Calculus in Lua

05 Oct 2015

Filed Under: Programming

Tagged: Lua

Lately I unearthed a file named 'maths.lua' which I've had sitting around on my hard drives for about two years. It contains a very small library of maths functions, most of which are to do with calculus. I remember writing these functions after learning numerical methods for integration and differentiation and while I was enthusiastically getting to grips with the Lua programming language[1]. It was satisfying to translate the knowledge I had picked up into code rather than just having it sit in my head waiting to be remembered badly when the exam came around, assuming there even was a question involving them, which there probably wasn't.

Here are some cool things about functions in Lua:

- They can output multiple return values.
- They can receive a variable number of arguments.
- They can be anonymous.
- They can be defined within other functions and can access variables of their enclosing functions, which is called 'lexical scoping'.
- Tail calls are done properly, meaning that if a function returns by calling another function, the program will not go back to the first function after the second is complete only to exit it. Hence tail calls can't overflow the stack.
- Finally, they're first-class values, so they can be stored in variables, passed as arguments to other functions and returned by other functions as output.

All the calculus functions I wrote take advantage of the last feature on that list, as it makes the conceptual jump from 'mathematical functions' to 'programming language functions' small, where in other languages like C++ the gap can be quite large. This is easiest to show by example. This Lua function takes another function f(x) and returns a new function which, when called, will return an approximate value for the derivative f'(x):

```lua
-- Approximates the derivative of a function by obtaining
-- the forward difference.
-- f: A function of x (e.g. f(x) = x^2).
-- delta: The interval used in the approximation.
approx_derivative = function (f, delta)
  local delta = delta or 1e-4
  return function (x)
           return (f(x + delta) - f(x))/delta
         end
end
```

To show this in action, here's a function which cubes its input value.

```
function cube(x)
  return x*x*x;
end
```

`print(cube(2))` gives us 8. `fdash = approx_derivative(cube)` stores the function generated by `approx_derivative` in `fdash`. Now I've done that, I can call `fdash(x)`, which should return $3x^2$.

```
> print(fdash(2))
12.000600010023
```

3 * 2 * 2 = 12. The output isn't quite right, because it's an approximate, but it's close enough.

The reason I dug this file up is because our coursework for this year's Applied Mathematics module involves using the classical fourth-order Runge-Kutta (RK4) algorithm to numerically solve some ordinary differential equations as part of a dynamics solution. And I distinctly remembered writing a Lua implementation of the algorithm, even though I didn't remember at all what the algorithm was or even what it did. And lo, here it is:

```
-- 'classical' 4th-order Runge-Kutta, or 'RK4'
runge_kutta = function (f, timestep)
  local timestep = timestep or 0.1
  return function (start_x, start_y, time)
           local x = start_x
           local y = start_y
           local t = time
           -- loop until i >= t
           for i = 0, t, timestep do
             local k1 = f(x, y)
             local k2 = f(x + (timestep/2), y + (timestep/2)
             local k3 = f(x + (timestep/2), y + (timestep/2)
             local k4 = f(x + timestep, y + timestep*k3)
             y = y + (timestep/6)*(k1 + 2*k2 + 2*k3 + k4)
             x = x + timestep
           end
           return y
         end
end
```

Having done this already gives me a bit of a head start on my coursework (hopefully), but I might have to adapt it to another language. It's cool that it's there, anyway. Thanks, past me!

It might seem like this blog post is ending rather abruptly, but I really need to get on with other things - I'll tidy the maths.lua file up and post it on Gist or something soonish as it could be a useful resource for others looking to learn Lua.

Note to self: posts involving maths will be much nicer once I've set up MathJax. I'd better get on with that.

1. I'm still pretty keen on it, I just don't have much use for it at the moment. ↩

0 Comments **moments of inertia** 1 Login

Recommend Share Sort by Oldest

Start the discussion…

LOG IN WITH

OR SIGN UP WITH DISQUS ?

Name

Be the first to comment.

Subscribe Add Disqus to your siteAdd DisqusAdd