Course Review

What you hopefully have learned:

1. How to navigate inside MIT computer system:
Athena, UNIX, emacs etc. (GCR)
2. General ideas about programming (GCR):
    - formulating the problem, “coding” in English
    - translation into computer language
    - editing
    - compiling + debugging
    - running
3. Quite a bit of knowledge of C (GCR, midterm).

Course Review

4. MATLAB:

- Matlab = matrix laboratory, matrix oriented.
- Any variable is an array by default, thus almost no declarations. All variables are by default double
- High level language:
- (i) quick and easy coding
- (ii) lots of tools (Spectral Analysis, Image Processing, Signal Processing, Financial, Symbolic Math etc.)
- (iii) relatively slow

Course Review

- Translator - interpreter, reads and executes line after line, but All Matlab functions are precompiled.
- One (YOU) may add extra functions by creating M-files.
- Language structure is similar to C:
  - MATLAB supports variables, arrays, structures, subroutines, files, flow of control structures
  -MATLAB does NOT support pointers and does not require variable declarations

Dealing with Matrices in Matlab

• Matlab has all standard operations & functions implemented for matrices (& vectors).

• ***Standard math. functions of matrices*** operate in array sense (act on each matrix entry independently: exp(A), sin(A), sqrt(A) = A.^0.5

>> B = exp(A)

B(i,j) = exp(A(i,j))

• Any matrix multiplication/division or rising matrix into some power may be both matrix and array operation.

Dealing with Matrices in Matlab

• A*B - matrix product of n x m and m x p matrices, but A.*B - array (element by element) product of two n x m matrices.

• B=sqrt(A) → B_{ij}=sqrt(A_{ij}), A&B - any matrices, but B=A^0.5 → A=B*B and A&B should be square.

• Submatrices: /

`A(1:4,3)` - column vector, first 4 elements of the 3-d column of A.

`A(:,3)` - the 3-d column of A

`A(:,[2 4])` - 2 columns of A: 2-d & 4-th.

Dealing with Matrices, Examples

```
>> C = A + B;
C(k,l) = A(k,l) + B(k,l)

>> C = A*B;
C(k,l) = A(k,m) * B(m,l)

>> C = A.*B
C(k,l) = A(k,l)*B(k,l)

>> C = A^alpha;
>> C = A.^alpha;
C(k,l) = A(k,l)^alpha
```

Matrix multiplication, summation over the repeating index is implied.

Element-wise (array) operation

Course Review

I hope you've learned the basic linear algebra and it's Matlab implications:

1. Matrix(vector) addition/subtraction & multiplication. Say, what is the difference between a*b' and a'*b, where a and b are vectors?
2. Colon notation and operating with submatrices. Remember, lots of summation, multiplication etc. loops may be eliminated by using the colon notation (see solution to hw8-10 for example).
3. Basic Matlab syntax: Given you know C, you may figure out many Matlab properties by analogy, but still you are expected to have a clear idea about:

Course Review

- writing functions in Matlab
- flow of control statements
- How to use desktop & maintain the workspace.
- How to deal with “black boxes”:

    i) help FUNCTIONNAME

    ii) what does the function do?

    iii) what’s in and what’s out?

    iv) how to prepare what’s in (usually vectors and matrices)

    v) ideally: what’s inside the black box, to predict possible problems

5. You have learned about a set of general problems, which can be solved by programming:

root finding (GCR),
systems of linear equations,
systems of non-linear equations,
fitting experimental data.

Matrix Formulation of SLE

System of linear equations, example:

$a_{11}x_1 + a_{12}x_2 + \ldots + a_{1n}x_n = b_1$

$a_{21}x_1 + a_{22}x_2 + \ldots. + a_{2n}x_n = b_2$

$\ldots..$

$a_{n1}x_1 + a_{n1}x_2 + \ldots + a_{nn}x_n = b_n$

A - coefficient matrix, b - load vector

$$\begin{bmatrix} a_{11} & a_{12} & \ldots & a_{1n} \\ a_{21} & a_{22} & \ldots & a_{2n} \\ \ldots.. & & & \\ a_{n1} & a_{n2} & \ldots & a_{nn} \end{bmatrix} \begin{bmatrix} x_1 \\ x_2 \\ \ldots \\ x_n \end{bmatrix} = \begin{bmatrix} b_1 \\ b_2 \\ \ldots \\ b_n \end{bmatrix} \qquad A \cdot x = b$$

Gaussian Elimination

Example:

$$x + 2y + z = 0$$

$$2x + 2y + 3z = 3$$

$$-x + 3y + 0z = -4$$

1. Eliminate x from eqs 2 & 3.
2. Eliminate y from equation 2.
3. Solve eq. 3 for z and backsubstitute to eqs 1&2
4. Solve eq. 2 for y and backsubstitute to eq. 1.
5. Solve eq. 1 for x.

Gaussian Elimination (last step)

Subtract m_{32} times (2) from (3)*

$$a_{11}x_1 + a_{12}x_2 + a_{13}x_3 = b_1 \qquad (1)$$

$$a_{22}^{(2)}x_2 + a_{23}^{(2)}x_3 = b_2^{(2)} \qquad (2)^*$$

$$a_{33}^{(3)}x_3 = b_3^{(3)} \qquad (3)^{**}$$

The new coefficients are give by

$$a_{33}^{(3)} = a_{33}^{(2)} - m_{32}a_{23}^{(2)}$$

$$b_3^{(3)} = b_3^{(2)} - m_{32}b_2^{(2)}$$

Scalability

Code for Gaussian elimination contains 3 loops:

1. makes *n-1* runs to eliminate variables

2. *k*-th run goes through *n-k* rows (*k = 1*,. . ., *n-1*)

3. in *i*-th row we calculate $a_{ij} = a_{ij} - m\, a_{kj}$ *n-k* times

Overall about $\sum_{k=1}^{n-1} (n-k)(n-k)$ operations.

Time scales as n^3 ! A rather poor scalability.

LU factorization.

$$A \xrightarrow{GE} U \text{ (upper diagonal)} \qquad A = LU$$

$$\mathbf{b} \xrightarrow{GE} \mathbf{c} \qquad \mathbf{b} = L\,\mathbf{c}$$

$$A = LU$$

L^{-1} describes Gussian elimination.

$$\mathbf{c} = L^{-1}\,\mathbf{b}$$

Numerical Stability

Problems appear if we have small numbers at the diagonal of the coefficients matrix. Solution - ”pivoting”.

Pivoting algorithm:

Searches for the largest a_{ik} in each row below the current one to use for the next elimination step, and rearranges the rows so that m_{ik} is always less than one.

Introduction to Data Analysis

- Random & Systematic Errors
- How to Report and Use Experimental Errors
- Statistical Analysis of Data
  - Statistics of random errors
  - Error propagation, functions of measurables
  - Plotting and displaying the data
  - Fitting the data: linear and non-linear regression

Statistical Analysis of Random Errors.

Mean $$x_{best} = \bar{x} = \frac{1}{N}\sum_{i=1}^{N} x_i$$

Deviation $$d_i = x_i - \bar{x}$$

Standard deviation, root mean square of the measurements $$\sigma_x = \sqrt{\frac{1}{N}\sum_{i=1}^{N}(d_i)^2} = \sqrt{\frac{1}{N}\sum_{i=1}^{N}(x_i - \bar{x})^2}$$

Propagation of Errors

If errors are independent and random:
for the sum/difference the errors of the independent items are *added in quadrature*.

$$q = x + y$$

$$\delta q = \sqrt{(\delta x)^2 + (\delta y)^2} \leq \delta x + \delta y$$

$$q = x + \ldots + z - (u + \ldots + w)$$

$$\delta q = \sqrt{(\delta x)^2 + \ldots + (\delta z)^2 + (\delta u)^2 + \ldots + (\delta w)^2} \leq$$

$$\leq \delta x + \ldots + \delta z + \delta u + \ldots + \delta w$$

Propagation of Errors

Relative error of product/quotient:

$$q = \frac{x \times \ldots \times z}{u \times \ldots \times w}$$

$$\frac{\delta q}{|q|} = \sqrt{\left(\frac{\delta x}{x}\right)^2 + \ldots + \left(\frac{\delta z}{z}\right)^2 + \left(\frac{\delta u}{u}\right)^2 + \ldots + \left(\frac{\delta w}{w}\right)^2} \leq$$

$$\leq \frac{\delta x}{x} + \ldots + \frac{\delta z}{z} + \frac{\delta u}{u} + \ldots + \frac{\delta w}{w}$$

Relative errors of independent measurables are added in quadrature.

Propagation of Errors

General case - function of several variables:

$$q(x,.....,z)$$

$$\delta q = \sqrt{\left(\frac{\partial q}{\partial x}\delta x\right)^2 + ...+ \left(\frac{\partial q}{\partial z}\delta x\right)^2}$$

Error of a power:

$$q = x^n$$

$$\frac{\delta q}{|q|} = |n|\frac{\delta x}{|x|}$$

Gaussian Distribution

For a large number of measurements and random small errors the distribution of experimental data is Gaussian:

$$G_{\bar{x},\sigma} = \frac{1}{\sigma\sqrt{2\pi}} \exp\left\{ -\frac{\left(x - \bar{x}\right)^2}{2\sigma^2} \right\}$$

$$P(x, x + dx) = G(x)dx \qquad \text{(probability density)}$$

$$P(a \le x \le b) = \int_a^b G(x)dx$$

$$\int_{-\infty}^{\infty} G(x)dx = 1 \qquad \text{(normalized)}$$

$$G(\bar{x} + dx) = G(\bar{x} - dx) \qquad \text{(symmetric with respect to } \bar{x})$$

Least Squares Fitting

Multiple measurables $y_1, y_2, \ldots, y_N$, at different values of x: $x_1, x_2, \ldots, x_N$; $y_i = f(x_i)$.

Need to fit measurables $y_{i,}$ to function y=(x).
The simplest case - linear dependence:
$y = \mathbf{a}\, x + \mathbf{b}$.

What are the best **a** and **b** to fit the data?

1. Suggest Gaussian distribution for each y_i, know s_y.
2. Construct $P(y_1, y_2, \ldots, y_N;a,b)$.
3. Maximize $P(y_1, y_2, \ldots, y_N;a,b)$ with respect to **a** & **b**.
4. Find **a** and **b** delivering the max. value of

$P(y_1, y_2, \ldots, y_N;a,b)$.

Least Squares Fitting

Probability of a single measurement:

$$P(y_i) \sim \frac{1}{\sigma_y} \exp\left\{ -\frac{\left(y_i - (ax_i + b)\right)^2}{2\sigma_y^{\,2}} \right\}$$

Probability of a set of measurements:

$$P(y_1, y_2, \ldots, y_N) \sim \frac{1}{\sigma_y^{\,N}} \exp\left\{ -\frac{\chi^2}{2} \right\},$$

$$\chi^2 = \sum_{i=1}^{N} \frac{\left(y_i - (ax_i + b)\right)^2}{\sigma_y^{\,2}}$$

$$\text{We look for: } \frac{\partial \chi^2}{\partial a} = 0 \;\&\; \frac{\partial \chi^2}{\partial b} = 0$$

Least Squares Fitting

We have a system of two linear equations for a&b with the solutions:

$$a = \frac{\sum x_i y_i - \sum x_i \sum y_i}{\Delta}$$

$$b = \frac{\sum x_i^2 y_i - \sum x_i \sum x_i y_i}{\Delta}, \text{ where}$$

$$\Delta = \sum x_i^2 - \left(\sum x_i\right)^2$$

A x + b - least squares fit to the data, or line of regression of y on x.

Linear Least Squares, General case

Our fitting function in general case is:

$$F(x) = a_1 f_1(x) + a_2 f_2(x) + \ldots + a_n f_n(x)$$

$f_1, f_2, \ldots, f_n$ - known functions of x,
$a_1, a_1, \ldots, a_n$ – unknown fitting parameters

Note that the function itself does not have to be linear for the problem to be linear in the fitting parameters.

There is a compact way to formulate the least squares problem: matrix form.

Linear Least Squares, General case

Let us express the problem in matrix notation:

$$Z = \begin{bmatrix} f_1(x_1) & f_2(x_1) & \dots & f_n(x_1) \\ f_1(x_2) & f_2(x_2) & \dots & f_n(x_2) \\ \dots & \dots & \dots & \dots \\ f_1(x_N) & f_2(x_N) & \dots & f_n(x_N) \end{bmatrix}$$

Overall we have now:

$$y = Z \cdot a + e$$

Fitting problem in matrix notation.

$$\text{Look for } \min\left(\sum_{i=1}^{N} e_i^{\,2} \right) = \min(e^T e)$$

$$z^T \cdot z \cdot a = z^T \cdot y \quad \text{or} \quad a = \left(z^T \cdot z\right)^{-1} \cdot z^T \cdot y$$

Nonlinear Regression (Least Squares)

What if the fitting function is not linear in fitting parameters?

We get a nonlinear equation (system of equations).

Example:

$f(x) = a_1(1 - e^{-a_2 x}) + e$

$y_i = f(x_i; a_1, a_1, ..., a_m) + e_i$ or just $y_i = f(x_i) + e_i$

Again look for the minimum of $\sum_{i=1}^{N} e_i^2$ with respect to the fitting parameters.

Nonlinear Regression (Least Squares)

$$y_i = f(x_i, a) + e_i \approx f(x_i, a0) + \sum_{j=1}^{n} (a_j - a0_j) \frac{\partial f(x_i, a0)}{\partial a_j} + e_i$$

$$y_i - f(x_i, a0) = \sum_{j=1}^{n} (a_j - a0_j) \frac{\partial f(x_i, a0)}{\partial a_j} + e_i, \qquad for i = 1,2,\ldots, N$$

or in matrix form:

$$D = z \cdot \Delta a + e, \quad where$$

$$D = \begin{pmatrix} y_1 - f(x_1, a0) \\ \ldots \\ y_N - f(x_N, a0) \end{pmatrix} \qquad z = \begin{pmatrix} \frac{\partial f(x_1, a0)}{\partial a_1} & \ldots & \frac{\partial f(x_1, a0)}{\partial a_n} \\ \ldots & \ldots & \ldots \\ \frac{\partial f(x_N, a0)}{\partial a_n} & \ldots & \frac{\partial f(x_N, a0)}{\partial a_n} \end{pmatrix}$$

Nonlinear Regression (Least Squares)

Linear regression: $y = z \cdot a + e \rightarrow z^T \cdot z \cdot a = z^T \cdot y$

Now, nonlinear regression: $D = z \cdot \Delta a + e \rightarrow z^T \cdot z \cdot \Delta a = z^T \cdot D$

Old good linear equations with Δa in place of a, D in place of y and z with partial derivatives in place of z with values of fitting functions.

Besides, in case of linear regression it was enough to solve the SLE once, while now solving the above system we just get the next approximation to the best fit.

Systems of Nonlinear Equations

A system of non-linear equations:

$f_1(x_1,x_2,x,\ldots,x_N)=0$

$f_2(x_1,x_2,x,\ldots,x_N)=0$ => $f(x)=0$

….. Start from initial guess $x^{[0]}$

$f_N(x_1,x_2,x,\ldots,x_N)=0$

As before expand each equation at the solution $\hat{x}$ with $f(\hat{x})=0$:

$$f_i(x) = f_i(\hat{x}) + \sum_{j=1}^{N}(x_j - \hat{x}_j)\frac{\partial f_i(\hat{x})}{\partial x_j} + \frac{1}{2}\sum_{j=1}^{N}\sum_{k=1}^{N}(x_j - \hat{x}_j)\frac{\partial^2 f_i(\hat{x})}{\partial x_j \partial x_k}(x_k - \hat{x}_k) + \ldots$$

Assume $x^{[0]}$ is close to $\hat{x}$ and discard quadratic terms:

$$f_i(x) \approx \sum_{j=1}^{N}(x_j - \hat{x}_j)\frac{\partial f_i(\hat{x})}{\partial x_j}$$

- Linearize the system of equations.
- Pick an initial guess x0.
- Solve linearized system for Δx – correction to the initial guess.
- Use $x0 + \Delta x$ as initial guess, and solve the linear system again.
- Repeat this procedure until the convergence criterion is satisfied.

Matrix formulation of linearized equations:

$J(x^{[i]}) \Delta x^{[i]} = -f(x^{[i]})$, where J – Jacobian matrix,

$J_{ij} = \partial f_i / \partial x_j$, $\Delta x^{[i]} = x^{[i+1]} - x^{[i]}$.