**Curve Fitting Exercise in MATLAB**

**Objectives:**

Scientists and engineers often try to quantitatively understand their experiments and the systems they study, in order to design solutions to various problems or to understand underlying mechanisms. A key skill for quantitative data analysis involves fitting models to data, to understand whether an underlying phenomenon might explain a particular experimental result. In this lab, you will learn to fit mathematical models (equations) to data.

In particular, you will learn to:

1. Fit a curve to data and determining goodness of fit
2. Use the function fminsearch in MATLAB to minimize a function
3. Understand vocabulary used to describe model fits to data
4. Use simple theory about model fitting to select the best model for a data set

**Definitions:**

**Polynomial:**A polynomial is of the form

 *f(x) = anxn + an-1xn-1 + ... + a1x + a0* Eq(1)

Where **n** is the **order** of the polynomial.

Examples: f (x) = 2x2 - 4x + 10 (Order 2)

 f (x) = 9 (Order 0)

In the following definitions, the “equation” can be a polynomial but also any other equation.

**Parameters**

The values of the constants in an equation. For example, [*an, an-1, …a1, a0*] are the parameters of an nth-order polynomial, [2, -4, 10] are the parameters of the 2nd order example above, and [C, τ] are the parameters of the equation f(x) = C\*exp(-x/). When parameters are constants that multiply terms (such as “C” but not “” in the last example), we also call them **coefficients**.

**Curve fitting:**

Fitting an equation to a set of data points. This is used to test how well a quantitative model fits the data. That is, if you have a data set (**x**,**y**), where **x** and **y** are vectors, then curve fitting is finding an equation f(**x**), so that f(x(i)) is close to y(i) for all indices i in the vectors.

**Residual:**

For each data point, its residual is the difference between the actual data point and the value of the equation used to model the data. That is, the residual is f(x(i)) –y(i).

**Least squares fit:**

The parameters of an equation that give the lowest value for the sum of the squares of all of the residuals. That is, the least squares fit finds the parameter values **P** that minimize the function sumerror = sum(f(**P**,x(i)) –y(i)), where f(**P**,x(i)) is the equation with parameters P calculated at the point x(i). For example, if the equation is a 2nd order polynomial and P is [2,-4,10], then f(**P**,x(i)) = 2(x(i))2 - 4x(i) + 10.

**Motivation for the lab:**

Most drugs have therapeutic benefits when the drug is above a “therapeutic threshold” but toxic side effects above a “toxic threshold”. Dosing protocols are intended to maintain the level of the drug in the patient’s blood within a “therapeutic window” that between these two thresholds. However, we could greatly extend the therapeutic window if we could reduce the side effects by targeting a drug only to the cells that need them. Current research in drug delivery combines pharmaceutical research with bionanotechnology to incorporate both the drug and a targeting molecule into a nanoparticle, which is a particle less than 1 micron in diameter, (typically a few tens or hundreds of nanometers.) The targeting molecule binds to receptors that are preferentially expressed on target cells in a target organ, thus concentrating the nanoparticles to these cells. Nanoparticles also require a mechanism for releasing the drug once bound to or internalized by these cells.

To understand the entire delivery process, a researcher might optimize various processes involved in drug delivery by using a number of *in vitro* and *in vivo* assays. These might include maximizing the binding of nanoparticles to target cells, minimizing the binding to control (nontarget) cells, and minimizing the rate of clearance of the nanoparticles in an animal. It might also be desirable to optimize the rate of release of the drug given the other processes. That is, the nanoparticle should bind to its target cells before much of the drug is released, and then release drug efficiently. Using mathematical models for each of these processes individually and together can help an engineer optimize the nanoparticle design so the entire delivery process is as efficient as possible and localized drug concentrations are kept within a therapeutic window. In today’s assignment, you will test mathematical models for just one part of this process - an *in vitro* experiment of drug release from a nanoparticle. (“Nanosystems for Simultaneous Imaging and Drug Delivery to T Cells”, by Fahmy et al, AAPS Journal, 2007 vol 9.)

**Lab exercise:**

Fahmy et al measured the total amount of drug released from a nanoparticle over time. Some of their results are provided in the table below:

Day Micrograms drug released

0 2.5

0.25 3.6

0.5 5.3

1 9.5

2 14.0

3 16.5

4 18.8

5 21.5

6 23.2

8 26.8

10 28.4

*At first, we are not seeking to explain why the data behaves as it does. Instead, we are simply seeking to describe the behavior mathematically. This kind of modeling is called “****black box****” modeling, with the idea that you cannot see through the model box to understand the underlying process. It is also called an “****empirical****” model, meaning it is derived from direct observation (the data) rather than by theory about the underlying mechanism (for example, a theory about how the drug is released, and thus what mathematical equation would describe that process.) As long as an empirical model describes data well, it can be used to predict what happens when this process is integrated into a larger system.*

1. Write a function that takes the data (in whatever form you want) as an input, fits the data with a straight line using polyfit and polyval, and returns the parameters and the model values at the datapoints. Plot both the model and the data on the same plot. **Include this plot and the parameters in your lab write up.**
2. Modify the function created in question 1 to compute and return the residuals. Plot the residuals of the model as a bar graph. **Include this plot in your lab write up. (**Hint: Check out the command “bar”)
3. Compute the sum of the squares of all of the residuals. **Include the computed value in your lab write up.**
4. Modify the main program and/or the function to fit the data using a **2nd** and **3rd** order polynomial as well. **Include the plots, residuals, and sum of squares of residuals in your write-up**. Note: if you already know how to use flow control, you can create an iteration loop to fit the given data with a 1st , 2nd and a 3rd order polynomial, plot the raw data and the model for each of them, and store the residuals for each of the models.
5. What is the order of the polynomial that gives the best fit in terms of the lowest sum of square of residuals?

The given data can also be modeled using the following **exponential** equation:

$$Y\left(t\right)=C\left(1-e^{-at}\right)$$

This equation models a situation where a system goes from a zero value to the maximum value $C$ with a rate constant $a$. This time, the parameters $C$ and $a$ have a conceptual meaning (maximum drug released and rate constant for the release) and are derived from a model that actually seeks to model the underlying processes. Because of this, this model is what is called a **gray-box** (partially transparent), or **parametric** (parameters have physical meaning) model.

1. Using MATLAB, fit the same data to this new model, using the initial estimates $C=10, a=2$. Because this equation is not a polynomial function, you cannot use polyfit, and will instead use a more generic minimization function called fminsearch (see hints at end). Plot the best model prediction and the data on the same plot. **Include this plot in your lab write up**. **What are the best-fit parameters?**

There is a principle called Occam's razor, that states that the simplest explanation that explains a phenomenon is likely the correct one. In modeling, this is taken to mean that the equation with the fewest parameters that gives the best fit to a data is considered the best model. There are several theories quantifying how much better a fit should be to justify an additional parameter that are beyond the scope of this exercise.

1. **Answer the following:** Which of your four models should you reject using Occam’s razor, and which ones are valid choices? If the number of parameters is more important than the best fit, which model is best? If the best fit is most important, which model is best?

One of the most valuable things a model can do is to predict the behavior of the system in conditions not originally measured. We held out on you until now, giving you only data through 10 days. However, the published data goes through 21 days:

Day Micrograms

12 28.4

16 28.5

21 29.5

1. Your models can be used to predict what will happen between days 10 and 21: *Without refitting the models*, calculate the values of your 4 models at days, and add this to your graphs, along with the complete data. **Include these plots in your write up and answer the following:** Which of the models is the most predictive through 21 days? How does this conclusion relate to your analysis in part 7?

**IMPORTANT MUST-READ HINTS: (**After this lab,you are responsible for **understanding** how using fminsearch helps you with a curve fit.**)**

Implementation of fminsearch in MATLAB:

What fminsearch does: Starting with your initial guesses for the parameters (you pass these as input), fminsearch calls a function that uses those parameters (you give the name of this function as input). This function is generally termed an “**objective function**” and must produce a single scalar value as output (called the value of the objective function). In general, you will write the objective function yourself for your particular minimization or optimization problem. In each loop, the fminsearch function changes the parameters slightly and determines whether the value of the objective function increased or decreased, and keeps the new parameter in the case of decrease. This iteration (loop) continues until it determines that it has found a (local) minimum point for the objective function, and then returns the final parameter values (and the final value of the objective function, if you desire).

To use fminsearch to fit an equation to data, you write an objective function that calculates the equation you are trying to fit, using the parameters it is passed, for each point (e.g. each time point) in your data set. It then compares these model values to the experimental data values, to get the residuals, calculates the sum of the square of the residuals, and returns this value. Thus, the objective function is simply the sum of the square of the residuals between the model and the data. Makes sense, because this is what you want to minimize.

Since you need to compare things to the data in so many functions, it is convenient to store it as a ‘**global**’ variable, which means all your functions can see it, without you having to pass it to them. This is OK to do since none of your functions alter the value of this actual data, so there is no chance for confusion.