Astro 585: HW 5

Codename: The Maxwell-Jüttner Distribution

February 27, 2014

My git repository is here: `https://github.com/hsgg/astro585`, clone URL `https://github.com/hsgg/astro585.git`.

1 Density Matrix

This was done in pair coding.

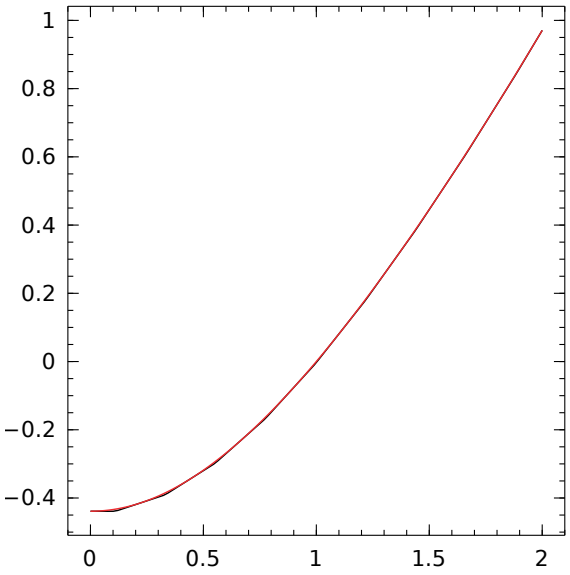
2 Lookup tables

```
#!/usr/bin/env julia

require("../Astro585/Lab5/HW5_Q2_lookup_table.jl")
require("../Astro585/Lab5/HW5_Q2_ecc_anom.jl")



# 2a
# 'lookup()' with an abstract type won't be able to make use of any of the
# internals of either of the lookup table types, so it doesn't need to know
# them. All it needs to know is which function 'lookup()' to call. When the
# first argument is passed, julia will know its type, and can choose based on
# that.

# 2b
function more_complex(x::Real)
    I, E = quadgk(atan, -1., x)
    I
end
function more_complex(x::Array)
    y = similar(x)
    for i in 1:length(x)
        y[i] = more_complex(x[i])
    end
    return y
end
```

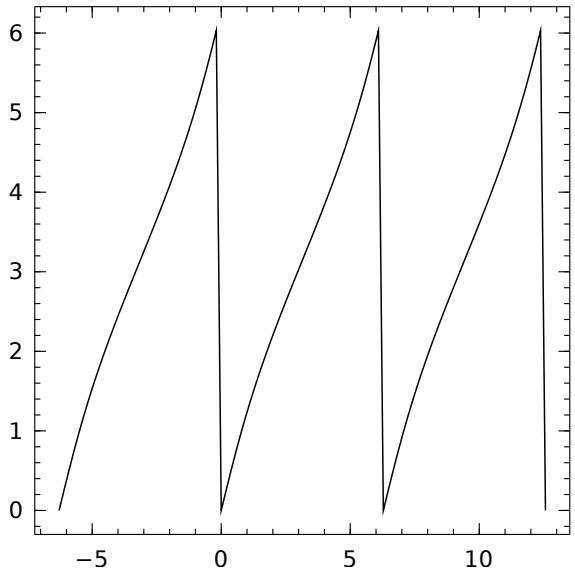
```
lookup_table = make_table_linear(more_complex, 0., 2., 10)

# 2c
using Winston
x = linspace(0, 2, 100)
ylookup = lookup(lookup_table, x)
yfunc = more_complex(x)
p = plot(x, ylookup, x, yfunc)
file("2c_test.pdf")
# yep, agree pretty well, only not quite as smooth for the interpolated one

# 2d
x = linspace(0, 2, 10000)
lookup_table = make_table_linear(more_complex, 0., 2., 10)
println("Lookup table:      ", @elapsed(lookup(lookup_table, x)), " seconds")
println("Direct evaluation: ", @elapsed(more_complex(x)), " seconds")
# Lookup table:      0.038435129 seconds
# Direct evaluation: 0.626747923 seconds
# Nice, lookup table is about 20 times faster.

# 2d, page 3
ecc = 0.25
x = linspace(-2pi, 4pi, 100)
y = ecc_anom(x, ecc)
plot(x, y)
file("2d_ecc_anom.pdf")
# The lookup table will probably have problems at multiples of 2pi.

# 2e
lookup_table = make_table_linear(x -> ecc_anom(x, ecc), 0., 2pi, 128)
x = linspace(0., 2pi, 1000)
y = ecc_anom(x, ecc)
ylook = lookup(lookup_table, x)
plot(x, y, x, ylook)
file("2e_ecc_anom_poor.pdf")
# yep, shitty at n*2pi. Discontinuities suck for an interpolation scheme that
# assumes they don't exist.

# 2f
lookup_table = make_table_linear(x -> ecc_anom(x, ecc), x -> decc_anom_dM(x, ecc), 0., 2pi, 128)
ylook2 = lookup(lookup_table, y)
plot(x, ylook, y, ylook2)
file("2f_ecc_anom_stillpoor.pdf")
# yep, still bad. what even is the correct value of the derivative at the
# discontinuities?
```

```
# 2g
# hm, probably not.
lookup_table = make_table_quadratic(x -> ecc_anom(x, ecc), 0., 2pi, 128)
ylook2 = lookup(lookup_table, y)
plot(x, ylook, y, ylook2)
file("2g_ecc_anom_stillpoor_quadratic.pdf")
# yeah, no better

# 2h
# The discontinuities a the edges of the interval were problematic. The
# interpolation algorithms were not designed to handle such cases. Hence, be
# careful, and understand whether they work with yur particular function.
```

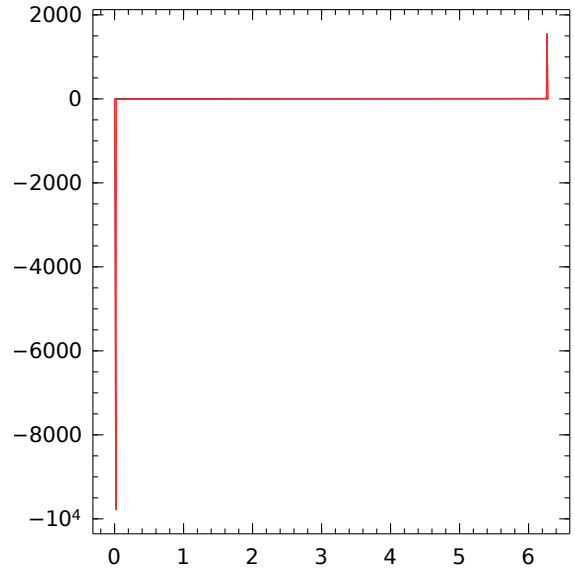
The simple graph for 2c:



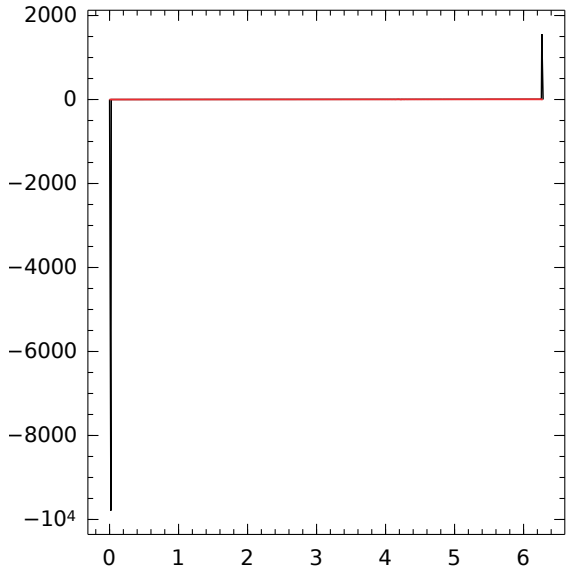
ecc_anom for 2d:



ecc_anom for 2e, when it gets ugly:



ecc_anom for 2f, when it remains ugly:



ecc_anom for 2g, when it quadratic interpolation doesn't help: