EMAT10001 Exercise Sheet 17.

Conor Houghton 2014-02-26

Exercise sheet

The difference between the work sheet and the exercise sheet is that the solutions to the exercise sheet won't be given and the problems are designed to be more suited to working on on your own, though you are free to discuss them in the work shop if you finish the work sheet problems. Selected problems from the exercise sheet will be requested as part of the continual assessment portfolio.

1. Establish that
$$\frac{1}{\pi}\int_{-\pi}^{\pi}\cos mt\ \ \cos ntdt=\delta_{mn} \tag{1}$$
for integers n and m.

2. Read the wikipedia page for the Fast Fourier Transform.

3. Using `gnuplot` or whatever plot the function, written here in a Fourier series form
$$f(t)=\sin(t)+0.1\sin(10t)+0.2\cos(11t) \tag{2}$$
and compare it to $g(t)=\sin(t)$. The original $f(t)$ has details at two scales, around 2π and around $2\pi/10$. $g(t)$ is obtained from $f(t)$ by dropping the higher frequency Fourier modes. Changing a function by removing some of the Fourier modes is called *band pass filtering*.

4. Find the Fourier series for $\sin^3 t$; a quick way to do this is to regard it as a trigonometry problem, rather than a Fourier series problem, that is use the trigonometric identities to express it in terms of sines and cosines, rather than doing all the integrals: so start by writing $\sin^3 t=\sin^2 t\sin t$ and then write $\sin^2 t$ in terms of $\cos 2t$.

Challenge

A well known Turing tarpit has been extended to give it function-call like functionality in a way that adheres to the original philosophy of the base language.

In this news language there is a list of 'brains', objects that each have a tape of unsigned integer variables, a pointer to the tape location, a list of commands and a pointer to the current command.

At any time only one brain is active, as describe below there is a command to change which brain is active. If the pointer is pointing beyond the end of the active brain's command list, the next command in the program is appended. The command at the command pointer is then executed if appropriate. It may not be appropriate, there are two modes, an execution mode where commands are executed and a non-execution mode where they are appended but not executed; commands are also not executed when skipping from a `[` to a matching `]`. Either way the command list pointer is then advanced.

Most of the commands are the same as in the base language

- `>` advance the tape pointer by one.
- `<` move the tape pointer back one, unless it already points to the start of the tape.
- `+` increase the value at the current tape locations by one.
- `-` decrease the value at the current tape location by one, unless it is already zero.
- `[` if the current tape value is non-zero advance the command pointer to the *matching* `]`, this may mean appending commands from the program.
- `]` if the current tape value is zero move the command pointer back to the matching `[`
- `.` output the current tape value as ASCII
- `,` input from the keyboard to the current tape location

There are three new commands

- `!` toggles between execution and non-execution mode. Uniquely this command is not appended to the command list when executed, also, uniquely, it is executed even in non-execution mode.
- } changes which brain is active by moving forward along the list of brains the number of steps dictated by the current tape location. It copies the values of the current tape location and its two immediate neighbours to the current tape location in the destination brain.
- { the same as } but it moves backwards, unless the execution brain is the first brain.

The puzzle is to work out the output from this program, currently the only program ever written in this language:

```
+}> >++++[>+++<-]>[<++++>-]![>[>+>+>+< < <-]> >[< <+> >-]<.> >[< <-> >-]< <
< <!< < <{+}> >++++++++++< <![> >.< <!{->+<}]+}]> >+++++[<+<-}+}> >-]
```