23]::boolean:=false`
   `[marksSwitch23,mS23]::float:=0.75`
   `[catSwitch23,cS23]::integer:=11`

   Defines a permutation where components two and three are interchanged.

4. `[useSwitch13,uS13]::boolean:=false`
   `[marksSwitch13,mS13]::float:=0.75`
   `[catSwitch13,cS13]::integer:=11`

   Defines a permutation where components one and three are interchanged.

5. `[useSwitchAny2,uSA2]::boolean:=false`
   `[marksSwitchAny2,mSA2]::float:=0.75`
   `[catSwitchAny2,cSA2]::integer:=11`

   Defines all three of the preceeding permutations. If they are set individually as well, the individual settings override these for the specified components.

6. `[useComponentTranspose,uCT]::boolean:=true`
   `[marksComponentTranspose,mCT]::float:=0.75`
   `[catComponentTranspose,cCT]::integer:=21`

   Defines all possible permutations of the components.

7. `[useVectorScalar,uVS]::boolean:=true`
   `[marksVectorScalar,mVS]::float:=0.5`
   `[catVectorScalar,cVS]::integer:=21`

   Defines all ways to match a scalar `$RESPONSE` to a vector `$ANSWER` or vice-versa. Combined with the fact that if both `$ANSWER` and `$RESPONSE` are scalars the question is forwarded to the scalar marker, one could call the vector marker alone for, what should be, fully scalar questions.

3.4.3 Examples

A few example scenarios:

1. Numeric. Full marks if numbers match to 3 sig figs. Give 0.9 if values differ by 1 in the third sig fig. Subtract 0.25 if they get the sign incorrect. Ignore any dimensionality. Scale total mark by 0.75 if they mix up the components in any way.

   ```
   PartialGrading[Grade]($ANSWER,$RESPONSE,subPackage=vectorNumericComponents,
   uSF,cSF=1,nSF=3,mSF=1.0,uUN,cUN=2,nUN=1,dUN=3,mUN=0.9,uSM,cSM=5,
   mSM=0.25,uDM=false,uCT,cCT=21,mCT=0.75);
   ```

2. Algebraic. Full marks if values match. Subtract 0.25 if they get the sign incorrect. Scale component marks by 0.75 if they mix up any two components. Scale total mark by 0.8 if they enter a scalar response.

```
PartialGrading[Grade]($ANSWER,$RESPONSE,subPackage=vectorAlgebraicComponents,
uSM,cSM=5,mSM=0.25,uSA2,cSA2=11,mSA2=0.75,uVS,cVS=21,mVS=0.8);
```