Errors and Complexity

1. What is the formula for relative error?

$$e_r = \frac{|x - \bar{x}|}{|x|}$$

x : exact
$\bar{x}$: approximate

2. What is the formula for absolute error?

$$e_a = |x - \bar{x}|$$

3. Which is usually the better way to calculate error?

relative

4. If you have k accurate significant digits, what is the tightest bound on your relative error?

$$e_r \leq 10^{-k+1}$$

5. Given a maximum relative error, what is the largest (or smallest) value you could have?

$\bar{x}$: approx
x : exact

$\bar{x} = x(1 \pm e)$

$\bar{x}_{min} = x(1 - e_r)$
$\bar{x}_{max} = x(1 + e_r)$

$e_r = \frac{|x - \bar{x}|}{|x|} \Rightarrow |x| e_r = |x - \bar{x}|$

$\bar{x} = x - |x| e_r$
$\bar{x} = x + |x| e_r$

$\begin{cases} |x| e_r = x - \bar{x} \\ -|x| e_r = x - \bar{x} \end{cases}$

6. How do you compute relative and absolute error for vectors?

Use the 2-norm

$$e_a = \|\bar{x} - x\|$$
$$e_r = \frac{\|\bar{x} - x\|}{\|x\|}$$

7. What is truncation error?

occurs when discrete values are used to approximate an expression
i.e. $\sin(\theta) \approx \theta$ for small θ

8. What is rounding error?

occurs when digits in a decimal are lost

9. What does it mean for an algorithm to take O(n^3) time?

The algorithm performs n^3 operations on n elements.

time $= \alpha n^3$

10. Can you give an example of an operation that takes O(n^2) time?

Bubble Sort

- matrix-vect mult.
- backward + forward solve

11. What does it mean for error to follow O(h^4)?

$e = \alpha h^4$

12. If you know an operation is O(n^4), can you predict its time/error on unseen inputs from inputs that you already have?

Yes. Ex: $t = O(n^4)$ Given: n_1 has t_1 runtime

$n_2 = 2n_1$

$t_1 = \alpha n_1^4$
$t_2 = \alpha n_2^4$
$\Rightarrow \frac{t_1}{t_2} = \left(\frac{n_1}{n_2}\right)^4 \Rightarrow \frac{t_1}{t_2} = \left(\frac{1}{2}\right)^4 \Rightarrow t_2 = 2^4 t_1$

13. If you are given runtime or error data for an operation, can you find the best x such that the operation is O(n^x)?

Yes. Use plotting and find the slope of the straight line ✓

OR

$t_1 = \alpha n_1^x$
$t_2 = \alpha n_2^x$

$\frac{t_1}{t_2} = \left(\frac{n_1}{n_2}\right)^x \quad x = \frac{\log(t_1/t_2)}{\log(n_1/n_2)}$

Floating point numbers

1. What are the differences between floating point and fixed point representation?

Fixed point numbers present a trade off between range (more integral bits) and precision (more fractional bits), whereas floating point numbers can have a variable number of either.

2. Given a real number, what is the rounding error involved in storing it as a machine number? What is the relative error?

rounding error

absolute error: $|x - fl(x)| \leq \epsilon_m \cdot 2^n$

relative error: $\epsilon_m \geq \frac{|x - fl(x)|}{|x|}$

fractional part of significand

exponent

$x = \pm 1.f \times 2^m$

sign

significand

3. Explain the different parts of a floating-point number: sign, significand, and exponent.

sign: first bit, 1 if number is negative, else 0

significand: significant bits of number, excluding leading 1 (fractional part) ✓

exponent: power of 2 used for calculating the original number

4. How is the exponent of a machine number actually stored?

stored as unsigned binary number with shift used to obtain original exponent ✓

shift single precison = 127
double = 1023

5. What is machine epsilon?

distance between 1 and next possible floating point number above 1 ✓

$\epsilon_m = 2^{-n}$ $\epsilon_m = 2^{-23}$ (single) $\epsilon_m = 2^{-52}$ (double)

6. What is underflow (UFL)? $= 2^L$

smallest possible floating point number (in magnitude) that can be stored (fractional part is 0, exponent is at minimum) ✓

7. What is overflow?

largest possible floating point number that can be stored (fractional part all 1, exponent is at maximum) ✓

$UFL = 2^{U+1}(1 - 2^{-p})$

1.00
1.01
0.25
$2^{-2} = 0.25$

8. Given a toy floating-point system, determine machine epsilon and UFL for that system.

given n fractional bits machine epsilon is defined as:

$\epsilon_m = 2^{-n}$; $UFL = 2^L$ where L is the lower bound on the exponent ($m \in [L, U]$)

9. How can we bound the relative error of representing a real number as a normalized machine number?

relative error is bound by machine epsilon.

$$e_r \leq \epsilon_m$$

10. How do you store zero as a machine number?

both the exponent and fraction are 0, note that this allows for ± 0

remember the definition for the gap between 2 numbers!

11. What is cancellation? Why is it a problem?

cancellation occurs ~~when~~ when we subtract two numbers that are very close to eachother (similar magnitude). It can result in loss of significance. Cancellation can also occur if we add a very ~~large num~~ small number to a very large number resulting in the smaller number being disregarded, this results in loss of precision

12. What types of operations lead to catastrophic cancellation?

subtraction
addition

13. Given two different equations for evaluating the same value, can you identify which is more accurate for certain x and why?

~~It depends on the input value~~

(1) $x = a + (b - c)$

(2) $x = (a + b) - c$

if b and c are close in magnitude then (2) is better than (1)

Taylor series

1. What is the general form of a Taylor series?

$$T_n(x) = \sum_{i=1}^{\infty} \frac{f^{(i)}(a)}{i!}(x-a)^i$$

2. How do you use a Taylor series to approximate a function at a given point?

Use the function f at the point a to the accuracy of n

3. How can you approximate the derivative of a function using Taylor series?

Compute the taylor series for a function and then take the derivative of each term

4. How can you approximate the integral of a function using Taylor series?

Compute the taylor series for a function and then integrate each term

5. Given a function and a center, can you write out the n-th degree Taylor polynomial?

Yes, see question 1.

6. For an n-th degree Taylor polynomial, what is the bound on the error of your approximation as a function of distance from the center?

$$\text{error} = f(x) - T_n(x) \leq C \cdot h^{n+1} = O(h^{n+1})$$

for a function f centered at x_0 with $h = x - x_0$.

7. Can you determine how many terms are required for a Taylor series approximation to have less than some given error?

Taylor Remainder Theorem:

$$R_n(x) = f(x) - T_n(x) = \frac{f^{(n+1)}(a)}{(n+1)!}(x-a)^n$$

stop when next term is less than given error error $= O(h^{n+1}) = Mh^{n+1}$

Vector and matrix norms

1. What is a vector norm? (What properties must hold for a function to be a vector norm?)
   - a generalization of abs val' to vects.
   - $f(x): \mathbb{R}^n \to \mathbb{R}_+$ rets. a magnitude of the input vect
   - often notated as $\|\bar{x}\|$

   Vector Norm Properties
   - $\|\bar{v}\| > 0 \iff v \neq 0$
   - $\|\alpha\bar{v}\| = |\alpha|\|\bar{v}\|$ for all scalars α
   - obey triangle ineq: $\|\bar{v}+\bar{u}\| \leq \|\bar{v}\| + \|\bar{u}\|$
2. What is the definition of an induced matrix norm? What do they measure?

Induced (or operator) matrix norms are associated with a specific vector norm $\|\cdot\|$ and are defined as $\|A\| := \max_{\|x\|=1} \|Ax\|$

It tells you the maximum amplification of the norm of any vector when multiplied by the matrix.

3. What properties do induced matrix norms satisfy?

① $\|Ax\| \leq \|A\|\|x\|$

② $\|AB\| \leq \|A\|\|B\|$

4. For an induced matrix norm, given |x|and |Ax| for a few vectors, can you determine a lower bound on |A|?

$\|A\bar{x}\| \leq \|A\|\|\bar{x}\|$

$\rightarrow \|A\| \geq \max_{\|x\|\neq 0} \frac{\|Ax\|}{\|x\|}$

5. For a given vector, compute the 1, 2 and ∞ norm of the vector.

$\|w\|_1 = \sum_{i=1}^{n} |w_i|$ $\quad \|w\|_2 = \sqrt{\sum_{i=1}^{n} |w_i|^2}$

$\|w\|_\infty = \max_{w_i \in w} |w_i|$

6. For a given matrix, compute the 1, 2 and ∞ norm of the matrix.

$\|A\|_1 = \max_j \sum_{i=1}^{n} |A_{ij}|$ // max abs col sum

$\|A\|_\infty = \max_i \sum_{j=1}^{n} |A_{ij}|$ // max abs row sum

$\|A\|_2 = \max_k \sigma_k$ // max singular val.

Sparse matrices

1. What does it mean for a matrix to be sparse?

An $n \times n$ matrix is sparse if it has $O(n)$ nonzero values.

2. Given a sparse matrix, put the matrix in CSR format.

```
For each row i
  Append Length(col) to RowPTR
  For each column j,
    If A[i, j] != 0,
      Append A[i, j] to Data
      Append j to Col
```

For CSR format, you need 3 vectors: data, col, and rowptr. For data, you fill it with the nonzero values of each row. For col, you append the corresponding column index of that data point. For rowptr, you start at zero, and append the number of data points seen in each row + the rows before it. So if you have 2 data points in row 0 and 3 in row 1, the corresponding rowptr vector is [0, 2, 5].

3. Given a sparse matrix, put the matrix in COO format.

For COO format, you need 3 vectors: data, row, and col. Data is filled with the nonzero values from each row. Row is filled w/ the row index of the corresponding data point. Col is filled w/ the column index of the corresponding data point.

4. For a given matrix, how many bytes total are required to store it in CSR format?

- to store data → sizeof(float) * # of data vals
- to store col → sizeof(int) * # of data vals
- to store rowptr → sizeof(int) * (# of rows + 1)

For total bytes, add everything together

5. For a given matrix, how many bytes total are required to store it in COO format?

The size of the type of the values times the number of nonzero values.

+

The size of the index type times the number of nonzero values times two.

e.g. sizeof(float) · 5 + sizeof(int) · 5 · 2 for 5 nonzero values

Linear System of Equations

1. Given a factorization PA=LU, how would you solve the system Ax=b?

first, solve for y in $Ly = Pb$ (forward substitution)
then, solve for x in $Ux = y$ (back substitution)

2. Recognize and understand Python code implementing forward substitution, back substitution, and LU factorization. Write pseudo-codes here.

```
def LU_solve(L,U,b):
    y = forward_sub(L,b)
    x = back_sub(U,y)
    return x

def back_sub(U,y):
    for i in range(Ushape,-1,-1):
        temp = y[i]
        for j in range(i+1, Ushape):
            temp -= U[i,j] * x[j]
        x[i] = temp/U[i,i]
    return x
```

```
def forward_sub(L,b):
    for i in range(shape of L[0]):
        temp = b[i]
        for j in range(i):
            temp -= L[i,j] * x[j]
        x[i] = temp / L[i,i]
    return x
```

algorithm fails

3. What happens if we try to solve a system Ax=b with a singular matrix A?

if it's a singular matrix, there is no unique solution

4. Why do we use pivoting when solving linear systems?

We use pivoting to rearrange the columns/rows of A to make sure the top-left element has the larger element (prevents division by zero)

5. How do we choose a pivot element?

- look at the first column of matrix A and choose the number with the largest absolute value - that is your pivot element (this is 1st partial pivot)
- in general - choose entry with largest absolute value from the column of the matrix that is currently being considered as the pivot element

6. What is the cost of matrix-matrix multiplication?

The cost of matrix multiplication of an nxn matrix multiplied by a nxn matrix is $O(n^3)$ an nxp multiplied by a pxm matrix is $O(nmp)$

7. What is the cost of computing an LU or LUP factorization?

The cost is $O(n^3)$ compute LU or LUP factorization

8. What is the cost of forward or back substitution?

The cost of forward substitution is $O(n^2)$

9. What is the cost of solving Ax=b for a general matrix?

$O(n^3)$

10. What is the cost of solving Ax=b for a triangular matrix?

$O(n^2)$

11. What is the cost of solving Ax=bi with the same matrix A and several right-hand side vectors bi?

$n^3 + n^2 * (\#\text{ right-hand side})$

$= O(n^3)$

12. Given a process that takes time O(n^k), what happens to the runtime if we double the input size (i.e. double n)? What if we triple the input size?

$t_1 = O(n^k)$

$t_1 = \alpha n_1^k \rightarrow \frac{t_1}{t_2} = \left(\frac{n_1}{2n_1}\right)^k \rightarrow \boxed{t_2 = 2^k t_1}$ double

$t_2 = \alpha (2n_1)^k$

triple 3^k

...

Conditioning

1. What is the definition of a condition number?

$\text{cond}(A) = \|A\|\|A^{-1}\|$ ~~$\|A\|\|A^{-1}\|$~~ The condition number measures how sensitive a matrix is to change.

2. What is the condition number of solving Ax=b?

The condition number of solving $Ax = b$ is the same as the condition number of A.

3. Calculate the p-norm condition number of a matrix for a given p.

$\|A\|_p \|A^{-1}\|_p$

4. Do you want a small condition number or a large condition number?

Small condition number

5. If you have p accurate digits in a A and b, how many accurate digits do you have in the solution of Ax=b if the condition number of A is 10^k?

$p - k$

6. When solving a linear system Ax=b, does a small residual guarantee an accurate result?

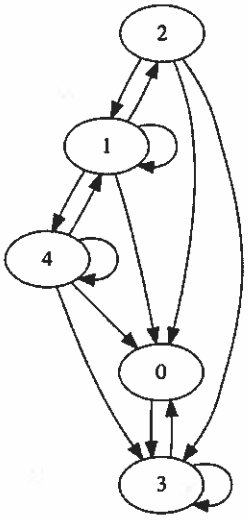
A small residual implies a small relative error only if A is well-conditioned (cond(A) is small)

7. Consider solving a linear system Ax=b. When does Gaussian elimination with partial pivoting produce a small residual?

Gaussian Elimination with partial pivoting always produces a small residual.

Graphs

1. Given a given graph (weighted or unweighted), determine the adjacency matrix



$$\begin{bmatrix} 0 & 1 & 1 & 1 & 1 \\ 0 & 1 & 1 & 0 & 1 \\ 0 & 1 & 0 & 0 & 0 \\ 1 & 0 & 1 & 1 & 1 \\ 0 & 1 & 0 & 0 & 1 \end{bmatrix}$$

2. What is a transition matrix? Given a graph representing the transitions or a description of the problem, determine the transition matrix. (Write the transition matrix for the graph above)

$$M = \begin{bmatrix} 0 & 1/4 & 1/3 & 1/2 & 1/4 \\ 0 & 1/4 & 1/3 & 0 & 1/4 \\ 0 & 1/4 & 0 & 0 & 0 \\ 1 & 0 & 1/3 & 1/2 & 1/4 \\ 0 & 1/4 & 0 & 0 & 1/4 \end{bmatrix}$$

3. Given an initial state, how can you use a transition matrix to determine the state (probability vector) after 1 timestep? After 2?

$X_1 = MX_0$

$X_2 = MX_1$

4. What properties must be true for a transition matrix?

- Sum of a column = 1
- each entry represents a probability (between 0 and 1)

Eigenvalues

1. What is the definition of an eigenvalue/eigenvector pair?

If A is an nxn matrix, $x \neq 0$ is an eigenvector of A if there exists a scalar λ such that $Ax = \lambda x$ where λ is called an eigenvalue.

2. If v is an eigenvector of A, what can we say about cv for any nonzero scalar c?

cv is also an eigenvector of A.

3. What is the relationship between the eigenvalues of A and the eigenvalues of

   a. cA for some scalar c,
   b. $(A-\sigma I)$ for some scalar σ
   c. A^{-1}?

a. λ for A $\Rightarrow$ $c\lambda$ for cA

b. λ for A $\Rightarrow$ $\lambda - \sigma$ for $(A - \sigma I)$

c. λ for A $\Rightarrow$ $1/\lambda$ for A^{-1}

4. What is the relationship between the eigenvectors of A and the eigenvectors of

   d. cA for some scalar c,
   e. $(A-\sigma I)$ for some scalar σ
   f. A^{-1}?

d. eigenvectors do not change

e. eigenvectors do not change

f. eigenvectors do not change

5. Be able to run a few steps of normalized power method.

Start with an initial guess (vector y), multiply by matrix A, normalize the result and replace your initial guess with said result, repeat the process.

* Final result will correspond to the dominant eigenvector of A!

6. When can power method (or normalized power method) fail?

→ If $\lambda_1 = -\lambda_2$, then method will not converge (this problem can't be discounted in practice)

→ risk of eventual over/underflow. but normaliging is done in practice

7. How can we approximate an eigenvalue of A given an approximate eigenvector?

If x is an eigenvector of A such that $Ax = \lambda x$,

$$\lambda = \frac{x^T A x}{x^T x}$$ (Raleigh coefficient)

8. What happens to the result of power method if the initial guess does not have any components of the dominant eigenvector? Does this depend on whether we are using finite or infinite precision?

The power iteration will converge to eigenvector u_2 using infinite precision ✓

Finite precision → converges to u_1

9. How can we find eigenvalues of a matrix other than the dominant eigenvalue?

Power Iteration — largest eigenvalue

Inverse Power Iteration — smallest eigenvalue

Inverse Shifted Power Iteration — eigenvalue close to known σ

10. Complete the table:

	(Normalized) Power Iteration	Inverse Iteration	Shifted inverse iteration	Rayleigh Quotient iteration
Converges to which eigenvalue?	Largest	Smallest	closest to known σ	finds eigval closest to σ
Cost	$O(n^2)$ kn^2	$O(n^3)$ $n^3 + 2kn^2$	$O(n^3)$ $n^3 + 2kn^2$	$O(n^3)$
Convergence rate	$\left\lvert \frac{\lambda_2}{\lambda_1} \right\rvert$	$\left\lvert \frac{\lambda_n}{\lambda_{n-1}} \right\rvert$	$\left\lvert \frac{\lambda_c - \sigma}{\lambda_{c_2} - \sigma} \right\rvert$	$> O(n^2)$

k: # iterations

↳ λ_c = closest eigenvalue to σ

λ_{c_2} = second closest eigenvalue to σ

Singular value decomposition

1. For a matrix A with SVD decomposition $A=U\Sigma V^T$, what are the columns of U and how can we find them? What are the columns of V and how can we find them? What are the entries of Σ and how can we find them?

Σ: a diagonal matrix where the diagonals are the singular values of A, or the eigenvalues of A^TA squared

V: right orthogonal vectors of A

U: left orthogonal vectors of A

2. What are the shapes of U, Σ, and V in the full SVD of an m×n matrix?

U - $m \times n$ orthogonal matrix

Σ - $m \times n$ diagonal matrix V - $n \times n$ orthogonal matrix

3. What are the shapes of U, Σ, and V in the reduced SVD of an m×n matrix?

U_R - $m \times k$ matrix V_R - $n \times k$ matrix

Σ_R - $k \times k$ matrix where $k = \min(m, n)$

4. What is the cost of computing the SVD?

$\Theta(mn^2)$ $mn^2 + n^3$ (more detailed)

5. Given an already computed SVD of a matrix A, what is the cost of using the SVD to solve a linear system Ax=b? How would you use the SVD to solve this system?

$Ax = b$

$U\Sigma V^T x = b$ where $A = U\Sigma V^T$ solve $\Sigma y = U^T b$ Eval $x = Vy$

$\Sigma V^T x = U^T b$ where $y = V^T x$ $\Theta(n^2)$ $\Theta(n^3)$

$x = V\Sigma^{-1}U^T b$

$O(n^2)$

6. Given an already computed SVD of a matrix A, what is the cost of using the SVD to solve a least squares problem Ax≅b? How do you solve a least squares problem using the SVD?

$Ax \cong b$

$A^TAx \cong A^Tb$ $\Theta(n^3)$ if A is full rank

$(U_R \Sigma_R V_R^T)^T (U_R \Sigma_R V_R^T) x = (U_R \Sigma_R V_R^T)^T b$

$V_R \Sigma_R^T U_R^T U_R \Sigma_R V_R^T x = V_R \Sigma_R^T U_R^T b$

$V_R \Sigma_R^2 V_R^T x = V_R \Sigma_R^T U_R^T b$

$\Sigma_R^2 V_R^T x = \Sigma_R^T U_R^T b$

A is full rank $x = V \Sigma_R^{-1} U_R^T b$

$\Sigma^2 V^T x = \Sigma U^T b \Rightarrow \Sigma y = U^T b \rightarrow x = Vy$

$x = V\Sigma^T U^T b$

$O(mn)$

$\Theta(n^2 + n^3)$; if A is rank deficient

7. How do you use the SVD to compute a low-rank approximation of a matrix?

$$\sum_{i=1}^{k} \sigma_i u_i v_i^T$$

8. Given the SVD of a matrix A, what is the SVD of A+ (the pseudoinverse of A)?

$$A = U \Sigma V^T$$

$$A A^+ = I$$

$$A^+ = V \Sigma^+ U^T$$

9. Given the SVD of a matrix A, what is the 2-norm of the matrix? What is the 2-norm condition number of the matrix?

$$2\text{-norm} = \sigma_{max}$$

$$\text{Cond} = \frac{\sigma_{min}}{\sigma_{max}}$$

$$\|A\|_2 = \sigma_{max}$$

$$\text{cond}_2(A) = \frac{\sigma_{max}}{\sigma_{min}} \quad \text{(full rank)}$$

$$= \infty \text{ (rank deficient)}$$

Linear Least-squares

1. What does the least-squares solution minimize?

   the solution minimizes the squared Euclidean norm of the residual vector $r = b - Ax$

   $\Rightarrow \min \|b - Ax\|_2^2$

2. For a given model and given data points, can you form the system $Ax \cong b$ for a least squares problem?

   for data points: $y_i = at_i + b$ where $\forall i \in [1, m]$

   the matrix:

$$\begin{bmatrix} t_1 & 1 \\ t_2 & 1 \\ \vdots & \vdots \\ t_n & 1 \end{bmatrix} \begin{bmatrix} a \\ b \end{bmatrix} = \begin{bmatrix} y_1 \\ y_2 \\ \vdots \\ y_n \end{bmatrix}$$

3. What are the differences between least squares data fitting and interpolation?

   interpolation: we look for the linear combination of parameters so that your function passes the points EXACTLY

   fitting: we look for the parameters so that our function BEST FITS the data points

4. Given the SVD of a matrix A, how can we use the SVD to compute the least squares solution? Consider also the case when A is rank deficient. Be able to do this for a small problem.

   So lets say $A = U \Sigma V^T$, then the least squares solution is:

$$x = \sum_{\sigma_i \neq 0} \frac{u_i^T b}{\sigma_i} v_i$$

   $\Rightarrow u_i$ is the ith colum

   $\Rightarrow v_i$ is the ith column of V

$$\underline{x} = V \Sigma_R^+ U_R^T \underline{b}$$

   when A is rank deficient, (reduced SVD)

   $\Rightarrow$ when $m > n$, $A = U_R \Sigma_R V^T$ (you "cut" off parts of U + Σ; U_R: $m \times n$)

   $\Rightarrow$ when $n > m$, $A = U \Sigma_R V_R^T$ (V_R^T: $n \times n \rightarrow m \times m$) (you "cut" off parts of Σ + V)

   after this you can use the same formula as above

5. Why would you use the SVD instead of normal equations to find the solution to $Ax \cong b$?

   while solving w/ normal equations is less costly, finding a solution w/ normal eqns worsens the conditioning of a problem

   $\Rightarrow$ changes the cond to $(\text{cond}(A))^2$ $\Rightarrow$ (bad)

6. Which costs less: solving a least squares problem via the normal equations or solving a least squares problem using the SVD?

NORMAL EQNS

Nonlinear Equations

1. For a given nonlinear equation (1D), you should be able to run a couple steps of:
   a. Bisection method

```
m = (b-a)/a
if sign(f(a)) == sign(f(b)):
    a = m
else:
    b = m
```

   b. Secant method

$$x_{k+1} = x_k - f(x_k) / f'(x_k)$$

$$f'(x_k) \approx \frac{f(x_k) - f(x_{k-1})}{(x_k - x_{k-1})}$$

   c. Newton's method

$$x_{k+1} = x_k - f(x_k) / f'(x_k)$$

2. How many function evaluations are required per iteration for bisection?

   One new function evaluation (a previous is reused).

3. What is the convergence rate of bisection method? Will it always converge?

   linear convergence

   No. f(a) and f(b) won't always have opposite signs.

   It will always converge if starting from intera such that $f(a) \cdot f(b) < 0$

4. Using the bisection method, given a specific initial interval [a,b] and a given tolerance tol, how many iterations would be required for the approximate root to be accurate to the given tolerance? Remember that the error for the bisection method is defined as the length of the interval.

$$\frac{a-b}{2^n} < tol$$

n = number of iterations

$$\log(2^n) > \log\left(\frac{b-a}{tol}\right)$$

$$n > \log\left(\frac{b-a}{tol}\right)$$

5. How many function evaluations are required per iteration for 1D Newton's method for root-finding? Which functions must be evaluated?

2 function evaluations

$f(x)$ and $f'(x)$

6. What is the convergence rate of Newton's method for solving 1D nonlinear equations?

quadratic

7. How many function evaluations are required per iteration for secant method?

one new function evaluation per iteration

8. What is the convergence rate of secant method? Will it always converge?

superlinear, $r \approx 1.618$ (golden ratio)

no, starting guesses must be near the root

9. For a given vector-valued function f(x), what is the Jacobian.

$$J(x) = \begin{pmatrix} \frac{\partial f_1(x)}{\partial x_1} & \cdots & \frac{\partial f_1(x)}{\partial x_n} \\ \vdots & \ddots & \vdots \\ \frac{\partial f_n(x)}{\partial x_1} & \cdots & \frac{\partial f_n(x)}{\partial x_n} \end{pmatrix}$$

10. For a simple system of nonlinear equations, you should be able to run one step of N-dimensional Newton's method.

$x_{k+1} = x_k + s$

$J(x_k)s = -f(x_k)$ → solve for s

11. What is the convergence rate of Newton's method for root-finding in N dimensions? Will it always converge?

quadratic,

no, only local convergence (starting guess must be close)

12. What operations are required per iteration for Newton's method in N dimensions?

computing Jacobian

solving for newton step

Optimization

1. What are the necessary and sufficient conditions for a point to be a local minimum in one dimension? For a point x, $f'(x) = 0$ and $f''(x) > 0$

2. What are the necessary and sufficient conditions for a point to be a local minimum in N dimensions?
At a point $x^* = (x_1, x_2, \ldots x_n)$ $\nabla f(x^*) = 0$ and $H_f(x^*)$ is positive definite (all eigenvalues are positive)

3. How do you classify extrema as minima, maxima, or saddle points (answer for 1D and ND)?

1d

$f'(x)=0$	$f''(x)<0$	min
$f'(x)=0$	$f''(x)>0$	max
~~$f'(x)=0$~~	~~$f''(x)=0$~~	~~saddle~~

Nd →

$H_f(x^*)$ pos def and $\nabla f(x^*)=0$	min
$H_f(x^*)$ neg def, $\nabla f(x^*)=0$	max
$H_f(x^*)$ indefinite, $\nabla f(x^*)=0$	saddle

4. Run one iteration of golden section search.

$f(x) = 4x^3 + 2x^2 + 5x + 40$

$-2 = a \quad x_1 \quad x_2 \quad b = 2$

$h_0 = b - a = 4$

$\tau = 0.618$

$x_1 = a + (1-\tau)h_0 = -2 + 0.382(4) = -0.472$

$x_2 = a + \tau h_0 = -2 + 0.618(4) = 0.472$

$f_1 = f(x_1) = 37.665$

$f_2 = f(x_2) = 43.226$

$f_2 > f_1$:

$b = x_2 = 0.472$

$x_2 = x_1 = -0.472$

$x_1 = a + (1-\tau)h_1$

$= a + 0.382(2.47$

$x_1 = -1.0557$

$a = -2$

interval after iter.

$[-2, 0.472]$

5. Calculate the gradient of a function (function has many inputs, one output).

$f(x,y) = 2x^2 + 5y + xy$

$$\nabla f = \begin{pmatrix} \partial f/\partial x \\ \partial f/\partial y \end{pmatrix} = \begin{pmatrix} 4x+y \\ 5+x \end{pmatrix} \quad \nabla f(1,1) = \begin{pmatrix} 5 \\ 6 \end{pmatrix}$$

6. Calculate the Hessian of a function.

$$H_f = \begin{bmatrix} \partial^2 f/\partial x_1^2 & \partial^2 f/\partial x_1 \partial x_2 & \cdots & \partial^2 f/\partial x_1 \partial x_n \\ \partial^2 f/\partial x_2 \partial x_1 & & & \\ \vdots & & & \vdots \\ \partial^2 f/\partial x_n \partial x_1 & \cdots & \cdots & \partial^2 f/\partial x_n^2 \end{bmatrix}$$

$* \quad \frac{\partial^2 f}{\partial x \partial x_2} = \frac{\partial f}{\partial x}\left(\frac{\partial f}{\partial x_2}(x)\right)$

↑ composition

7. Find the search direction in steepest/gradient descent.

Evaluated using $-\nabla f(x_k)$

8. Why must you perform a line search each step of gradient descent?

To get how far along the gradient to travel

9. Run one step of Newton's method in one dimension.

ex. $f(x) = 4x^3 + 2x^2 + 5x + 40$, initial guess $x_0 = 2$

$f'(x) = 12x^2 + 4x + 5$ $\quad f'(2) = 12(4) + 4(2) + 5 = 61$

$f''(x) = 24x + 4$ $\quad f''(2) = 52$

$$x_1 = x_0 - \frac{f'(x_0)}{f''(x_0)} = 2 - \frac{61}{52} = 0.8269$$

10. Run one step of Newton's method in N dimensions.

$f(x,y) = 3x^2 + 2y^2$, with some guess $X_0 = \begin{bmatrix} x_0 \\ y_0 \end{bmatrix}$

$$\nabla f = \begin{bmatrix} 6x \\ 4y \end{bmatrix} \quad H_f = \begin{bmatrix} 6 & 0 \\ 0 & 4 \end{bmatrix} \quad X_1 = X_0 - H_f^{-1} \nabla f \begin{bmatrix} x_0 \\ y_0 \end{bmatrix}$$

11. What operations do you need to perform each iteration of golden section search?

Only 1 of $f(x_1)$, $f(x_2)$ → perform $f(x_2)$ if we changed x_2

$f(x_1)$ if we changed x_1

12. What operations do you need to perform each iteration of Newton's method in one dimension?

$f'(x)$, $f''(x)$

13. What operations do you need to perform each iteration of Newton's method in N dimensions?

∇f, H_f (Hessian), Solve $H_f(x_k)(s_k) = -\nabla f(x_k)$

(s_k: step)