THE FINAL ASSIGNMENT: FITTING TO THE LUX DATA
Code Health Check-Up

- Everyone downloaded & installed ROOT?
- Sean and I will come around now
  - Windows, Mac, Linux/Unix help, for everyone
- We will also show how to compile the “rootNest” code and show how to run it
- Due date extended: until Wed. May the 4th
  - Last name A-R you do NR, while S-Z is ER
  - Project due end of day on Monday, May 9th
The Goal

- Data points provided on the course website
- Find model with a reduced $\chi^2 < 1.5$ please
- 5 column text files to work with: x (S1c), y (log of S2c/S1c), error, width, error
More Details

- You will fit both the band mean plus the band width, while varying the parameters within the built-in physics model (NEST)
  - Chi^2/DOF less than 1.50 for mean & width
  - Use conjugate gradient method or brute-force

- **BONUS:** Use different models I will give you OR come up with your own, better!!
Gradient descent is based on the observation that if the multi-variable function $F(x)$ is defined and differentiable in a neighborhood of a point $a$, then $F(x)$ decreases fastest if one goes from $a$ in the direction of the negative gradient of $F$ at $a$, $-\nabla F(a)$. It follows that, if

$$b = a - \gamma \nabla F(a)$$

for $\gamma$ small enough, then $F(a) \geq F(b)$. In other words, the term $\gamma \nabla F(a)$ is subtracted from $a$ because we want to move against the gradient, namely down toward the minimum. With this observation in mind, one starts with a guess $x_0$ for a local minimum of $F$, and considers the sequence $x_0, x_1, x_2, \ldots$ such that

$$x_{n+1} = x_n - \gamma_n \nabla F(x_n), \quad n \geq 0.$$ 

We have

$$F(x_0) \geq F(x_1) \geq F(x_2) \geq \cdots,$$

so hopefully the sequence $(x_n)$ converges to the desired local minimum. Note that the value of the step size $\gamma$ is allowed to change at every iteration. With certain assumptions on the function $F$ (for example, $F$ convex and $\nabla F$ Lipschitz) and particular choices of $\gamma$ (e.g., chosen via a line search that satisfies the Wolfe conditions), convergence to a local minimum can be guaranteed. When the function $F$ is convex, all local minima are also global minima, so in this case gradient descent can converge to the global solution.
Parameters

- NR: TIB, $N_{ex}/N_i$, biexcitonic quenching (2)
- ER: TIB, $N_{ex}/N_i$, corrections at hi, low E (3)
- Same width parameter for each (‘C’ in $F_r$)

For the extra credit

- NR: Compare power law $L_y$ and exponential $Q_y$ with versions with flattening/roll-off
- ER: Sigmoid or 1-exp for L or Q (in log(E) ?)
- If you do both for both then in addition to +1% you get 0-forgiveness for 1 past HW
Where to Find Them in rootNest.cc

- **NR:**
  - Lines 164, 167, 230, 253, 254
  - Replace 0.482, power law, 0.0075, 13.2, and 0.111 respectively

- **ER:**
  - Lines 165 (x2), 169 (x2), 178, 234
  - Replace linear density function and either 1.5 or 0.19, replace power law and choose 0.035, 4, or 0.13, 45.836 or 0.203 free, and 0.0075
Progress Check-Up

- Both current assignment and final project
- Problems? Ask now, last in-class chance.
Corrections and Suggestions

- <1.3 keV if’s (overrides work)
- Number of free parameters fix
  - Does not require work restart
  - Make sure total # of data points OK
- Remember width too: 3 changes and new sum^2-diff (initialize!)
- Windows: Visual C plus more
- spEff and g1_gas (two places)
  - If you were looping over my code
- --cflags versus --ldflags (version)
- Faster: use L_y and Q_y values!

Email Sean S-B

Follow-up email
Extra Credit / Bonus

ER line 243 (omit kludge). NR line 247 onward

- NR: power law fit for total $N_{ph}$ and total $N_{e-}$
- ER: sigmoidal fit for $Q_y$, while $L_y = 73 - Q_y$
  - If $L_y < 0$ then make it zero of course
  - $Q_y$ is just $N_{e-} / E$ and $L_y$ is $N_{ph} / E$
- Still $F_r$ (so NR 7 free, ER 5 free)
- Define a total number of quanta
  - $N_q = N_{ph} + N_{e-}$ from the formulae =>
- Need to do fluctuations still, so consider $N_{i} = N_{e-} / (1 + a)$, where $a_{NR} = 0.5$ and $a_{ER} = 0.2$
- $R = 1 - (Q_y * E) / N_i = 1 - N_{e-} / N_i$. Recalc $N_{e-}$

$N_{ph}^{NR} = A * X^B + C$  $N_{e-}^{NR} = A * X^B + C$

<p>| | | |</p>
<table>
<thead>
<tr>
<th></th>
<th></th>
<th></th>
</tr>
</thead>
<tbody>
<tr>
<td>A</td>
<td>4.2482</td>
<td>A</td>
</tr>
<tr>
<td>B</td>
<td>1.2687</td>
<td>B</td>
</tr>
<tr>
<td>C</td>
<td>0.0000</td>
<td>C</td>
</tr>
</tbody>
</table>

$X$ or $x$ Energy in keV

$Q_y^{ER} = m4 + (m1 - m4) / (1 + (x/m3)^m2)$

<table>
<thead>
<tr>
<th>Value</th>
<th></th>
</tr>
</thead>
<tbody>
<tr>
<td>m1</td>
<td>74.629</td>
</tr>
<tr>
<td>m2</td>
<td>1.6298</td>
</tr>
<tr>
<td>m3</td>
<td>2.0731</td>
</tr>
<tr>
<td>m4</td>
<td>24.797</td>
</tr>
</tbody>
</table>

Use for $F_r$

$N_{ph} = N_q - N_{e-}$

Can skip $N_{ex}$