Day 3 Problems (Jason)

Easy Exercises

Define a function `GiveFit` like `FindFit` but it instead returns the fully fitted function with the coefficients fully substituted into the fitted function. Example: if
$\texttt{theCoeffs = FindFit[data, a}\,e^{b\,x} + \texttt{c, \{a, b, c\}, x]}$
returns
$\{a \to 1.24367 \times 10^{-6},\ b \to 0.482733,\ c \to 0.000500732\}$
then $\texttt{GiveFit[data, a}\,e^{b\,x} + \texttt{c, \{a, b, c\}, x]}$ should return
$0.000500732 + 1.24367 \times 10^{-6}\, e^{0.482733\,x}$

Create a function `children` which returns the list of the children of the argument without evaluation. For instance `children[h[a,b]]->{a,b}` and `children[2+2]->{2,2}`

Given a list of functions, find the multiplier in the power x of any terms at any level (hint generalized use of cases.) Example input : $\texttt{coefPowerOfX[\{a}\,e^{-i\,x} + 3\texttt{, Sin[}x^2\texttt{], Sin[}2^{3\,x}\texttt{], f\}]} \to \{-i,\ 3\}$

Look at the documentation for `Optional`, and change `coefPowerOfX` so it can handle the following case and return $\texttt{coefPowerOfX[\{a}\,e^{-i\,x} + 3\texttt{, Sin[}x^2\texttt{], Sin[}2^{x}\texttt{], f\}]} \to \{-i,\ 1\}$

Define a function `ReplaceDerivatives[expr, replacements]` which will replace all derivative terms in `expr` with the replacements in `replacements`. For example
$\texttt{ReplaceDerivatives[f}'\texttt{[x]} + \texttt{k f}''\texttt{[x], f[x]} \to x^3\texttt{]}$ should yield $6\,k\,x + 3\,x^2$

Create a simple `Manipulate` exploring `BessleJ[n,x]`

Create a function which will replace every repeated element in a list by it's index in the list. Example
`RepeatedToIndex[{a,b,a,c,d,e,c,f,f}] → {1,b,3,4,d,e,7,8,9}`

Create a table verifying the orthogonality of the ChebyshevT polynomials
$\int_{-1}^{1} \frac{\texttt{ChebyshevT[n,x] ChebyshevT[m,x]}}{\sqrt{1-x^2}}\, dx$ for n and m running over 1...6 by distributing these integrals in parallel.

Medium Exercise

Resistor Pairings

A power supply stabilizer chip supplies a stable output voltage dependent on two resistances:

$$v_{\text{out}} == 1.22 \frac{(R_1 + R_2)}{R_2}$$

Typical resistors you can order only come in the following standard values:

```
OhmT = {10.0, 10.2, 10.5, 10.7, 11.0, 11.3, 11.5, 11.8, 12.1, 12.4, 12.7, 13.0, 13.3,
    13.7, 14.0, 14.3, 14.7, 15.0, 15.4, 15.8, 16.2, 16.5, 16.9, 17.4, 17.8, 18.2, 18.7,
    19.1, 19.6, 20.0, 20.5, 21.0, 21.5, 22.1, 22.6, 23.2, 23.7, 24.3, 24.9, 25.5, 26.1,
    26.7, 27.4, 28.0, 28.7, 29.4, 30.1, 30.9, 31.6, 32.4, 33.2, 34.0, 34.8, 35.7, 36.5,
    37.4, 38.3, 39.2, 40.2, 41.2, 42.2, 43.2, 44.2, 45.3, 46.4, 47.5, 48.7, 49.9, 51.1,
    52.3, 53.6, 54.9, 56.2, 57.6, 59.0, 60.4, 61.9, 63.4, 64.9, 66.5, 68.1, 69.8, 71.5,
    73.2, 75.0, 76.8, 78.7, 80.6, 82.5, 84.5, 86.6, 88.7, 90.9, 93.1, 95.3, 97.6};
```

1. Write a function **`closestPair[vout]`** which will return **`{vclose, r1, r2}`** where **`vclose`** is the closest voltage to **`vout`** and **`r1`** and **`r2`** are the resistors which will produce this close voltage.

Compute the resistors which give the closest output to 5.13V and 5.14V

2. Graph **`vclose`** and **`vout`** over 5.0V to 5.3V

(Anecdotal note: I actually used this in finding the best pairing for over volting a raspberry pi 2 so it didn’t need an external power hub to drive a USB 3G stick modem.)

Notation Exercises

Create a notation $\underset{\sim}{\mathtt{v}}$ to represent `Vector[v]`

Create a notation for `[a, b]`$_{\mathtt{c}}$ to represent `Commutator[a,b]`

Create a notation for some function you use in physics. (There are several things I haven’t explained yet so if you have problems then wait for further lectures, information here.)

Challenging Exercises

Create a Non-commutative multiply function which distributes over addition, and constants can be factored out

Using your Non-commutative multiply function implement a canonicalization for a cluster of creation / annihilation operators which commute according to:

$a_i \cdot a_j = -a_i \cdot a_j$ if $i \neq j$

$a^\dagger{}_i \cdot a^\dagger{}_j = -a^\dagger{}_i \cdot a^\dagger{}_j$ if $i \neq j$

$a_i \cdot a^\dagger{}_j = \delta_{ij} - a^\dagger{}_j \cdot a_i$

Write an external C program to add the machine sized integers from iman to imax and link it into *Mathematica* as an external library. So from *Mathematica* you would call this via `SumRange[20,40]`. (The *Mathematica* code for this is of course trivially `Sum[i,{i,imin,imax}]` but the exercise is really just working through the *Mathematica* tutorial steps. The actual C program is almost trivial.)

Create a Demonstration for the demonstrations website

Another Medium Exercise

Differentation

- Implement your own version of partial differentiation, including linearity, chain and product rules.
- Add the derivatives of Sin, Cos, Tan, Exp, Log