problem2tex

version = 0.8.5 (2021-06-15)

David Johns,
david.johns@icewire.ca

Document Date - 2021-06-15

Contents

1 The need for problem to tex

Latex is an excellent way to create educational material such as textbooks, examples, exams, and problem sets. The purpose of `problem2tex` creating a .tex file is to allow one to create a numerical example (or problem) .prb file so that parameters can change and have the solution be automatically updated for that choice of parameters. In addition, `problem2tex` has an expression solver that is used to quickly write the solution for an example (or problem).

Example usage:

```
problem2tex -export=example.tex -random=false -sigDigits=4 example.prb
```

2 Installation and Setup

For installation/setup of problem2tex, see https://www.icewire.ca

3 Running problem2tex

3.1 Command Line Options

The command line options for `problem2tex` are the following:

- `-help`

  Print out help info

- `-version`

  Print out version info

- `-export=path/filename.tex`

  where the output should be placed

  path is the directory path that can include .. or . and subdirectories

- `-random=option`

  where option is one of ... (problem to tex option)

  true: Parameters are randomized

false: Parameters are the first values in the sets (default setting)

positive integer: Seed for random number generator

min: All smallest numbers

max: All largest numbers

minMax: Random mix of largest and smallest numbers

- `-sigDigits=value`

  where value is an integer that sets the default number of significant digits in the output (If not specified, then it is set to 4). RunConfig can also be used to set number of significant digits.

3.2 Error Logging

When running problem2tex, error information is placed as comments at the beginning of the output .tex file. If things do not work as you expect, check the error log information first as Latex error information can sometimes be misleading.

3.3 Examples

problem2tex is a way to make problems have parameters that can change and the solution is re-calculated for that solution. In addition, the solution is easily written as equations which are then displayed as latex solutions. For this to work, problem2tex has a built-in expression solver similar to Julia or Matlab.

This example is available at testBasic.zip

In this example, basic01.prb is a user generated problem file and contains the following text:

```
\runParam{x = [2, 3, 4, 5]}
\runParam{y = [6, 7, 8, 9] }
\runParam{k = [8]}
\question Given \val={x}, \val={y}, and \val={k} find $z = x^2+y-k$\\
\textbf{Solution}\\
\run(){z=x^2+y-k}\\
\hlite{\val={z}}
```

For the above problem, x, y and k are all parameters where x is any one of 2,3,4 or 5 and y is any one of 6,7,8 or 9 and k is 8. For the solution, it is simply written as the equation and a built-in expression solver solves the expression and writes it out in correct Latex form. The solution in this case is written as

```
\run(){z=x^2+y-k}
```

and when running with the following command

```
problem2tex -export=example.tex example.prb
```

the following tex file is created ...

```
% Created with problem2tex: version = (version info)
\question Given \mbox{$x = \mbox{$2 \units{ }$}$}, \mbox{$y = \mbox{$6 \units{ }$}$},
and \mbox{$k = \mbox{$8 \units{ }$}$} find $z = x^2+y-k$\\
\textbf{Solution}\\
\mbox{$z=x^2+y-k  = (2)^2+(6)-(8)$}\\
\hlite{\mbox{$z = \mbox{$2 \units{ }$}$}}
```

The command \mbox is used so that inline equations can be used either within or outside other inline equations. The command \units is used to improve the font when displaying units (there are no units for this example).

In the same testBasic.zip file, the second example basic01.prb contains the following text:

```
\runParam{V_1 = [2, 3, 4, 5]# \units{V}}
\runParam{R_1 = [6, 7, 8, 9]# \units{k \Omega}}
\question Given \val={V_1} and \val={R_1}, find the current $I_R = V_1/R_1$\\
\textbf{Solution}\\
\run(){I_R=V_1/R_1}\\
\hlite{\val={I_R}}
```

What is interesting here is that units can be assigned to parameters and the expression solver takes into account unit prefixes to generate the correct solution value prefix (i.e, f, p, n, μ, m, k, M, G, etc).

Also related to units, the units for any variable can be set using the \units{} command. In addition, a parameter in runConfig called defaultUnits can be used to set the default units depending on the first letter of the variable. (see runConfig)

NOTE: micro uses \mu (and not the letter u)

In this example, the above 2 problem files are included in a filed called basic.tex which contains the following text:

```
\documentclass[11pt]{exam}
\usepackage{import,xcolor}
\newcommand{\incProb}[1]{
    \immediate\write18{problem2tex -export=Problems/tmp/#1.tex
    -random=false -sigDigits=4 Problems/#1.prb}
  \import{Problems/tmp/}{#1.tex}
}
\newcommand{\hlite}[1]{%
  \colorbox{yellow!50}{#1} }
\newcommand{\units}{\,\mathrm}

\begin{document}
\begin{questions}

\incProb{basic01}
\incProb{basic02}

\end{questions}
\end{document}
```

The tex file is using the exam class (for question numbering) and has 3 commands defined: \incProb; \hlite; \units.

When the above basic.tex is compiled as a text document, the following output is generated:

1. Given $x = 2$, $y = 6$, and $k = 8$ find $z = x^2 + y - k$

   Solution
   $z = x^2 + y - k = (2)^2 + (6) - (8)$
   $z = 2$

2. Given $V_1 = 2\,\mathrm{V}$ and $R_1 = 6\,\mathrm{k\Omega}$, find the current $I_R = V_1/R_1$

   Solution
   $I_R = V_1/R_1 = (2)/(6e3)$
   $I_R = 333.3\,\mu\mathrm{A}$

In this case, all default parameters are used but it is one line change to obtain random parameters and change the significant number of digits.

```
problem2tex --export=Problems/tmp/#1.tex --random=true --sigDigits=6 Problems/#1.prb
```

When run using the above random flag, one example random case is the following: (each run will be random)

1. Given $x = 5$, $y = 8$, and $k = 8$ find $z = x^2 + y - k$

   Solution
   $z = x^2 + y - k = (5)^2 + (8) - (8)$
   $z = 25$

2. Given $V_1 = 3\,\text{V}$ and $R_1 = 7\,\text{k}\Omega$, find the current $I_R = V_1/R_1$

   Solution
   $I_R = V_1/R_1 = (3)/(7e3)$
   $I_R = 428.571\,\mu\text{A}$

3.4 Configuration settings

Configuration settings can be changed at any time and then they are used going forward until changed again. Configuration settings can be changed using the \runConfig command.
Example: \runConfig{random = min}

- random - choice of false, true, min, max, minMax and positive integer
  - false: defaults elements chosen
  - true: random elements chosen
  - min: min sized elements chosen
  - max: max sized elements chosen
  - minMax: random choice of min and max sized elements chosen
  - positive integer: seed for random generator so same elements can be chosen
- fmtVal - format of output values for \val
  - set as \runConfig{format=X} where X is one of ...
    (the number of significant digits can be from 1-9)
  - E4 for engineering format with 4 significant digits
  - S4 for scientific format with 4 significant digits
  - D4 for decimal format with 4 significant digits
  - $ for dollar format (always has 2 digits after decimal point)
  - U4 for SI format (including units) with 4 significant digits
  - DEFAULT IS U4
- fmtRun= - format of output values after equal in \run= or \run()= commands
  - Same as format above
  - DEFAULT IS U4
- fmtRun() - format of output values in bracketed numbers in \run() or \run()= commands
  - Same as format above except U format not allowed

- DEFAULT IS E4

- KFactor - Used for default set generation if \runParam sets var to a nominal number.

  - Set KFactor example: \runConfig{KFactor = 1.5:5} where 1.5 is the factor and 5 is the number of elements in the set.

  - Factor must be a number greater than 1.

  - The range of the elements are from nominal/factor to nominal*factor and are geometrically spaced.

- defaultUnits - used to set the default units depending on the first letter of a variable

  - example: \runConfig{defaultUnits = [[iI:A][vV:V][R:\Omega]]}

  - above example results in variables starting with the letter i or I having default units of "A"

  - variables starting with letter v or V having default units of "V"

  - variables starting with letter R having default units of "\Omega"

  - use of \units within a run command will override the default unit setting

- verbose - choice of true or false (default false)

  - prints out the elements sets in commented out lines in the .tex file (useful for debugging)

  - Also prints out the configuration settings

3.5 Commands for .prb files

There are 4 main commands for problem to tex:

- \runConfig (discussed above)
- \runParam
- \run
- \val

3.5.1 \runParam command

The command \runParam can be used to set configuration parameters (see above) as well as being used for setting the randomly generated parameters.

- \runParam{var = [x1, x2, x3, ...]# \units{units} \symbol{varLatex} }

  - var will be a random selection from the set of x1, x2, x3, ...

  - if random=false, the default value for var is the first element

  - if \units{units} is present, then the units for that variable will be units

  - if \symbol{varLatex} is present, then when printing out, var will be replaced with varLatex

- \runParam{var = min;max;stepsize# \units{units} \symbol{varLatex} }

  - The set for var will be generated from min;max;stepsize

  - The first element will be min so the default will be the min element

  - Otherwise, it is the same as the array generation above

- \runParam{var = nominal# \units{units} \symbol{varLatex} }

    - The set for var will be generated from nominal and the KFactor configuration parameter

    - KFactor: factor:numElements... factor is a number greater than 1 and numElements is a positive integer (numElements is the number of elements in the set).

    - Elements range from nominal/factor to nominal*factor and are geometrically spaced

    - Example: if nominal is 10 and factor is 1.5, then the range is from 6.667 to 15

    - The default is nominal which would be 10 in the above example

    - Otherwise, it is the same as the array generation above

3.5.2 \run commands

The commands \run are used for evaluating expressions and setting new variables.
The format for an expression is the same as Matlab or Julia (NOT a latex equation).
See the Expression Solver section below for more information.
Options are the same as in \runParam.
In all cases below, the expr is evaluated and the result is assigned to var

- \run{var = expr # options }

    - Print out var = expr.

    - There can be multiple var = expr separated by ";"

- \runSilent{var = expr # options}

    - Do not print anything out

- \run(){var = expr # options}

    - Print out var = expr AND print out intermediate () expr for clarity

- \run={var = expr # options}

    - Print out var = expr AND print out "= result"

- \run()={var = expr # options}

    - Print out var = expr AND print out intermediate () expr AND print out "= result"

3.5.3 \val command

Below is the \val command for printing out variable value or an expression value.

- \val{expr,format}

    - Print out the result of expr with format set by format (see \runConfig)

    - ,format is optional. If not present, then the default setting for format is used which was set by \runConfig{format = X}

    - expr in a \val command should NOT contain a ","

    - expr in a \run command may contain ","s

    - expr can be a single variable or a full expression

3.6 Expression Solver

Problem to tex makes use of a built in expression solver to solve expr within \run{expr}. Expressions are made similar to Julia or Matlab.

Example valid expr are:

- \run{A = sqrt(B)*abs(-4)}
- \run{R_4 = parll(R_1,parll(R_2,R_3))}

3.6.1 Functions

The functions currently known in problem2tex are:

> abs, asin, asinh, acos, acosh, atan, atanh, ceil, cos, cosh, exp, floor, log, log10, round, sin, sinh, sqrt, tan, tanh

the above make use of the math package for golang.
In addition, extra functions are:

- cosd(x), sind(x), tand(x)

  returns cos(x)/sin(x)/tan(x) but x value is in degrees

- acosd(x), asind(x), atand(x)

  returns acos(x)/asin(x)/atan(x) but returns the value in degrees

- dB(x)

  returns 10*log10(x)

- dbV(x)

  returns 20*log10(x)

- parll(a,b)

  returns the numeric parallel value (returns $(1/a + 1/b)^{-1}$)

  the latex printout of this function is || to make it more readable

3.6.2 More on Units

problem2tex will automatically calculate proper unit prefixes IF SI unit notation is used. For micro, "\mu" must be used instead of "u". In some situations, one might want to use a unit notation that is NOT SI proper. For example, one might want to use V/μm instead of MV/m. In this case a way to achieve this is shown with the following example...

```
\runParam{VAprime = [5,3,4,6]# \units{V/\mu m} \symbol{V_A'}}
\runParam{V_A = [6, 7, 8, 9]}
\question Given \val={VAprime} and \val={V_A}, find $L$ \\
\textbf{Solution}\\
\runSilent{VAprime = 1e6*VAprime}
\run()={L=V_A/VAprime#\units{m}}\\
```

The use of \runSilent corrects the value for VAprime (since μm is in the denominator of a unit) by multiplying the values by 1e6. In this example, "L" would default to units of "H" so it is corrected to meters using \units{m}.

The output becomes ...

1. Given $V_A' = 5\,\text{V}/\mu\text{m}$ and $V_A = 6\,\text{V}$, find L
   Solution
   $L = V_A/V_A' = (6)/(5e6) = 1.2\,\mu\text{m}$

4 License

4.1 Files created by problem2tex

All files created by problem2tex are owned by the creators of the work (that is you) and/or by the original authors of the work in cases you use derivative works.

4.2 Software License