# Example of a question with various versions generated randomly via the package fp

Tony Roberts

8:32, 8 September 2018

## 1 Example question

The following question changes every minute because I set a random seed from the number of minutes so far in this day. In practice, instead set a random seed which includes "`\the\year`" so the question changes from year to year.

1. Consider the orthogonal matrix

$$Q = \frac{1}{2}\begin{bmatrix} 1 & 1 & 1 & 1 \\ 1 & 1 & -1 & -1 \\ 1 & -1 & 1 & -1 \\ 1 & -1 & -1 & 1 \end{bmatrix}.$$

(a) Let $\vec x = (2, 2, 4, 1)$, compute $\vec u = Q\vec x$ and verify that $\|\vec x\| = \|\vec u\|$.

**Solution** $\|\vec x\|^2 = 2^2 + 2^2 + 4^2 + 1^2 = 25$.

$$\vec u = Q\vec x = \frac{1}{2}\begin{bmatrix} 1 & 1 & 1 & 1 \\ 1 & 1 & -1 & -1 \\ 1 & -1 & 1 & -1 \\ 1 & -1 & -1 & 1 \end{bmatrix}\begin{bmatrix} 2 \\ 2 \\ 4 \\ 1 \end{bmatrix} = \begin{bmatrix} 4.5 \\ -\,0.5 \\ 1.5 \\ -\,1.5 \end{bmatrix}$$

$$\implies \|\vec u\|^2 = 4.5^2 + (\,-\,0.5)^2 + 1.5^2 + (\,-\,1.5)^2 = 25 = \|\vec x\|^2.$$

(b) Let $\vec y = (-3, 1, 4, -2)$, compute $\vec v = Q\vec y$ and verify that $\|\vec y\| = \|\vec v\|$.

**Solution** $\|\vec y\|^2 = (\,-\,3)^2 + 1^2 + 4^2 + (\,-\,2)^2 = 30$.

$$\vec v = Q\vec y = \frac{1}{2}\begin{bmatrix} 1 & 1 & 1 & 1 \\ 1 & 1 & -1 & -1 \\ 1 & -1 & 1 & -1 \\ 1 & -1 & -1 & 1 \end{bmatrix}\begin{bmatrix} -3 \\ 1 \\ 4 \\ -2 \end{bmatrix} = \begin{bmatrix} 0 \\ -\,2 \\ 1 \\ -\,5 \end{bmatrix}$$

$$\implies \|\vec v\|^2 = 0^2 + (\,-\,2)^2 + 1^2 + (\,-\,5)^2 = 30 = \|\vec y\|^2.$$

(c) Use the dot product to confirm that the angle between $\vec{x}$ and $\vec{y}$ is the same as that between $\vec{u}$ and $\vec{v}$.

**Solution** $\vec{x} \cdot \vec{y} = \vec{x}^T \vec{y} = \begin{bmatrix} 2 & 2 & 4 & 1 \end{bmatrix} \begin{bmatrix} -3 \\ 1 \\ 4 \\ -2 \end{bmatrix} = 10.$

$$\vec{u} \cdot \vec{v} = \vec{u}^T \vec{v} = \begin{bmatrix} 4.5 & -0.5 & 1.5 & -1.5 \end{bmatrix} \begin{bmatrix} 0 \\ -2 \\ 1 \\ -5 \end{bmatrix} = 10 = \vec{x} \cdot \vec{y}.$$

Since these dot products are the same, and $\|\vec{u}\| = \|\vec{x}\|$ and $\|\vec{v}\| = \|\vec{y}\|$, the angle between $\vec{x}$ and $\vec{y}$ is the same as that between $\vec{u}$ and $\vec{v}$.

## 2 Useful fp information

- `\usepackage[nomessages]{fp}` in the preamble. This package was last revised in 1999.

- Also define a useful command to print any number that internally may not be an integer:

  ```
  \newcommand{\FP}[2][4]{%
      \FPeval\TempRes{clip(round(#2:#1))}%
      \FPprint\TempRes}
  ```

  `\FP\x` or `\FP{expression}` prints the variable/expression rounded to four decimal places, and clipped to remove trailing zeros; `\FP[n]...` rounds to `n` decimal places. Even if the result of a computation should be an integer, the fixed point arithmetic used by `fp` often causes error. Do most printing via this command, and thereby keep full accuracy for internal variables.

- And also useful to define this command that uses `\FP` to print a variable (not an expression) with parentheses when negative, and without parentheses when positive or zero.

  ```
  \newcommand{\FPP}[2][4]{%
      \FPifneg#2(\FP[#1]#2)\else\FP[#1]#2\fi}
  ```

- The package `fp` does fixed point arithmetic on signed decimal numbers up to $10^{18}$, with a fixed resolution of 18 decimal places.

- `\FPeval#1{#2}` assigns to `#1` the value computed by the expression `#2`. Occasionally I get a cryptic error message that is fixed by inserting an extra pair of parentheses: `\FPeval#1{(#2)}`.

- Defined infix operations for evaluation are: `+`, `-`, `*`, `/`, `^` for add, sub, mul, div, pow. The unary minus is not defined: use `neg()`.

- Defined functions include abs, neg, min, max, round, trunc, clip, exp, ln, pow, root, sin, cos, tan, cot, arcsin, arccos, arctan, arccot

  Note: trig functions use radians; `root(a,b)` $= \sqrt[a]{b}$ so a square-root is `root(2,b)`; `pow(a,b)` $= b^a =$ `exp(a*ln(b))` so fails for negative `b` (even if `a` $= 2$); `round(a:n)` has the value of `a` rounded to `n` decimal places; `clip(a)` has the value of `a` but with trailing zeros clipped.

- Defined constants are `pi` and `e`.

- `\FPrandom#1` assigns to `#1` a random number between 0 and 1.

  When scaling and rounding to include random negative integers, sometimes get `-0` which appears poorly: however, `\FP...` prints the value `-0` properly as `0`.

- `\FPseed=#1` where `#1` is some string of digits sets the seed for the random number generator. For example, `\FPseed=67\the\year` is this year the same as `\FPseed=672018` and so sets the seed unique to the problem via 67, and unique to the year via `\the\year`.

- These are some of the conditional statements:

  - `\FPifneg#1 ...\else...\fi` tests `#1` $< 0$ ?
  - `\FPiflt#1#2...\else...\fi` tests `#1` $<$ `#2` ?
  - `\FPifeq#1#2...\else...\fi` tests `#1` $=$ `#2` ?
  - `\FPifgt#1#2...\else...\fi` tests `#1` $>$ `#2` ?

## 3 More examples

**To iterate** One may use `\foreach` from `\usepackage{pgffor}` (or `pgfplots`). For example, the following iterates the map $x_k = \cos x_{k-1}$ ten times via the mysterious magic of `\xdef`. The following code gives "From $x_0 = 0$, iteration gives $x_{10} = 0.7314$."

```
\FPeval\xk{0}
\foreach \k in {1,2,...,10}
    {\FPeval\xNext{cos(\xk)}\xdef\xk{\xNext}}
From \(x_0=0\), iteration gives \(x_{10}=\FP\xk\).
```

One could just typeset some of the iterates from within the loop: "the iterations are $x_0 = 0$, $x_1 = 1$, $x_2 = 0.5403$, $x_3 = 0.8576$, and so on to $x_{20} = 0.7389$." Typeset this with the code

```
\FPeval\xk{0}the iterations are \(x_0=\FP\xk\)%
\foreach \k in {1,2,...,20}%
    {\FPeval\xNext{cos(\xk)}\xdef\xk{\xNext}%
     \ifnum\k<4, \(x_{\k}=\FP\xk\)\fi}%end-for
    , and so on to  \(x_{20}=\FP\xk\).
```