1 Introduction: Complexity and Stability

Complexity: the minimum time needed to solve a problem. If no discrete complexity is known find lower and upper bounds on computing time. Assume real number model when calculating complexity. Consequently, complexity does not depend on the algorithm or size of numbers. It is controled only by the computational difficulty of the problem and the set of permissable operations.

Real Number Model: Arithmetic operations can be performed exactly with unit cost. No matter how large the numbers being operated upon are the cost of the operation remains the same. The assumption is valid since most modern computers use floating point arithmetic.

Floating Point Arithmetic: Every number is represented by $x = m2^c$ where $\frac{1}{2} \leq \|m\| < 1$ and $c \in Integer$. The error in the floating point operation is related to the mantissa bit t.

$$m \approx \sum_{i=1}^{t} b_i 2^{-t} \quad b_i \in [0, 1]$$

If $t = \infty$ no rounding errors would exists. But t is finite and consequently $fl(x) = x(1 - \epsilon x)$ where $\|\epsilon x\| = 2^{-t}$.

Numerical Stability: the computed result is the exact solution for a slightly perturbed data.

2 Matrix Manipulation

Algebraic Complexity: the minimum time needed to solve a problem when complete information is given and exact solutions can be found.

Matrix Multiplication: $AB = C$
Input: $A = a_{ij}, B = b_{ij} \quad A, B \in \Re^2$
Output: $C = c_{ij} \quad C \in \Re^2$

$$c_{ij} = a_{ik} b_{ki}$$

Full matricies with no given structure are generally solved by gaussian elimination or housholders algorithm with in order n^3 computations. Large sparse matricies can be solved by more efficient algorithms, about order n.

1. Typical Algorithm
   For example gaussian elimination and hausholder's algorithm.

   $$(2n^2)(n)(n-1) \approx 2n^3$$

   This class of algorithm exhibits weak stability. The formulation of the error depends on the order of the indicies.

   $$fl(c_{ij}) = \sum_{k=1}^{n} a_{ik} b_{kj} (1 - \epsilon_{i,j,k})$$

   $\tilde{A}B$ or $A\tilde{B}$ is the slightly perturbed input.

2. **Strassen's Algorithm**
   Perhaps an improvement on typical approach. Break Problem into Quarters and Solve Recursively.

   $$A = \begin{pmatrix} A_{11} & A_{12} \\ A_{21} & A_{22} \end{pmatrix}, \quad B = \begin{pmatrix} B_{11} & B_{12} \\ B_{21} & B_{22} \end{pmatrix}$$

   $$C = AB = \begin{pmatrix} C11 & C12 \\ C21 & C22 \end{pmatrix}, \quad n = 2^k$$

   Seven values need to be solve each loop.

$$\begin{aligned} Q_1 &= (A_{11} + A_{22})(B_{11} + B_{22}), & C_{11} &= Q_1 + Q_4 - Q_5 + Q_7 \\ Q_2 &= (A_{21} + A_{22})(B_{11}), & C_{21} &= Q_2 + Q_4 \\ Q_3 &= A_{11}(B_{12} + B_{22}), & C_{12} &= Q_3 + Q_5 \\ Q_4 &= A_{22}(-B_{11} + B_{21}), & C_{22} &= Q_1 + Q_3 - Q_2 + Q_6 \\ Q_5 &= (A_{11} + A_{12})B_{22}), & & \\ Q_6 &= (-A_{11} + A_{21})(B_{11} + B_{12}) & & \\ Q_7 &= (A_{11} - A_{22})(B_{21} + B_{22}) & & \end{aligned}$$

The cost of the algorithm is $\approx n^{2.81}$. No proof exists that this is the best possible method. Gaussian elimination is still more practical. **Strassen's Algorithm** is only efficient when n is very big epecially since recursion is more costly than iterative operations. However, most large matricies are sparce and can be solved in On.

3 Polynomials, Polynomial Equations

Again analyze the algebraic complexity. Complete information is known

$$P_n(x) = a_n x^n + a_{n-l} x^{n-l} + \cdots + a_1 x + a_0 \quad a_i \in \Re, a_n \neq 0$$

Calculate the value of $P_n(x), P_n'(x), P_n''(x)$ using **Horner's Rule**. The complexity of the algorithm is n multiplications and n additions: optimal complexity. The algorithm is stable as well.

$$P_n(x) = a_0 + x(a_1 + x(\cdots + x(a_{n-1} + x(a_n))\cdots)$$

$$fl(P_n(x)) = \sum_{i=0}^{n} a_i x^i (1 - \epsilon_i), \|\epsilon\| \leq (2n+1)$$

The derivatives can be calculated using a permutation of the same algorithm. Apply the chain rule of integration.

```
s:=an
for i:=n-1 step -1 until 0 do
{
s2:=2*s1+s2*x
s1:=s+s1*x
s:=x*s+a[i]
}
Pn'':=s2
Pn':=s1
Pn:=s
```

This process can be formulated for the n^{th} derivative. It can be proved by induction see appendix.

```
s[m]:=0
s[m-1]:=0
...
s2:=0
s1:=0
s:=an
for i=n-1 step -1 until 0 do
{
```

```
s[m]:=m*s[m-1]+s[m]*x
s[m-1]:=(m-1)*s[m-2]+s[m-1]*x
...
s2:=2*s1+s2*x
s1:=s+s1*x
s:=x*s+a[i]
}
Pn[m]:=sm
Pn[m-1]:=s[m-1]
...
Pn'':=s2
Pn':=s1
Pn:=s
```

Next, apply **Horner's Rule** and its extension to finding the roots of a polynomial equation. $P_n(x) = 0$ can only be solved in closed form for $n < 4$. Other methods such as **Newton's Method**, **Secant Method**, the **Bisection Method**, or **Laguerre's Iteration** need to be applied.

Laguerre's Iteration: Make an initial guess x_o and solve the problem iteratively using the following equation:

$$x_{k+1} = x_k - \frac{np_n(x_k)}{P'_n(x_k) + sign(P'_n(x_k)\sqrt{H(x_k)}}$$

$$H(x) = (n-1)[(n-1)(P'_n(x)^2 - nP_n(x)P''_m(x)$$

The method produces a cubic convergent solution (slow). Also, the convergence is not always stable. **Williamson's Example** shows a case of false convergence.

$$P_{20}(x) = (x-1)(x-2)\cdots(x-20) \quad \tilde{P}_20(x)$$

The iterative solution of this example has many complex roots. Other methods like **deflation** are more effective for such problems.

Deflation: Make one good approximation of a zero $z = x_n$ then $p_n(x) = (x - z)p_{n-1}(x) + b_n$. If z is chosen well $p_{n-1}(x) \approx 0$.

4 Systems of Linear Equations

Algorithms used to solve systems of linear equationsn can be analyzed with algebraic complexity. Linear quations are of the form $Ax = b$ where $A \in \Re^2, x \in \Re, b \in \Re$.

Case I: n small ≤ 100. No assumptions are made on the structure of the A matrix.

Case II: n large $\approx 10^5, 10^6$. Typically the A matrix is sparse.

To solve **Case I** use factorization. The functions below are solved in order n^3.

1. Gaussian elimination with pivoting

$$A = LU$$

A: Model of permutations of rows and columns
L: Lower Triangular Matrix
U: Upper Triangular Matrix

Process of method is as follows:

(a) pivot matrix so that the largest values is in the [1,1] position

(b) look at the first column

$$\vec{a} = a_{i1} = \begin{pmatrix} a_{11} \\ a_{21} \\ a_{31} \\ \vdots \\ a_{n1} \end{pmatrix} \quad [L_1] = L_{i1} = \begin{pmatrix} 1 & 0 & 0 & \cdots & 0 \\ -L_{21} & 1 & 0 & \cdots & 0 \\ -L_{31} & 0 & 1 & \cdots & 0 \\ \vdots & & & & \vdots \\ -L_{n1} & 0 & 0 & \cdots & 1 \end{pmatrix}$$

$$L_1\vec{a} = \begin{pmatrix} a_{11} \\ -L_{21}a_{11} + a_2 \\ -L_{31}a_{11} + a_3 \\ \vdots \\ -L_{n1}a_{11} + a_{n1} \end{pmatrix}$$

(c) let $L_{i1} = \frac{a_{i1}}{a_{n1}}$ so that $\mathrm{L}_i\vec{a}_i = (\, aii \quad 0 \quad 0 \quad \cdots \quad 0\,)^T$ thus:

$$L_1A = \begin{pmatrix} a_{11} & a_{21} & a_{31} & \cdots & a_{n1} \\ 0 & x & x & \cdots & x \\ \vdots & \vdots & \vdots & & \vdots \\ 0 & x & x & \cdots & x \end{pmatrix}$$

2. Householder's Algorithm

$$A = QU$$

A: Model of permutations of rows and columns
Q: Orthogonal Matrix
U: Upper Triangular Matrix

To solve **Case II** use iteration. The maticies are usually spare and thus the operation Az can usual be computed in the cost and space of Θn.

Transform $Ax = b$ into an iteration $x = Bx+g$. Notice that $y = Ax$, $y = Bx$ and g are easy to compute. x_0 is an initial guess.

$$x_k = Bx_{k-1} + g$$

The iteration converges if $\rho(B) < 1$ where $\rho(B)$ is the spectral radius [1] of the matrix B.

The error of such an iteration is of the following form

$$||x_k - x^*||_\epsilon \leq (\rho(B) + \epsilon)^k \ ||x_0 - x^*||_\epsilon$$

If $B = B^T$, the symmetric case, and the error ϵ is 0 then, $\rho^k(B) = ||A||_2^k$ where A is the original matrix $Ax = b$.

Usually polynomial iterations converge more rapidly. In such an iteration the matrix A is assumed to be some polynomial function. The zero $poly(A)x-b = 0$ can be found using known properties of polynomials. Also, such iterations rely only on partial information about the polynomial function.

$$x_{i+1} = \phi(x_i, N(f, x_i))$$

where x_i is the current iterative step, N is the partial information function and x_{i+1} is the value of the next step.

The measure of an iteration is the **index** Φ of iteration

$$\Phi = \frac{cost\ of\ one\ step}{\log p}$$

where p is the order of the iteration.

5 Newton and Secant Algorithms

1. Newton's Method

   $$N(f, x_i)) = [f(x_i), f'(x_i)]$$

   The interpolatory polynomial for Newton's method is a line with value $f(x_i)$ at point x_i and the slope of $f'(x_i)$ at that same point. The line crosses zero at

   $$x_{i+1} = x_i + f(x_i)/f'(x_i)$$

   consequently x_{i+1} is the next value of the iteration.

   The convergence of Newton's Method according to taylor series expansion is quartic.

   $$(x_{i+1} - \alpha) = \Theta(x_i - \alpha)^2$$

[1] the spectral radius of a matrix is the absolute value of the largest magnitude eigenvalue of that matrix. $\rho(b) = \max\{|\lambda| \text{ where } Bx = \lambda x; \quad x \neq 0\}$

2. Secant Algorithm

$$N(f, x_i)) = [f(x_i), f(x_{i-1})]$$

The secant algorithm does not rely on derivatives. The interpolatory polynomial for the Secant algorithm is also a line, the the slope is estimated using the value of the previous step. The line crosses zero at

$$x_{i+1} = x_i + f(x_i)\frac{(x_i - x_{i-1})}{f(x_i) - f(x_{i-1})}$$

consequently x_{i+1} is the next value of the iteration.

The convergence of the Secant Algorithm is $\frac{1+\sqrt{5}}{2}$ according to taylor series expansion.

$$(x_{i+1} - \alpha) = \Theta(x_i - \alpha)(x_{i-1} - \alpha)$$

6 Information Based Complexity: Problems of Partial Information

Lipschitz functions: A lipschitz function is defined as follows for

$$f : [0, 1] \in \Re$$

$$|f(t) - f(x)| \leq \mathbf{L}|t - x| \quad \forall\, t, x \in [0, 1]$$

To integrate a lipschitz function finc the value of the function at the best sample points $t_i = \frac{1}{2n} + \frac{i-1}{n}$, $i = 1, 2, ..., n$. The number of iterations required to increase precision by ϵ is $n = \lceil \frac{1}{4\epsilon} \rceil$.

$$\int_0^1 f(x)dx \approx \sum_{i=1}^{n} \frac{1}{2n}(t_i + t_{i+1})$$

For Lipschitz functions the set of information can not be optimized by adaption.

Bisection: The bisection method finds zeros of functions by looking (1) at the start and endpoint of the function to check if a zero occurs ther and if not that the endpoints are of different signs (2) checks the value in half way between the two endpoints and chooses the new set of endpoints as the span with different signs (3) the process continues untill the zero is found within acceptable error.

The bisection method also uses partial information, however adaption is essential to this algorithm.

7 Scalar Integration

$$S(f) = \int_a^b f(x)dx$$

Classically, Scalar integration is performed using partial informtion to construct an interpolatory polynomial of degree $\leq k$ and integrating the polynomial.

quadrature: A linear combination of $f(x_i)$ used to solve an approximate integration.

$$\int_a^b [f(x)]_{poly}dx = \sum_{i=1}^{k} f(x_i) \int_a^b l_i(x) = \sum_{i=1}^{k} f(x_i)q_i(x)$$

error: The error of such integration can be found using the trapezoidal rule. The effectiveness of such an approach depends on the choice of values (x_i) and the smoothness of the function f.

Smolyah's Theorem: If the set of functions F is balanced and convex [2] there exists an optimal quadrature Φ which is **linear**.

8 Multivariate Integration

$$f(x) = \begin{pmatrix} f_1(\vec{x}) \\ f_2(\vec{x}) \\ \vdots \\ f_d(\vec{x}) \end{pmatrix} \quad f_i(\vec{x}) \subset D \in \Re^1$$

$$\nabla f(\vec{x}) = (\frac{\partial f_i}{\partial \vec{x}_j}\vec{x})_{i,j=1}^{d} \quad \in \Re^{d \times d}$$

Multivariate Taylor's theorem:

$$f(\vec{x}) = \sum_{n=1}^{\infty} a_n(\vec{x} - \vec{x}_i)$$

where $a_n = \frac{\nabla^n f}{n!}$.

Multivariate Newton's Iteration: As before $x_{i+1} = \vec{x}_i + f(\vec{x}_i)\ (\nabla f(\vec{x}_i))^{-1}$ only now calculating the inverse of the gradient is a matrix operation requiring gaussian elimination or some other inverting operation with cost of order d^3. The remaining vector combinations are of order d. Again the order of the newton iteration is 2.

[2] balanced: if $f \in F$ then $-f \in F$, convex: any two points in the function space can be connected by a polygonal line which does not cross out of the space.

Multivariate Secant Iteration: As before $\vec{x}_{i+1} = \vec{x}_i + f(x_i)(\Delta X_i)(\Delta F_i)^{-1}$ both ΔX_i and ΔF_i must be nonsingular. Again calculating the inverse of a matrix is an order d^3 operation. Conditions upon multivariate secant series convergence include

1. D is open and convex and $\vec{\alpha}$ is a simple zero of an f' Lipschitz function.
2. there exists a $c \in (0, 1]$ such that the absolute value of the determinant of the normalized differences is greater than that c.
3. $x_0, x_1, x_2, ..., x_d$ is sufficiently close to $\vec{\alpha}$.

Then the oder of convergence of the series is P^{d+1} where P is the oder of convergence of the method and d is the dimension of the vectors.

9 Intractability on the worst case setting

In the worst case setting $\sup_{f(x)\in F} |N(x)|$ an algorithm can be intractible while it is strongly tractible in the Randomized setting of the Average case setting.

discrepency: the discrepency of a vector is defined as

$$disc(\vec{x}) = x_1 x_2 ... x_d - \frac{1}{n}\chi_{[0,x]}(t_i)$$

where

$$\chi_{[0,x]}(t_i) = \begin{array}{ll} 1 & t_i \in [0, x) \\ 0 & t_i \notin [0, x) \end{array}$$

The discrepency of a sequence of values is defined as

$$disc(t_1, t_2, ...t_n) = \left(\int_{[0,1]^d} disc^2(\vec{x}) dx \right)^{1/2}$$

also,

$$disc(n, d) = \inf_{t_1 ... t_n \in [0,1]^d} disc(t_1 ... t_n) = \theta \frac{1}{n} (\ln n)^{\frac{d-1}{2}}$$

low discrepency sequence: For a low discrepency sequence

$$disc(t1, t2, ..., tn) \leq Cd \frac{1}{n} (\ln n)^{\frac{d-1}{2}}$$

The solution is intractible for large d. It suffers from the curse of dimensionality.

10 Randomized Setting:

Quasi Monte Carlo: Usually t_i are chosen as low discrepancy points.

$$Q(f) = \frac{1}{n}\sum_{i=1}^{n} f(t_i)$$

Monte Carlo: t_i are chosen randomly. For piecewise continuous functions

$$\left(\int_{[0,1]^d} f^2(x)dx\right)^{\frac{1}{2}} \leq M$$

The error $e_n(MC) \leq \frac{M}{\sqrt{n}}$ and the number of iterations required to decrease the initial error by *epsilon*.

$$n = \Theta(\frac{M}{\epsilon})^2$$

11 Weighted Classes

Weight the values with an ordered set of variables

$$\gamma_1 \geq \gamma_2 \geq \gamma_3 \cdots \geq 0$$

$$\gamma_u = \prod_{j \in u} \gamma_j$$

Weighted Discrepaency:

$$disc^2_\gamma(t_1, ..., t_n) = \sum_{phi \not\in u \subset D} \gamma_u \int_{[0,1]^{|u|}} disc^2(x_u, 1)dx_u$$

Weighted Norm:

$$||f||^2_\gamma = \sum_{u \subset D} \frac{1}{\gamma_u} \int_{[0,1]^{|u|}} (\frac{\partial^{|u|} f}{\partial x_u})^2 dx_u$$

Using Zarembe's Identity find

$$disc_\gamma(t_1, ..., t_n) = \sup_{f \in F || f ||_\gamma \leq 1} \left| \int_{[0,1]^d} f(x)dx - \frac{1}{n}\sum_{i=1}^{n} f(t_i) \right| = e_n(t_1, ..., t_n)$$

Tractibility: Polynomial dependence on parameters. The weighted low discrepency set has the following tractible solution $n(\epsilon, d) \leq C\ d^q \epsilon^{-p}$. If $q = 0$ the solution is strongly tractible.

Compare this solution to the worst case stetting sollution which is intractible.

12 Average Case Setting

For integration:

$$e^{avg}(\phi, N) = \left(\int_F ||S(f) - \phi(N(f))||^2 \mu(df) \right)$$

Usually μ is a Gaussian measure. Recall

$$\mu(A) = \frac{1}{(2\pi)^{\frac{n}{2}} \sqrt{\sigma_1 \sigma_2 ... \sigma_n}} \int_A e^{-\sum_{i=1}^n \frac{x^2}{2\sigma_i}} dx_1 dx_2 ... dx_n$$

σ_i are the variances.

Mean:

$$\int_F f(x)\mu(df) = m_\mu(x), \quad \forall x \in D$$

Usually assume that $m_\mu = 0$ symmetric measure.

Covariance:

$$\int_F f(x)f(t)\mu(df) = K(x,t), \quad \forall x,t \in D$$

K is symmetric and non-negative definite,
$K(x,t) = K(t,x)$ and $K(x,x) \geq 0$.

Wiener Measure:

$$\int_F f(x)f(t)\mu(df) = K(x,t) = \prod_{i=1}^n \min(x_i, t_i)$$

F is the class of continuous functions defined on $D = [0,1]^d$.

Notice that the wiener measure is related to the discrepancy.

$$\int_F (S(f) - \frac{1}{n} \sum_{i=1}^n f(t_i))^2 w(df) = disc^2(1 - t_1, 1 - t_2, ..., 1 - t_n)$$

accordingly the error is that of the discrepency and is tractible. Approximation is also tractible. For integration:

$$comp^{avg}(\epsilon, d) \leq 7.26 \left(\frac{1}{\epsilon} \right)^{2.454}$$

For approximation:

$$comp^{avg}(\epsilon, d) \leq 2.38 \left(\frac{1}{\epsilon} \right)^{5.672}$$