Quick Review of some Prerequisites

• Three areas:
  − solid calculus, particularly calculus of several variables.
  − linear algebra
  − Programming (Coding)

• The term project requires that you use a programming language that can handle Unix standard input and output. Languages like C, C++, java, python, Fortran, ... all work well, Matlab does not. Matlab may be useful for some homework problems. More info on Friday.

• The shortest programs I have seen in the past were in python. However, my favorite language is java.
Errors

- Suppose we approximate a quantity $X$ by an approximation $\hat{X}$. Then the associated error is simply

$$E = |X - \hat{X}|.$$ 

\[\text{The word “error” in this context does not mean some kind of mistake. It’s simply the difference between a true value and its approximation.}\]

- The relative error is

$$\tilde{E} = \left| \frac{E}{A} \right|.$$ 

\[\text{Sometimes the sign of the error matters. In that case, the absolute values are omitted.}\]

- Example: We know from Calculus that

$$e^x = 1 + x + \frac{x^2}{2} + R_2$$

where

$$R_2 = \frac{e^\xi}{3!} x^3$$

for some $\xi$ between zero and $x$.

- Supposing that $0 < x < 1$ we obtain, for example, that

$$E = |R_2| \leq \frac{e}{3!} x^3$$
Taylor Series

\[ f(x + h) = \sum_{k=0}^{n} \frac{f^{(k)}(x)}{k!} h^k + R_{n+1} \]

where

\[ R_{n+1} = \frac{f^{(n+1)}(\xi)}{(n + 1)!} h^{n+1} \]

- You want to be able to compute Taylor Series and understand the concept of the radius of convergence of a Taylor Series.

- Rule of Thumb: In Numerical Analysis, if you don’t know what else to do you expand into a Taylor Series.
The Big O notation

- We say that Assuming that \( f(0) = g(0) = 0 \) we say
  \[
f(h) = \mathcal{O}(g(h))
  \]
as \( h \) goes to zero if there exists a constant \( C \) such that
  \[
  \lim_{h \to 0} \frac{f(h)}{g(h)} = C \neq 0. \tag{1}
  \]

- With that notation, assuming that \( f \) is sufficiently often differentiable and \( f^{(n+1)}(x) \neq 0 \), Taylor’s formula implies that
  \[
f(x + h) = \sum_{k=0}^{n} \frac{f^{(k)}(x)}{k!} h^k + \mathcal{O}(h^{n+1}).
  \]

- Similarly, assuming that
  \[
  \lim_{x \to \infty} f(n) = \lim_{n \to \infty} g(n) = \infty
  \]
we say that
  \[
f(n) = \mathcal{O}(g(n))
  \]
as \( n \) goes to infinity if there exists a constant \( C \) such that
  \[
  \lim_{n \to \infty} \frac{f(n)}{g(n)} = C \neq 0. \tag{2}
  \]
• For example, the standard algorithm of solving an $n \times n$ linear system by Gaussian elimination requires $O(n^3)$ operations.

• Basically (1) means that two methods with discretization parameter $h$ have essentially the same accuracy, and (2) means that two methods requires the same effort to solve a problem of size $n$.

• Constant factors can be ignored in this context.

• There is a corresponding “little O” notation where the constant $C$ is replaced with zero. We will not use that notation.
Application of big O: Optimization

- Consider the problem of minimizing a function. We know from Calculus that
  \[ f(x_0) = \min \]
  if \( f'(x_0) = 0 \) and \( f''(x_0) > 0 \).
- Let's revisit this problem using a Taylor Series and then apply the same idea to a function of several variables. We have
  \[ f(x_0+h) = f(x_0) + hf'(x_0) + \frac{1}{2}h^2 f''(x_0) + O(h^3). \]
- For small values of \( h \) the term \( hf'(x_0) \) will dominate the higher order terms if \( f'(x_0) \neq 0 \).
- The word dominate means that the term \( hf'(x_0) \) will be arbitrarily much larger than the higher order terms if \( h \) is sufficiently small.
- We can make that term positive or negative if \( f'(x_0) \neq 0 \) by picking \( h \) of the appropriate sign.
- So clearly, \( f'(x_0) \) must be be zero for \( f(x_0) \) to be a local minimum.
- If in addition \( f''(x_0) > 0 \) then the term \( \frac{1}{2}h^2 f''(x_0) \) will be positive for non-zero \( h \) and it can only increase the value of \( f \). For small non-zero values of \( h \) that term dominates higher order terms and we must have a minimum.
- The same idea works for scalar valued functions of several variables!
• \( \mathbb{R} \) is the set of real numbers.

• \( \mathbb{R}^n \) is the set of vectors with \( n \) real entries (usually considered a column vector).

• \( f \) is a scalar valued function of \( n \) variables. We write this as

\[
f : \mathbb{R}^n \longrightarrow \mathbb{R}
\]

• Suppose \( h \) and \( x^{(0)} \) are vectors in \( \mathbb{R}^n \).

• Then

\[
f \left( x^{(0)} + h \right) = f \left( x^{(0)} \right) + \left( \nabla f \left( x^{(0)} \right) \right)^T h + \frac{1}{2} h^T \nabla^2 f \left( x^{(0)} \right) h + \mathcal{O}(\|h\|^3)
\]

where

\[
\nabla f \left( x^{(0)} \right) = \left[ \frac{\partial f}{\partial x_i} \left( x^{(0)} \right) \right]_{i=1,...,n}
\]

is the gradient of \( f \) and

\[
\nabla^2 f \left( x^{(0)} \right) = H = \left[ \frac{\partial^2 f \left( x^{(0)} \right)}{\partial x_i \partial x_j} \right]_{i,j=1,...,n}
\]

is the Hessian (matrix) of \( f \).

• Mixed partial derivatives commute, so \( H \) is symmetric, i.e., \( H = H^T \).

• In order for \( f \left( x^{(0)} \right) \) to be a local minimum, again we must have

\[
\nabla f \left( x^{(0)} \right) = 0,
\]
i.e., the gradient must be zero. Otherwise we could make the term \((\nabla f (x^{(0)}))^T h\) negative by picking \(h\) a multiple of the negative gradient. If \(\|h\|\) is sufficiently small than that term would dominate the higher order terms and we would locally decrease the function value.

- If, in addition,

\[ h^T H h > 0 \]

for all non-zero vectors \(h\) we can be sure that \(f (x^{(0)})\) is a local minimum. This is because the term \(h^T H h\) can only make the function value larger, and for small values of \(\|h\|\), that term will dominate the higher order terms.

- A symmetric real matrix \(A\) is **positive definite** if

\[ x^T Ax > 0 \]

for all non-zero vectors \(x\).

- So we have a minimum if the gradient is zero and the Hessian is positive definite.

- **Positive Definiteness is the right generalization of positivity of numbers.**

- Your first thought when minimizing a function is: there must be a positive definite matrix somewhere. (Ronald Reagan\(^{-1-}\))

\(^{-1-}\) Actually, he said: **My first thought when I enter a room full of manure is: there must be a pony in here, somewhere.**
You want to be familiar with these terms of linear algebra:
- vectors
- vector spaces, linear spaces,
- linear combination of vectors
- linear independence of sets of vectors
- span of a set of vectors
- spanning set of a vector space
- basis of a vector space
- dimension of a vector space.
More Linear Algebra

- Let $A$ be an $m \times n$ real matrix.

\[ A \in \mathbb{R}^{m \times n} \]

- understand the concepts of rows and columns of $A$, and the $(i,j)$-entry of $A$.

- Matrices can be added and multiplied with scalars.

- They can also be multiplied with vectors $x \in \mathbb{R}^n$:

\[
Ax = \left[ \sum_{j=1:n} a_{ij}x_j \right]_{i=1,\ldots,m}
\]

- $Ax$ is a linear combination of the columns of $A$, with the coefficients being the entries of $x$.

- Understand the concepts of the
  - column space
  - row space
  - null space
  - rank
  of $A$. 
Matrix Multiplication

- let

\[ AB = C \]

where

\[ A \in \mathbb{R}^{m \times p}, \quad B \in \mathbb{R}^{p \times n}, \quad \text{and} \quad C \in \mathbb{R}^{m \times n} \]

- We know that

\[ c_{ij} = \sum_{k=1}^{p} a_{ik} b_{kj} \]

- Examples:
• There is a better way.
Interpretation of Matrix Product

• $c_{ij}$ is the dot product of the $i$-th row of $A$ and the $j$-th column of $B$:
  \[ c_{ij} = \text{row}_i(A) \cdot \text{col}_j(B). \]

• The $j$-th column of $C$ equals $A$ multiplied with the $j$-th column of $B$:
  \[ \text{col}_j(C) = A\text{col}_j(B). \]

• The $i$-th row of $C$ equals the $i$-th row of $A$ multiplied with $B$:
  \[ \text{row}_i(C) = \text{row}_i(A)B. \]

• $C$ is a sum of rank 1 matrices:
  \[ A = \sum_{k=1}^{p} \text{col}_k(A)\text{row}_k(B). \]

• Note that row vector times column vector gives a matrix (of rank 1).
Why do we multiply matrices this way?

- The quick answer: Matrices are linear functions, and matrix multiplication means function composition.
- Everything flows from there!
- A function

\[ T : \mathbb{R}^n \rightarrow \mathbb{R}^m \]

is linear if

\[ T(u + v) = T(u) + T(v) \]

and

\[ T(ku) = kT(u) \]

for all \( u \) and \( v \) in \( \mathbb{R}^n \) and real numbers \( k \).

- Given an \( m \times n \) matrix \( A \) we can define \( T \) by

\[ T(x) = Ax. \]

- The function

\[ T : \mathbb{R}^m \rightarrow \mathbb{R}^n \]

is linear!
- Remarkably, we can also go the other way.
- Suppose \( T \) is a linear function.
• Let $e_i$ be the $i$-th standard basis vector in $R^n$, i.e.,

$$e_i = \begin{bmatrix} 0 \\ \vdots \\ 0 \\ 1 \\ 0 \\ \vdots \\ 0 \end{bmatrix}.$$

• Then

$$T(x) = T\left( \sum_{i=1}^{m} x_i e_i \right) = \sum_{i=1}^{m} x_i T(e_i) = Ax$$

where

$$A = [T(e_1) \ T(e_2) \ \ldots \ T(e_m)]$$
• Example: Rotation in the Plane
• Linear Functions can be composed!

• The composition \( f \circ g \) of two linear functions \( f \) and \( g \) is linear, and its matrix is the product of the matrices of the constituent functions.

\[
f \circ g
\]

\[
\begin{array}{ccc}
\mathbb{R}^n & \xrightarrow{g} & \mathbb{R}^p & \xrightarrow{f} & \mathbb{R}^m \\
\xrightarrow{B} & & \xrightarrow{A} & & \\
p \times n & & m \times p & & \\
C = AB \\
\end{array}
\]

• Exercise: Work out the details!

• Note the switch in the sequence. \( B \) comes first in the diagram and second in the product, just like \( g \) comes first in the diagram and second in the composition.
Eigenvalues and Eigenvectors

- Suppose $A \in \mathbb{R}^{n \times n}$, $x$ is a (possibly complex) vector, $\lambda$ is a (possibly complex) number, and

$$Ax = \lambda x.$$ 

Then $x$ is an eigenvector of $A$, and $\lambda$ is the corresponding eigenvalue.

- “eigen” is the German word for “own”.

- The eigenvalues are the roots of the characteristic polynomial of $A$:

$$p(\lambda) = \det(A - \lambda I) = 0 \quad (3)$$

where $I$ is the $n \times n$ identity matrix. $p$ is a polynomial of degree $n$ with leading term $(-\lambda)^n$ (exercise). The equation (3) is the characteristic equation of the matrix $A$. 

• Computing the eigenvalues by solving the characteristic equation only works for very small matrices (since the roots of a polynomial are very sensitive to small perturbations of the coefficients of the polynomial).

• However, the reverse process, computing the roots of a polynomial by solving a related eigenvalue problem, works very well.

• Suppose that the polynomial \( p \) is given by

\[
p(\lambda) = (-1)^n \left( \lambda^n - \alpha_{n-1}\lambda^{n-1} - \alpha_{n-2}\lambda^{n-2} - \ldots - \alpha_0 \right)
\]

• The companion matrix \( C \) of \( p \) is defined by

\[
C = \begin{bmatrix}
\alpha_{n-1} & \alpha_{n-2} & \cdots & \alpha_1 & \alpha_0 \\
1 & 0 & \cdots & 0 & 0 \\
0 & 1 & \cdots & 0 & 0 \\
0 & 0 & \cdots & 0 & 0 \\
\vdots & \vdots & \ddots & \vdots & \vdots \\
0 & 0 & \cdots & 1 & 0
\end{bmatrix}
\]

• Exercise: Show that

\[
p(\lambda) = \det(C - \lambda I).
\]
The Gershgorin Theorem

• Among all mathematical theorems the Gershgorin Theorem maximizes the ratio of utility and notoriety.

• There is a book on this Theorem: R.S. Varga, Gershgorin and His Circles, Springer, 2010, 978-3-642-05928-5.

• Suppose $\lambda$ is an eigenvalue of $A$. Then, for some $i$ in \{1, 2, \ldots, n\}

$$|\lambda - a_{ii}| \leq \sum_{\substack{j=1 \\ j \neq i}}^{n} |a_{ij}|.$$ 

• In other words all eigenvalues of $A$ lie in circles around the diagonal entries, with the radii being the sum of the absolute values of the off diagonal entries in the corresponding row.
• This is easy to see:
Floating Point Arithmetic

- The wikipedia article for “floating point arithmetic” is a good source of information on this topic.

Computers are finite

- This implies that only finitely many numbers can be expressed on a computer. Most numbers can’t.

- A floating point number $z$ is of the form

$$z = \pm 0.m_1m_2\ldots m_k \times b^{n - L}$$

where

$$1 \leq n \leq M \quad \text{and} \quad 0 \leq m_i < b, \quad i = 1, \ldots, k$$

for some values of $b$, $k$, $N$, and $M$.

- $b$ is the base, $n$ is the exponent, and $0.m_1m_2\ldots m_k$ is the mantissa or significand. The integer $L$ is the exponent bias.

- Typically the base is 2. The exponent in that case is also expressed as a binary number. The digits and the signs of the mantissa and the exponent each correspond to one bit.

- Most computers use the IEEE 754 standard where a single double precision floating point number (the most common type) occupies 64 bits and the significand has 53 bits. (There are also single and quadruple precision.)
Hewlett Packard calculators use base 10.

In binary systems the leading bit of the mantissa is usually 1, and not stored. It is referred to as the **hidden bit**.

- **Example:** (modified IEEE 754). Consider 64 bit floating point numbers of the form:

\[
z = \pm 0.1b_2b_3\ldots b_{52} \times 2^{e_1e_2\ldots e_{10} - L}
\]

where the \(b_i\) and \(e_i\) are binary digits and \(L = 1023\).

If all the \(e_i\) are zero then \(z\) is considered zero, if all the \(e_i\) are one \(z\) is considered to be NaN, not a number.

Note that all floating point numbers are in fact **rational numbers**.

Floating point numbers are bunched at the origin.

- For illustration consider the much smaller set of floating point numbers of the form

\[
z = \pm 0.1b_1b_2 \times 2^{e_1e_2e_3 - L}
\]

where \(L = 3\) and, as before, the \(b_i\) and \(e_i\) are binary digits.
There are six bits and hence 64 possible numbers. They are all rational with a common denominator of 64. Half of these are negatives of the others. The following Table lists the positive numbers in increasing sequence.
<table>
<thead>
<tr>
<th>count</th>
<th>$b_1$</th>
<th>$b_2$</th>
<th>$e_1$</th>
<th>$e_2$</th>
<th>$e_3$</th>
<th>64z</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>4</td>
</tr>
<tr>
<td>2</td>
<td>0</td>
<td>1</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>5</td>
</tr>
<tr>
<td>3</td>
<td>1</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>6</td>
</tr>
<tr>
<td>4</td>
<td>1</td>
<td>1</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>7</td>
</tr>
<tr>
<td>5</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>1</td>
<td>8</td>
</tr>
<tr>
<td>6</td>
<td>0</td>
<td>1</td>
<td>0</td>
<td>0</td>
<td>1</td>
<td>10</td>
</tr>
<tr>
<td>7</td>
<td>1</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>1</td>
<td>12</td>
</tr>
<tr>
<td>8</td>
<td>1</td>
<td>1</td>
<td>0</td>
<td>0</td>
<td>1</td>
<td>14</td>
</tr>
<tr>
<td>9</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>1</td>
<td>0</td>
<td>16</td>
</tr>
<tr>
<td>10</td>
<td>0</td>
<td>1</td>
<td>0</td>
<td>1</td>
<td>0</td>
<td>20</td>
</tr>
<tr>
<td>11</td>
<td>1</td>
<td>0</td>
<td>0</td>
<td>1</td>
<td>0</td>
<td>24</td>
</tr>
<tr>
<td>12</td>
<td>1</td>
<td>1</td>
<td>0</td>
<td>1</td>
<td>0</td>
<td>28</td>
</tr>
<tr>
<td>13</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>1</td>
<td>1</td>
<td>32</td>
</tr>
<tr>
<td>14</td>
<td>0</td>
<td>1</td>
<td>0</td>
<td>1</td>
<td>1</td>
<td>40</td>
</tr>
<tr>
<td>15</td>
<td>1</td>
<td>0</td>
<td>0</td>
<td>1</td>
<td>1</td>
<td>48</td>
</tr>
<tr>
<td>16</td>
<td>1</td>
<td>1</td>
<td>0</td>
<td>1</td>
<td>1</td>
<td>56</td>
</tr>
<tr>
<td>17</td>
<td>0</td>
<td>0</td>
<td>1</td>
<td>0</td>
<td>0</td>
<td>64</td>
</tr>
<tr>
<td>18</td>
<td>0</td>
<td>1</td>
<td>1</td>
<td>0</td>
<td>0</td>
<td>80</td>
</tr>
<tr>
<td>19</td>
<td>1</td>
<td>0</td>
<td>1</td>
<td>0</td>
<td>0</td>
<td>96</td>
</tr>
<tr>
<td>20</td>
<td>1</td>
<td>1</td>
<td>1</td>
<td>0</td>
<td>0</td>
<td>112</td>
</tr>
<tr>
<td>21</td>
<td>0</td>
<td>0</td>
<td>1</td>
<td>0</td>
<td>1</td>
<td>128</td>
</tr>
<tr>
<td>22</td>
<td>0</td>
<td>1</td>
<td>1</td>
<td>0</td>
<td>1</td>
<td>160</td>
</tr>
<tr>
<td>23</td>
<td>1</td>
<td>0</td>
<td>1</td>
<td>0</td>
<td>1</td>
<td>192</td>
</tr>
<tr>
<td>24</td>
<td>1</td>
<td>1</td>
<td>1</td>
<td>0</td>
<td>1</td>
<td>224</td>
</tr>
<tr>
<td>25</td>
<td>0</td>
<td>0</td>
<td>1</td>
<td>1</td>
<td>0</td>
<td>256</td>
</tr>
<tr>
<td>26</td>
<td>0</td>
<td>1</td>
<td>1</td>
<td>1</td>
<td>0</td>
<td>320</td>
</tr>
<tr>
<td>27</td>
<td>1</td>
<td>0</td>
<td>1</td>
<td>1</td>
<td>0</td>
<td>384</td>
</tr>
<tr>
<td>28</td>
<td>1</td>
<td>1</td>
<td>1</td>
<td>1</td>
<td>0</td>
<td>448</td>
</tr>
<tr>
<td>29</td>
<td>0</td>
<td>0</td>
<td>1</td>
<td>1</td>
<td>1</td>
<td>512</td>
</tr>
<tr>
<td>30</td>
<td>0</td>
<td>1</td>
<td>1</td>
<td>1</td>
<td>1</td>
<td>640</td>
</tr>
<tr>
<td>31</td>
<td>1</td>
<td>0</td>
<td>1</td>
<td>1</td>
<td>1</td>
<td>768</td>
</tr>
<tr>
<td>32</td>
<td>1</td>
<td>1</td>
<td>1</td>
<td>1</td>
<td>1</td>
<td>896</td>
</tr>
</tbody>
</table>
Notes

• The (positive) numbers that can be represented in this system range from

\[
\begin{align*}
z_{\text{min}} &= \frac{4}{64} = \frac{1}{16} \\
\text{to} \\
z_{\text{max}} &= \frac{896}{64} = 56.
\end{align*}
\]

• The spacing among neighboring numbers starts at 1/64, and then keeps doubling until it reaches 2.
What is $\sin \left( 10^{20} \pi \right)$?
My first program

- the first program I ever wrote was meant to compute a certain result for 10 values of the probability

\[ x = 0.1, \; 0.2, \; \ldots, \; 1.0. \]

- The very simple FORTRAN program

```fortran
1  Do 1 i = 1,10
2   x = i/10.0
3  1   write(*,2) x, exp(x)
4  2   continue
5  end
```

does just that. It computes the exponential of \( x \) for those values and produces the output

<table>
<thead>
<tr>
<th>x</th>
<th>( \exp(x) )</th>
</tr>
</thead>
<tbody>
<tr>
<td>0.1</td>
<td>1.105</td>
</tr>
<tr>
<td>0.2</td>
<td>1.221</td>
</tr>
<tr>
<td>0.3</td>
<td>1.350</td>
</tr>
<tr>
<td>0.4</td>
<td>1.492</td>
</tr>
<tr>
<td>0.5</td>
<td>1.649</td>
</tr>
<tr>
<td>0.6</td>
<td>1.822</td>
</tr>
<tr>
<td>0.7</td>
<td>2.014</td>
</tr>
<tr>
<td>0.8</td>
<td>2.226</td>
</tr>
<tr>
<td>0.9</td>
<td>2.460</td>
</tr>
<tr>
<td>1.0</td>
<td>2.718</td>
</tr>
</tbody>
</table>

- That’s not how I wrote the program, however. I wrote something like
1  x = 0.0
2  1  continue
3  x = x + 0.1
4  write(*,3) x, exp(x)
5  if (x .eq. 1.0) go to 2
6  go to 1
7  2  continue
8  3  format(f5.2,3x,f7.4)
9  end

The output from that second program traumatized me for life! No kidding!

- What does the second program do?