`problem2tex`

version = 0.8.6 (2022-01-13)

David Johns,
`david.johns@icewire.ca`

Document Date - 2022-02-09

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 example.prb
```

2 Installation and Setup

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

3 Quick Start Guide

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

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.

In this first example, example01.prb is a user generated problem file and contains the following text:

```
PARAM{x = [2, 3, 4, 5]}
PARAM{y = [6, 7, 8, 9] }
PARAM{k = [8]}
Given VAL{x,=}, VAL{y,=}, and VAL{k,=} find $z = x^2+y-k$

Solution
RUN{z=x^2+y-k}
```

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=example01.tex example01.prb
```

a tex file is created.

When example01.tex is included into a document, the following output is generated:

> Given $x = 2$, $y = 6$, and $k = 8$ find $z = x^2 + y - k$
>
> Solution
> $z = x^2 + y - k = (2)^2 + (6) - (8) = 2$

In a second example, example02.prb file is created that contains the following text:

```
PARAM{V_1 = [2, 3, 4, 5]# units=V}
PARAM{R_1 = [6, 7, 8, 9]# units=k \Omega}
 Given VAL{V_1,=} and VAL{R_1,=}, find the current  $I_R = V_1/R_1$

 Solution:
RUN{I_R=V_1/R_1#units=A}
```

After running problem2tex on this file and including it into a latex document, the following output is generated:

> Given $V_1 = 2\,\mathrm{V}$ and $R_1 = 6\,\mathrm{k\Omega}$, find the current $I_R = V_1/R_1$
>
> Solution:
> $I_R = V_1/R_1 = (2)/(6e3) = 333.3\,\mu\mathrm{A}$

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 option. In addition, a parameter in CONFIG called defaultUnits can be used to set the default units depending on the first letter of the variable. (see CONFIG settings)

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

The process can be automated such that problem2tex is run and then the resulting .tex files are included into the document. An example latex file is shown below

```
\documentclass{article}
\usepackage{import}
\newlength{\currentparindent} % used to store current parindent
\newlength{\currentparskip}
\newcommand{\incProb}[1]{
    \immediate\write18{problem2tex --export=Problems/tmp/#1.tex
    --random=false Problems/#1.prb}
    \setlength{\currentparindent}{\parindent} % store current parindent
    \setlength{\currentparskip}{\parskip}
    \setlength{\parindent}{0em}
    \setlength{\parskip}{0em}
    \import{Problems/tmp/}{#1.tex}
    \setlength{\parindent}{\currentparindent} % restore parindent
    \setlength{\parskip}{\currentparskip}
}

\newcommand{\units}{\,\mathrm}
\newcommand{\skipLine}{\vspace{2ex}}

\begin{document}

\incProb{example01}

\end{document}
```

For the above example, a new command "\incProb" has been defined that is used to bring in problem example01 (it should be in the directory Problems below the main directory. The command "\incProb" first runs problem2tex using the \immediate\write18 command, next the current paragraph settings are stored, then the paragraph settings are set to no indent and no spacing, then the import of the newly generated .tex file is done and finally, the original paragraph settings are restored. Any number of \incProb commands with other .prb files can be included in the latex document.

In the above examples, all default parameters are used but it is one line change in the latex code to obtain random parameters by changing the flag -random from false true.

3.4 Syntax difference with Latex

In the above examples, there are 3 main changes from regular latex. First, the problem2tex commands do not make use of \but instead are determined by keywords that are capitalized. Second, CONFIG and PARAM statements are not printed. Finally, the paragraph and line spacing is modified from regular latex (so that non-latex users could easily generate problems as well).

Each carriage return in the .prb file results in a new paragraph. However, the spacing between paragraphs is the same as regular line spacing. In addition, if extra blank lines are included in the .prb file, they result in \skipline in the final output. The \skipline needs to be defined in the latex document and it should generate a vertical line space. So visually, a new paragraph in the final output can be achieved using a blank line. Note that more blank lines in a row will result in more spacing.

4 Commands for .prb files

There are 5 commands for problem to tex:

- CONFIG{}
- PARAM{}
- RUN{}
- VAL{}
- INCLUDE{}

4.1 CONFIG Command

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 CONFIG command. Each CONFIG command should be on a separate line with no other commands on the same line.

Example: CONFIG{random = min, fmtVal=E3, KFactor=2:6, verbose}

- 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

  - En for engineering format with n significant digits
  - Sn for scientific format with n significant digits
  - Dn for decimal format with n significant digits
  - $ for dollar format (always has 2 digits after decimal point)
  - Un for SI format (including units) with n significant digits
  - n is a digit from 1-9
  - DEFAULT IS U4

- fmtRun= - format of output value after equal in RUN command

  - Same as format above
  - DEFAULT IS U4

- fmtRun() - format of output values in bracketed numbers in RUN command

  - Same as format above
  - DEFAULT IS E4

- KFactor - Used for default set generation if PARAM sets variable to a nominal number.

  - Set KFactor example: CONFIG{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: CONFIG{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

4.2 PARAM Command

The command PARAM can be used for setting the randomly generated variables (i.e. parameters). Each PARAM command must be on a separate line with no other commands on the same line.

- PARAM{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, the symbol for var will be replaced with varLatex

- PARAM{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

- PARAM{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

4.3 RUN Command

The command RUN is 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.

- RUN{var = expr # units=units, symbol=varLatex, fmt=format }

This command will evaluate expr, assign the result to var and the units and symbol for var will be set by the options units and symbol.

The fmt option determines what is printed according to the following:

- fmt = long

  - Print out var = expr = ()expr = result

- ()expr is the expression with the values in it where the values are bracketed.
- For example, RUN{x=y+3} where y=2 would print: x=y+3=(2)+3=5
- This is the default if fmt option is not in the RUN statement

- fmt = short
  - Print out var = expr = result
  - For example, RUN{x=y+3} where y=2 would print: x=y+3=5
- fmt = equation
  - Print out var = expr
  - For example, RUN{x=y+3} where y=2 would print: x=y+3
- fmt = silent
  - Evalute the expression but do not print anything out

4.4 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 below for format choices)
  - format is optional. If not present, then the default setting for format is used which was set by CONFIG{fmtVal=...}
  - 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
- VAL format types
  - 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
  - L for only printing VAL latex symbol (not the value of VAL)
  - = for printing "latex symbol = value"
  - DEFAULT IS U4

4.5 INCLUDE Command

Include figures or graphics is an important part of many problems. To make this easier, problem2tex makes use of the INCLUDE command. The INCLUDE command syntax is

- INCLUDE{filename.ext#options}
  - filename.ext is that file name and filename extension for the graphic
- The filename extension .ext can be one of
  - .asc (an LTSpice schematic file where VAL{} can be in the schematic)
  - .svg (a scalar vector graphics file where VAL{} can be in the graphic)
  - .pdf/.png/.jpg (a fixed graphic file where no VAL{} are in graphic)
- The options are
  - width

    determines size of graphic
    default: 100
  - spaceAbove/spaceBelow/spaceHoriz

    in units of 1 "x" character
    add or subtract space (negative value reduces space)
    default: 0
  - textScale

    scale size of text when latex is drawing the text
    default: 1.0
  - svgFormat

    latex - writes text using latex
    noLatex - writes text as is with no latex for text
    default: latex

5 Other Information

5.1 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))}

5.1.1 Functions

The functions currently available 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

5.2 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 officially SI correct. 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...

```
PARAM{VAprime = [5,3,4,6]# units=V/\mu m, symbol=V_A'}
PARAM{V_A = [6, 7, 8, 9]}
Given VAL{VAprime,=} and VAL{V_A,=}, find $L$
Solution
RUN{VAprime = 1e6*VAprime#fmt=silent}
RUN{L=V_A/VAprime#units=m}
```

The use of RUN with silent format 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:

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

6 License

6.1 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.

6.2 Software License