CSCD37H - Analysis of Numerical Algorithms for Computational Mathematics

©
W.H. Enright
Computer and Mathematical Sciences Division,
University of Toronto at Scarborough
(enright@cs.utoronto.ca)

1 Overview and Course Organization

1.1 General Information and Mathematical Background [2 weeks]

- Administration Details:
  Grading Scheme, Course Website, Tutorials, Office hours Prerequisites etc.
- What is Numerical Analysis ?
  The generic difficulty, Mathematical Modelling and Numerical issues, Conditioning and Stability of a problem, The need to approximate, Using existing methods as implemented in available Software libraries
- An interesting example showing that Numerical Algorithms can be very important (and developing them can be rewarding).
- Mathematical Preliminaries and Notation:
  Floating Point arithmetic, Relevant notation and results/theorems from Linear Algebra and Calculus.

1.2 QR Decomposition and Applications [4 weeks]

- Review of Gaussian Elimination [2.4]
- Matrix Norms and Condition Number [2.3]
- QR Decomposition
- Solving Linear Systems with QR [3.5]
- Linear Least Squares (LLSQ) Problems [3.1, 3.5]
- Eigenvalue Problems [4.1 – 4.5]

1.3 Nonlinear Systems and Optimization [2 weeks]

- Review of Methods for Scalar (1 dimensional) Problems [5.1 – 5.5]
- Extension of Newton's method to Systems of Nonlinear Equations [5.6]
- Optimization Problems [6.1]

1.4 Numerical methods for Ordinary Differential Equations [4 weeks]

- Motivation and Mathematical Preliminaries [5.1]
- Taylor Series Methods [5.2, 5.3]
- Runge-Kutta Methods [5.4]
- Higher-order Runge-Kutta Methods [5.4, 5.5]
- Error Estimates [5.5]
- Stepsize Control [5.5]

1.5 Gauss Quadrature and Multidimensional integration [optional]

- Errors in Interpolatory Rules [4.3]
- Orthogonal Polynomials and Gauss Quadrature [4.7]
- Two Dimensional Quadrature [4.8]

2 General information / Mathematical Background

2.1 What is Numerical Analysis?

- Consider the investigation of a well defined mathematical model arising in any application area. The problem is 'well defined' in the sense that there exists a 'solution' and it is unique.
- In addition to investigating the solution of these models, we are interested in the 'Conditioning' (or sensitivity) of the underlying mathematical problem. That is, do small changes in the data defining the problem lead to 'small' changes in the exact (unique) solution?
- For virtually all mathematical models of practical interest one cannot determine a useful 'closed form' expression for the exact solution and one must approximate the exact solution.

- In Numerical Analysis we develop and analyse algorithms to approximate the exact solution to mathematical problems.
  - Algorithms must be constructive and finite (time and space).
  - We will analyse the errors in the approximation.
  - We will also quantify the stability and efficiency of the algorithms.
- The focus of this course is on the intelligent use of existing algorithms embedded in widely used numerical software. (It is NOT focused on deriving algorithms or writing code.)
  - How to interpret the numerical (approximate) results.
  - What method (algorithm) should be used.
  - Can a 'standard' method (eg. a library routine) be applied to a particular problem?
  - What methods are available in the usual 'Problem Solving Environments' that scientists, engineers and students work in. For example in MATLAB, MAPLE or Mathematica.
  - In order to appreciate the limitations of the methods we will work with (primarily in MATLAB) we will analyse and understand the underlying algorithms on which the methods are based.

2.2 An Example of An Important Numerical Algorithm:

A readable reference providing more details for this example is [A $25 Billion Dollar Eigenvector Algorithm, SIAM Review, September 2006, pp. 569-581.] This example is concerned with the key insight that led to the development of the effective algorithm behind the Google search engine. To understand this development, we must first identify the tasks of a search engine:

- Locate and access all public web pages.
- Identify those pages that satisfy a search criteria. Let this set of pages be $p_1, p_2, \dots p_n$.
- Rank these 'hit pages' in order of their importance.
  1. The important pages (the most relevant) must be listed first.
  2. This ordering is accomplished using a Page Rank Algorithm.
  3. A 'score', x_i, is assigned to each page, $p_i, i = 1, 2 \dots n$ with $x_i \geq 0$.
- Pages are returned (listed) in order of decreasing scores.

Consider using a directed graph, G, to represent the set of all 'hit' pages with vertices, $v_1, v_2 \dots v_n$ and with an edge (v_i, v_k) iff p_i has a link to p_k. Now one assumes that a page is relatively important (an authority) if several pages link to it (in particular if important pages link to it). The key observation behind a Google search is that the links can be as important as the contents when determining the ordering (of the 'hit' pages).

For example consider the case where there are four hit pages represented by, v_1, v_2, v_3, v_4, where the first page has links to p_2, p_3, p_4; the second page has links to to p_3, p_4; the third page has a link to p_1; and the fourth page has links to p_1, p_3. This graph can be represented by its adjacency (or incidence) matrix, A:

$$A = \begin{bmatrix} 0 & 1 & 1 & 1 \\ 0 & 0 & 1 & 1 \\ 1 & 0 & 0 & 0 \\ 1 & 0 & 1 & 0 \end{bmatrix}.$$

Note that, with this representation of the hit pages, and with, $e = (1, 1 \ldots 1)^T$ we have:

1.
$$A\, e \ = (m_1, m_2 \ldots m_n)^T,$$
where m_i is the number of pages that p_i links to and similarly the i^{th} component of $A^T\, e$ is the number of pages that have links to p_i.

2. If x_i is the score of p_i and $x = (x_1, x_2 \ldots x_n)^T$ then,
$$y \ \equiv \ A^T\, x \ = \ (y_1, y_2 \ldots y_n)^T,$$
where y_i is the sum of all the scores of pages that link to p_i.

3. Therefore, a natural definition for x_i is y_i. That is, the vector x is identified by $x \ = \ A^T x$ or $A^T x \ = \ x$. This implies that x is an eigenvector of A^T corresponding to the eigenvalue $\lambda \ = \ 1$.

4. This definition for x_i gives too much influence to those pages with lots of links (m_i large) and we can improve the measure of importance by modifying our definition of G and A, by assigning a weight of $1/m_i$ to the edge from v_i to v_k (if it exists). That is, for the above example, A is modified and becomes:
$$A = \begin{bmatrix} 0 & 1/3 & 1/3 & 1/3 \\ 0 & 0 & 1/2 & 1/2 \\ 1 & 0 & 0 & 0 \\ 1/2 & 0 & 1/2 & 0 \end{bmatrix}.$$

5. With this modified definition (for A as well as the corresponding x and y) we will always have:

   (a) The rows of A sum to 1. That is, $A\, e \ = \ e$, and therefore 1 is an eigenvalue of A with an associated eigenvector $v = e$.

(b) The corresponding importance of $p_i,\ y_i$, is then (as above), the i^{th} component of $y \ = \ A^T \ x$.

We have shown that an appropriate score vector, x, is the solution of:

$$x \ = \ A^T x \ \text{ or } A^T x \ = \ x.$$

Note that $\lambda = 1$ is an eigenvalue of A^T, since it is an eigenvalue of A. As a result of this observation, a suitable Page Rank Algorithm can be designed based on finding the eigenvalue of A^T corresponding to the 'known' eigenvalue $\lambda = 1$.

Questions:

1. Is such an x unique and does it matter?
2. Will the resulting scores all be non-negative (ie. $x_i \geq 0$)?
3. Is there a fast algorithm for computing x?

For the above example, with $n = 4$, the eigenvector is $v = (12, 4, 9, 6)^T$, which when normalised becomes, $x = v/\|v\|_2 \ = \ (.72, .24, .54, .36)^T$.

2.3 Mathematical Preliminaries and Notation (A Review):

1. Floating Point Arithmetic (from CSCC37H or CSCB70H):
   Recall that a floating point number system, Z, can be characterized by four parameters, (β, s, m, M), and each element of Z is defined by:

   $$z = .d_1 d_2 \cdots d_s \times \beta^e,$$

   where $d_1 \neq 0$, $0 \leq d_i \leq \beta$, and $m \leq e \leq M$.

   The floating point representation mapping, $fl(x)$, is a mapping from the Reals to Z that satisfies:

   $$fl(x) = x(1+\epsilon), \quad \text{with } \ |\epsilon| \leq \mu.$$

   where μ is the 'unit roundoff' and is defined to be $1/2 \ \beta^{1-s}$.

   For any standard elementary arithmetic operation (+, -, $\times$ and /), we have the corresponding F.P. approximation (denoted by $\oplus, \ominus, \otimes$ and $\oslash$) which satisfies, for any $a, b \in Z$,

   $$a \odot b = fl(a \cdot b) = (a \cdot b)(1+\epsilon),$$

   where $|\epsilon| \leq \mu$ and $\cdot$ is any elementary operation.

   For any real-valued function, $F(a_1, a_2, \cdots a_n)$, the most we can expect is that the floating point implementation $\bar{F}$, will return (when invoked) the value $\bar{y}$ satisfying:

   $$\begin{aligned} \bar{y} &= \bar{F}(fl(a_1), fl(a_2), \cdots fl(a_n)), \\ &= \bar{F}(a_1(1+\epsilon_1), a_2(1+\epsilon_2), \cdots a_n(1+\epsilon_n)), \\ &= fl(F(a_1(1+\epsilon_1), a_2(1+\epsilon_2), \cdots a_n(1+\epsilon_n)). \end{aligned}$$

Therefor if $y = F(a_1, a_2, \cdots a_n)$ is the desired result (defined by exact arithmetic over the Reals), the computed value (computed in FP arithmetic), $\bar{y}$, will at best satisfy (for differentiable functions, F):

$$\begin{aligned} \frac{|\bar{y} - y|}{|y|} &\leq \frac{\|(\frac{\partial F}{\partial \underline{x}})^T\| \, \|\epsilon\|}{\|F\|}, \\ &\leq \frac{\|\frac{\partial F}{\partial \underline{x}}\| \, \|\epsilon\|}{\|F\|}, \end{aligned}$$

where $\epsilon = [\epsilon_1, \epsilon_2 \cdots \epsilon_n]^T$, and

$$(\frac{\partial F}{\partial \underline{x}})^T = [\frac{\partial F}{\partial x_1}, \frac{\partial F}{\partial x_2}, \cdots \frac{\partial F}{\partial x_n}],$$

evaluated at $\underline{x} = (a_1, a_2, \cdots a_n)$. From this expression we see that we could be in trouble with an inherent (independent of the approximation used) amplification of relative error whenever

$$\frac{\|\frac{\partial F}{\partial \underline{x}}\|}{\|F\|}$$

is large.

2. Linear Algebra – Notation and Review (from MATB24H and CSCC37H):
We will first review results from Linear Algebra. In doing so we introduce our notation and recall the standard definitions and results which you have seen from previous work.

   (a) The $n \times m$ matrix, A is represented by,

$$A = \begin{bmatrix} a_{11} & a_{12} & \cdots & a_{1m} \\ a_{21} & a_{22} & \cdots & a_{2m} \\ \vdots & \vdots & \vdots & \vdots \\ a_{n1} & a_{n2} & \cdots & a_{nn} \end{bmatrix}, \text{ where } a_{ij} \in \Re.$$

   $\Re^{n\times m}$ denotes the set of all such matrices.

   (b) Basic Definitions:

      i. The elements $\{a_{ii} : i = 1, 2 \cdots \min(n, m)\}$ form the diagonal of A.

      ii. $\{a_{i\,i+1} : i = 1, 2 \cdots \min(n, m-1)\}$ is the superdiagonal of A.

      iii. $\{a_{i\,i-1} : i = 2, 3 \cdots \min(n, m+1)\}$ is the subdiagonal of A.

      iv. A is Lower Triangular if $a_{ij} = 0$ for $i < j$. A is Upper Triangular if $a_{ij} = 0$ for $i > j$. Furthermore we will say that A is 'strictly' Lower (Upper) Triangular if it is Lower (Upper) Triangular and the diagonal of A is $= 0$.

(c) Matrix Multiplication:
If A and B are both $n \times n$ (square) matrices then the product is,

$$C = A\ B, \text{ where } C \equiv [c_{ij}]$$

and c_{ij} is the inner product of row i of A with column j of B. That is,

$$c_{ij} \equiv \sum_{r=1}^{n} a_{ir} b_{rj}.$$

For nonsquare matrices ($m \neq n$) the definition of matrix multiplication holds as well provided the inner product is well defined.

(d) Properties of Matrix Multiplication:

- Matrix Multiplication is Associative:

$$A(BC) = (AB)C.$$

- Matrix Multiplication is not Commutative:

$$AB \text{ may not } = BA.$$

- The cancellation law does not hold in general. That is

$$AB = AC \text{ and } A \neq \begin{bmatrix} 0 & 0 & \cdots & 0 \\ 0 & 0 & \cdots & 0 \\ \vdots & \vdots & \vdots & \vdots \\ 0 & 0 & \cdots & 0 \end{bmatrix},$$

does not imply $B = C$. To see this consider the example,

$$A = \begin{bmatrix} 1 & 0 \\ 0 & 0 \end{bmatrix}, B = \begin{bmatrix} 0 & 0 \\ 0 & 1 \end{bmatrix}, C = \begin{bmatrix} 0 & 0 \\ 0 & 0 \end{bmatrix}.$$

- The unit element for matrix multiplication is the identity matrix, denoted by I_n or $diag(1, 1 \cdots 1)$. It is the $n \times n$ square matrix defined by,

$$I_n \equiv \begin{bmatrix} 1 & 0 & \cdots & 0 \\ 0 & 1 & \cdots & 0 \\ \vdots & \vdots & \vdots & \vdots \\ 0 & 0 & \cdots & 1 \end{bmatrix}.$$

For any $A \in \Re^{n \times m}$ we have

$$I_n A = A I_m = A.$$

Note that we will often write I for I_n if the dimension is obvious from the context of the expression.

(e) The Transpose of a matrix:

- For $A \in \Re^{n\times m}$, $A^T \in \Re^{m\times n}$ and is defined by,

$$A^T \equiv (\alpha_{ij}), i = 1, 2 \cdots m,\ j = 1, 2 \cdots n,$$

where

$$\alpha_{ij} = a_{ji}.$$

Note that A^T is called the transpose of A and can be considered the 'reflection' of A about the diagonal.

- For vectors $\underline{x} \in \Re^{n\times 1}$ we have,

$$\underline{x} = \begin{bmatrix} x_1 \\ x_2 \\ \vdots \\ x_n \end{bmatrix}, x^T = [x_1, x_2 \cdots x_n], x^T \in \Re^{1\times n}.$$

- The matrix A is symmetric iff $A = A^T$.
- Properties of matrix products: For matrices A and B with the dimensions such that the products and sums are well defined we have,
  i. $(A^T)^T = A$
  ii. $(A + B)^T = A^T + B^T$
  iii. For $\lambda \in \Re$, $(\lambda A)^T = \lambda A^T$
  iv. $(A\ B)^T = B^T A^T$
  v. $A^T A$ and $A\ A^T$ are symmetric

(f) Solving Linear Equations:

i. From mathematics we know that the problem,

$$Ax = b,$$

where $A \in \Re^{n\times n}$ and $b,\ x \in \Re^{n\times 1}$ has a solution iff b is linearly dependent on the columns of A. That is, if

$$A = \left[\begin{pmatrix} \\ \underline{a_1} \\ \\ \end{pmatrix} \begin{pmatrix} \\ \underline{a_2} \\ \\ \end{pmatrix} \cdots \begin{pmatrix} \\ \underline{a_n} \\ \\ \end{pmatrix} \right], \text{ then } b = \sum_{r=1}^{n} c_r \underline{a_r},$$

for some $c_1, c_2 \cdots c_n$. The solution is unique $\Leftrightarrow$ $\det A \neq 0 \Leftrightarrow A$ is nonsingular $\Leftrightarrow \exists B \in \Re^{n\times n} \ni BA = AB = I_n$. Such a B is the inverse of A and is denoted A^{-1}.

ii. The matrix $Q \in \Re^{n\times n}$ is orthogonal if $Q^{-1} = Q^T$. (Note if Q is both symmetric and orthogonal then $Q^2 = Q\ Q^{-1} = I_n$.)

iii. Properties of inverses:

A. $(A^{-1})^{-1} = A$.

B. $(\lambda A)^{-1} = \frac{1}{\lambda} A^{-1}$ for $\lambda \in \Re$.

C. $(AB)^{-1} = B^{-1}A^{-1}$ for nonsingular A, B.

iv. Formally, for nonsingular A, we can 'solve' the equation

$$Ax = b$$

by multiplying both sides of the equation by A^{-1} to obtain,

$$\begin{aligned} A^{-1}(Ax) &= A^{-1}b \\ (A^{-1}A)x &= A^{-1}b \\ x &= A^{-1}b. \end{aligned}$$

This is useful from a theoretical (or notational) point of view but is not algorithmically useful.

v. An alternative technique is to 'solve' the equation by first factoring (decomposing) A into the product of two matrices

$$A = S\,T,$$

where S and T are $n \times n$ nonsingular matrices with special structure such that 'solving' the linear systems $Sx = b$, and $Tx = b$ are both 'easy'. (For example these matrices may be Triangular or Orthogonal.)
If we have such a decomposition and let $z = Tx$ we have,

$$\begin{aligned} Ax &= b \\ S\,Tx &= b \\ Sz &= b. \end{aligned}$$

Therefore to determine x we first solve the 'easy' problem $Sz = b$, and then solve the 'easy' problem $Tx = z$. The special cases we will consider are when S and T are either triangular (forward or Back substitution is used) or orthogonal (inverse is free).

vi. Classification of Numerical Methods for Linear Equations

- <u>Direct</u> methods yield the exact answer if exact arithmetic is used (the only error is that caused by FP arithmetic). These methods are effective for $n \leq\approx 1000$ and Gaussian elimination is the best known direct method.
- <u>Iterative</u> methods are based on an underlying iteration,

  $$\underline{x_i} = G(\underline{x_{i-1}}) \text{ for } i = 1, 2 \cdots,$$

  where G must be carefully chosen as a function of A and an initial guess supplied, x_0, to ensure $\underline{x_i} \to \underline{x}$. These methods are used for $n >> 1000$ when A has special structure. They avoid $O(n^2)$ storage and $O(n^2)$ floating point operations (FLOPS).

3. Calculus – Notation and Review (from MATA37H/MATA36H and MATB41H):
We will use standard notation from calculus and analysis. For example:

- $[a, b]$ is the closed interval, ($x \in R$, such that $a \leq x \leq b$).
- (a, b) is the open interval, ($x \in R$, such that $a < x < b$).
- $f^n(x) = \frac{d^n}{dx^n} f(x)$.
- $f \in C^n[a, b] \;\Rightarrow\; f$ is n times differentiable on $[a, b]$ and $f^n(x)$ is continuous on (a, b).
- $g_x(x, y) \equiv \frac{\partial}{\partial x} g(x, y)$, $g_y(x, y) \equiv \frac{\partial}{\partial y} g(x, y)$, $g_{xy}(x, y) \equiv \frac{\partial^2}{\partial x \partial y} g(x, y)$ etc.
- $g(h) = O(h^n)$ as $h \to 0 \Leftrightarrow \exists h_0 > 0 \; and \; K > 0 \ni |g(h)| < K h^n \; \forall \; 0 < h < h_0$.

Some Relevant Theorems from Calculus:

We will also frequently use the classical theorems from analysis such as:

Intermediate Value Theorem Let $f(x)$ be continuous on $[a, b]$. If $f(x_1) < \alpha < f(x_2)$ for some α and $x_1,\ x_2 \in [a, b]$, then $\alpha = f(\eta)$ for some $\eta \in [a, b]$.

Max-Min Theorem Let $f(x)$ be continuous on $[a, b]$. Then $f(x)$ assumes its maximum and minimum values on $[a, b]$. (That is, $\exists \underline{x} \; and \; \bar{x} \; \in [a, b] \ni \forall x \in [a, b]$, we have $f(\underline{x}) \leq f(x) \leq f(\bar{x})$.)

Mean Value Theorem for Integrals Let $g(x)$ be a non-negative (or non-positive) integrable function on $[a, b]$. If $f(x)$ is continuous on $[a, b]$ then

$$\int_a^b f(x) g(x) dx = f(\eta) \int_a^b g(x) dx,$$

for some $\eta \in [a, b]$.

Mean Value Theorem for Sums Let $f(x) \in C^1[a, b]$, let $x_1, x_2, \cdots, x_n$ be points in $[a, b]$ and let $w_1, w_2, \cdots, w_n$ be real numbers of one sign, then

$$\sum_{i=1}^{n} w_i f(x_i) = f(\eta) \sum_{i=1}^{n} w_i,$$

for some $\eta \in [a, b]$.

Rolle's Theorem Let $f(x) \in C^1[a, b]$. If $f(a) = f(b) = 0$ then $f'(\eta) = 0$ for some $\eta \in (a, b)$.

Mean Value Theorem for Derivatives If $f(x) \in C^1[a, b]$ then

$$\frac{f(b) - f(a)}{b - a} = f'(\eta),$$

for some $\eta \in (a, b)$.

Fundamental Theorem of Calculus If $f(x) \in C^1[a,b]$ then $\forall x \in [a,b]$ and any $c \in [a,b]$ we have

$$f(x) = f(c) + \int_c^x f'(s)ds.$$

Taylor's Theorem (with remainder) If $f(x) \in C^{n+1}[a,b]$ and c is any point in $[a,b]$, then for $x \in [a,b]$, we have

$$f(x) = f(c) + f'(c)(x-c) + f''(c)\frac{(x-c)^2}{2} \cdots + f^n(c)\frac{(x-c)^n}{n!} + R_{n+1}(x),$$

where $R_{n+1}(x) = \frac{1}{n!}\int_c^x (x-u)^n f^{n+1}(u)du$.

Note that Taylor's Theorem is particularly relevant to this course. We can observe that, since $(x-u)^n$ is of constant sign for $u \in [c,x]$ we can write

$$R_{n+1}(x) = \frac{1}{n!}\int_c^x (x-u)^n f^{n+1}(u)du = f^{n+1}(\eta)\frac{(x-c)^{n+1}}{(n+1)!},$$

for some $\eta \in [c,x]$.

We can also observe the first few terms of the Taylor Series provides an accurate approximation to $f(c+h)$ for small h since we have

$$f(c+h) = f(c) + hf'(c) + \cdots \frac{h^n}{n!}f^n(c) + \frac{h^{n+1}}{(n+1)!}f^{n+1}(\eta).$$

where the error term, $E(h)$ is $O(h^{n+1})$.

3 QR Decomposition and Applications

3.1 Solving Linear Systems with Gaussian Elimination

The linear system of equations,

$$Ax = b, \text{ where } A \text{ is } n \times n \text{ and } b \in R^n,$$

can be solved using Gaussian elimination with partial pivoting. We have seen that this is equivalent to determining the permutation matrix P and lower and upper triangular matrices, L and U so that $P\ A = L\ U$. One then solves the system, $A\ x = b$, by solving the equivalent system $PAx = LUx = Pb$.

To solve for x using this decomposition we first transform $b \to Pb$ and then solve,

$$LUx = Pb,$$

using standard forward substitution and back substitution. That is, first solve $Lz = Pb$ followed by $Ux = z$.

3.2 Error Analysis

In exact arithmetic we have no truncation error in implementing GE and, from the previous analysis we would have $LU = PA$, $Lz = Pb$ and $Ux = z$. When implemented in FP arithmetic we compute $\overline{L}\,\overline{U} \approx PA$ and corresponding $\overline{x}, \overline{z}$. We will now investigate the accuracy of the computed $\overline{x}$.

- It can be shown that if the Decomposition (DECOMP) stage and the Solve stage (SOLVE) of GE are implemented in a FP system with $n\mu < .01$ then the computed $\overline{x}$ of $\overline{U}x = \overline{z}$ is the exact solution of,

$$(A + E)\overline{x} = b,$$

  with $E = (e_{i\,j})$ satisfying

$$|e_{i\,j}| \;<\; 1.02(2n^2 + n)\rho|A| \;\max_{i,j} |L_{i\,j}|\mu,$$

  where $|A| = max_{i,j}|a_{i\,j}|$ and ρ, (the 'growth factor') is defined by

$$\rho = \frac{1}{|A|} \max_{i,j,r} |a_{i\,j}^{(r)}|.$$

  If 'partial pivoting' is used when implementing the $L\,U$ decomposition, then one can show $\max_{i,j} |L_{i\,j}| = 1$ and $\rho < 2^{n-1}$.

  Note that:

  1. The bound $\rho < 2^{n-1}$ is pessimistic for most problems.
  2. With this strategy the corresponding error bound reduces to,

$$|e_{i\,j}| \leq 1.02\rho|A|(2n^2 + n)\mu,$$

     where $\rho|A|$ can be monitored during the computation and is guaranteed to be bounded by a quantity that doesn't grow faster than 2^n.

  What about $|x - \overline{x}|$??

  [For a detailed, long but elementary proof of this result see Forsythe and Moler, Computer Solution of Linear Equations, Prentice Hall, pp. 87-108.]

- The true error, $(x - \overline{x})$ (due to FP arithmetic):
  Consider the following example in FP arithmetic:

$$.780x_1 + .563x_2 = .217$$

$$.913x_1 + .659x_2 = .254$$

  The true solution (exact arithmetic) is $x = (1, -1)^T$. Consider two approximate solutions generated in FP arithmetic by different methods, $\overline{x} = (.999, -1.001)^T$ and $\hat{x} = (.341, -.087)^T$. Which is the 'better' approximation?

First consider the corresponding <u>residuals</u> (the amount by which the approximations fail to satisfy the system),

$$\overline{r} \equiv A\overline{x} - b = (-.00136,\ -.00157)^T,$$

while

$$\hat{r} \equiv A\hat{x} - b = (-.000001,\ .000000)^T.$$

That is $\hat{x}$ is the exact solution of

$$Ax = b + \epsilon,$$

where $|\epsilon| < (10^{-5}, 10^{-5})^T$. This demonstrates that for general problems a small residual (which can be confirmed as it is computable) does not necessarily reflect a small error.

3.3 Matrix Norms and Condition Number

To quantify and investigate the potential 'size' of $\|x - \bar{x}\|$ we will need to use <u>matrix norms</u> and this will allow us to describe the sensitivity of this error measure to small changes in the problem.

1. Definitions:

   (a) For $x \in \Re^n$ consider two common vector norms,

   $$\|x\|_\infty \equiv \max_{i=1}^{n} |x_i| \text{ and } \|x\|_2 \equiv (x^T x)^{1/2}.$$

   (b) For $A \in \Re^{n \times n},\ A = (a_{i\,j})$, we define the <u>induced</u> or <u>subordinate</u> matrix norm (corresponding to any vector norm) to be, $\|A\| \equiv max_{\|v\|=1} \|Av\|$.

   For the above two examples of vector norms we then have:

   $$\begin{aligned} \|A\|_\infty &= \max_{i=1}^{n} [\sum_{j=1}^{n} |a_{i\,j}|], \\ \|A\|_2^2 &= \max_{\|x\|_2=1} (Ax)^T (Ax) \\ &= \max_{\|x\|_2=1} [x^T (A^T A) x]. \end{aligned}$$

   Properies of matrix norms:

   - $\|AB\| \leq \|A\| \|B\|$.
   - If $y = Ax$ we have $\|y\| \leq \|A\| \|x\|$.

2. Recall that for the computed approximation, $\bar{x}$, associated with GE with partial pivoting we have have the corresponding residual vector, $\bar{r} \equiv A\bar{x} - b$, and

$$(A+E)\bar{x} = b \;\Rightarrow\; \bar{r} = A\bar{x} - b = -E\bar{x}.$$

This implies from the properties of matrix norms, $\|\bar{r}\|_\infty \leq \|E\|_\infty \|\bar{x}\|_\infty$. But we also know from our previous analysis that $E = (e_{i\,j})$ satisfies,

$$|e_{i\,j}| < 1.02\rho(2n^2 + n)|A|\mu,$$

so

$$\|E\|_\infty = \max_{i=1}^{n}[\sum_{j=1}^{n} |e_{i\,j}|] \leq 1.02\rho(2n^3 + n^2)|A|\mu,$$

and

$$\|\bar{r}\|_\infty \leq \|E\|_\infty \|\bar{x}\|_\infty \leq 1.02\rho(2n^3 + n^2)|A|\|\bar{x}\|_\infty \mu.$$

Therefore, GE with partial pivoting for any system $Ax = b$ will generate a approximate solution, $\bar{x}$ with a guaranteed small residual. In fact since $|A|$, $\|\bar{x}\|_\infty$, and ρ are all known we have a precise bound on the size of the residual.

3. The true solution x need not be close to $\bar{x}$ since,

$$(\bar{x} - x) = \bar{x} - A^{-1}b = A^{-1}(A\bar{x} - b) = A^{-1}\bar{r},$$

and this implies

$$\begin{aligned} \|(\bar{x} - x)\|_\infty &\leq \|A^{-1}\|_\infty \|\bar{r}\|_\infty \\ &\leq \|A^{-1}\|_\infty \|A\|_\infty 1.02\rho(2n^3 + n^2)\mu\|\bar{x}\|_\infty. \end{aligned}$$

Note:

- In general we do not know how large $\|A^{-1}\|_\infty$ might be.
- If A is singular then A^{-1} is not defined so it is clear that if A is 'nearly singular' then $\|A^{-1}\|_\infty$ must be very large.

4. We define the Condition Number of A, wrt linear equations, to be.

$$cond(A) \equiv \|A\|_\infty \|A^{-1}\|_\infty.$$

Clearly $cond(A)$ is an indication of how far away the computed $\bar{x}$ might be from the true x.

When $cond(A)$ is large the problem is said to be Ill-Conditioned since a small change in the RHS vector, b, can cause a large change in the solution vector, x. Consider the previous Example where we have,

$$A = \begin{bmatrix} .780 & .563 \\ .913 & .659 \end{bmatrix}, \text{ and } A^{-1} = 10^6 \times \begin{bmatrix} .659 & -.563 \\ -.913 & .780 \end{bmatrix}.$$

In this case we have $cond(A) = \|A\|_\infty \|A^{-1}\|_\infty = 2.6 \times 10^6$ which indicates an ill-conditioned problem.

5. Although we have used the concept of the condition number to quantfy the errors arising in GE with partial pivoting, it is a more general concept and can be defined for any matrix norm. In particular it describes the inherent sensitivity of the exact solution to small changes in the data defining the problem. To see this consider x to be the exact solution of $Ax = b$ and x' the exact solution of $Ax' = b + \epsilon$. We then have,
$$x' = A^{-1}(b + \epsilon) = x + A^{-1}\epsilon = x + \delta,$$
where $\|\delta\|_\infty = \|A^{-1}\ \epsilon\|_\infty$. It is possible to choose the perturbation, ϵ so that the resulting δ satisfies
$$\|\delta\|_\infty = \|A^{-1}\|_\infty \|\epsilon\|_\infty,$$
and we see how small perturbation in b can result in a large change in the corresponding exact solution.

3.4 QR Decomposition - for Linear Systems

1. A Householder Reflection is an elementary matrix of the form,
$$Q = I - 2ww^T, \ \text{ where } w \in \Re^n \text{ satisfies } \ \|w\|_2 = 1.$$
(Recall that $\|w\|_2 = 1 \Leftrightarrow ww^T = 1$.)

Properties of Householder reflections:

(a) $Q^T = Q$ (symmetric) since $[I - 2ww^T]^T = [I^T - 2(ww^T)^T] = [I - 2(w^T)^T w^T] = [I - 2ww^T] = Q$.

(b) $Q^T Q = Q^2 = I$ since ,
$$\begin{aligned}[I - 2ww^T][I - 2ww^T] &= I - 4ww^T + 4ww^T ww^T \\ &= I - 4ww^T + 4w(w^T w)w^T \\ &= I - 4ww^T + 4ww^T \\ &= I.\end{aligned}$$

Therefore we have $Q^{-1} = Q = Q^T$.

(c) $\|Q\|_2 = 1$, since for a general matrix A we have,
$$\|A\|_2^2 = \max_{\|x\|_2 = 1} \{x^T(A^T A)x\},$$
and therefore, for a Householder reflection,
$$\|Q\|_2^2 = \max_{\|x\|_2 = 1} \{x^T(Q^T Q)x\} = \max_{\|x\|_2 = 1} \{x^T x\} = 1.$$

(d) if $y = Qx$ then $\|y\|_2 = \|x\|_2$ since,
$$\|y\|_2^2 = y^T y = (Qx)^T Qx = x^T(Q^T Q)x = x^T x = \|x\|_2^2.$$

2. We usually define Q in terms of an arbitrary vector $u \in \Re^n$ by,

$$Q = [I - 2\frac{uu^T}{\|u\|_2^2}],$$

Note that this corresponds to $w = u/\|u\|_2$ but it avoids computing a square root and the normalization of u.

Consider the affect of this transformation on a vector x, $y = Qx$,

$$y = [I - 2\frac{uu^T}{\|u\|_2^2}]x = x - 2\frac{u^Tx}{\|u\|_2^2}u = x + \gamma u.$$

That is, $u = (y - x)/\gamma$ and u is a multiple of $(y - x)$. This implies that for any y such that $\|y\|_2 = \|x\|_2$ we can map x onto y using

$$Q = [I - 2\frac{(y-x)(y-x)^T}{\|y-x\|_2^2}].$$

3. **An Example:**

Determine the value(s) of $t \in \Re$ such that there exists a Householder reflection, Q that maps $(7, 0, 1)^T \rightarrow (0, 5, t)^T$ and find the corresponding transformation(s).

To do this we first observe that since the 2-norm must be preserved, we must have $7^2 + 1^2 = 5^2 + t^2$ or $t = \pm 5$.

Consider the solution corresponding to $t = -5$,

$$u = y - x = \begin{bmatrix} 0 \\ 5 \\ -5 \\ -5 \end{bmatrix} - \begin{bmatrix} 7 \\ 0 \\ 1 \end{bmatrix} = \begin{bmatrix} -7 \\ 5 \\ -6 \end{bmatrix}.$$

We then have $\|u\|_2^2 = 110$ and $-2/\|u\|_2^2 = (-1/55)$. The corresponding $Q = I - 2\frac{uu^T}{\|u_2^2\|}$, is then,

$$Q = I - \frac{1}{55}\begin{bmatrix} 49 & -35 & 42 \\ -35 & 25 & -30 \\ 42 & -30 & 36 \end{bmatrix} = \begin{bmatrix} 6/55 & 35/55 & -42/55 \\ 35/55 & 30/55 & 30/55 \\ -42/55 & 30/55 & 19/55 \end{bmatrix}.$$

Exercise. Determine the corresponding Q for $t = 5$.

4. A Householder reflection can be used to transform a given vector $x = [x_1, x_2 \cdots x_r \cdots x_n]^T$ onto $[x_1, x_2 \cdots x_{r-1}, s, 0, 0 \cdots 0]^T = y^T$ where, to preserve the norm we must have,

$$s^2 = x_r^2 + x_{r+1}^2 \cdots x_n^2.$$

In this case the corresponding u satisfies,

$$u = y - x = \begin{bmatrix} 0 \\ 0 \\ \vdots \\ 0 \\ -x_r \pm s \\ -x_{r+1} \\ \vdots \\ -x_n \end{bmatrix},$$

where the sign of s is usually chosen to agree with the sign of $-x_r$ (no cancellation). With this choice of u, applying Q to any vector v becomes:

$$Qv = [I - 2\frac{uu^T}{\|u\|_2^2}]v = v - \left(\frac{2u^T v}{\|u\|_2^2}\right) u.$$

That is, we form the scalar $u^T v$ and then add a multiple of u to v. This transformation will leave the first $(r-1)$ entries of v unchanged (as it adds 'multiples' of u to v). It also has no affect on v if $u^T v = 0$ (in particular, if $v_{r+1} = v_{r+2} = \cdots = v_n = 0$).

Now consider factoring $A = \mathcal{Q}R$ (rather than $A = LU$), where R is upper triangular and $\mathcal{Q} = Q_1 Q_2 \cdots Q_{n-1}$, a product of Householder reflections. Note that,

$$\mathcal{Q}^{-1} = \mathcal{Q}^T = Q_{n-1}^T Q_{n-2}^T \cdots Q_1^T = Q_{n-1} Q_{n-2} \cdots Q_1 \neq \mathcal{Q}.$$

This factoring (or decomposition of A) is accomplished (analogous to LU) by setting $A_0 = A$ and choosing Q_1 to introduce zeros below the diagonal of the first column of $A_1 = Q_1 A_0$,

$$A_1 = \begin{bmatrix} s_1 & (Q_1 a_2^{(0)}) & (Q_1 a_3^{(0)}) & \cdots & (Q_1 a_n^{(0)}) \\ 0 & \times & \times & \cdots & \times \\ 0 & \times & \times & \cdots & \times \\ \vdots & \vdots & \vdots & \vdots & \vdots \\ 0 & \times & \times & \cdots & \times \end{bmatrix}.$$

From above we see that this can be done with Q_1 defined by u_1,

$$u_1 = \begin{bmatrix} -a_{1\,1}^{(0)} \pm s_1 \\ -a_{2\,1}^{(0)} \\ \vdots \\ -a_{n\,1}^{(0)} \end{bmatrix},$$

where $s_1^2 = (a_{1\,1}^{(0)})^2 + (a_{2\,1}^{(0)})^2 \cdots + (a_{n\,1}^{(0)})^2$.

In general at the r^{th} stage we have,

$$A_{r-1} = \begin{bmatrix} a_{1\,1}^{(r-1)} & a_{1\,2}^{(r-1)} & \cdots & a_{1\,r}^{(r-1)} & \cdots & a_{1\,n}^{(r-1)} \\ 0 & a_{2\,2}^{(r-1)} & \cdots & a_{2\,r}^{(r-1)} & \cdots & a_{2\,n}^{(r-1)} \\ 0 & 0 & \cdots & a_{3\,r}^{(r-1)} & \cdots & a_{3\,n}^{(r-1)} \\ \vdots & \vdots & \vdots & \vdots & \vdots & \vdots \\ 0 & 0 & \cdots & a_{r\,r}^{(r-1)} & \cdots & a_{r\,n}^{(r-1)} \\ 0 & 0 & \cdots & a_{r+1\,r}^{(r-1)} & \cdots & a_{r+1\,n}^{(r-1)} \\ \vdots & \vdots & \vdots & \vdots & \vdots & \vdots \\ 0 & 0 & \cdots & a_{n\,r}^{(r-1)} & \cdots & a_{n\,n}^{(r-1)} \end{bmatrix}.$$

We choose Q_r to map $a_r^{(r-1)} \to (a_{1\,r}^{(r-1)}, a_{2\,r}^{(r-1)} \cdots a_{r-1\,r}^{(r-1)}, s_r, 0 \cdots 0)^T$,

$$s_r^2 = (a_{r\,r}^{(r-1)})^2 + (a_{r+1\,r}^{(r-1)})^2 \cdots + (a_{n\,r}^{(r-1)})^2.$$

That is,

$$u_r = \begin{bmatrix} 0 \\ 0 \\ \vdots \\ 0 \\ -a_{r\,r}^{(r-1)} \pm s_r \\ -a_{r+1\,r}^{(r-1)} \\ \vdots \\ -a_{n\,r}^{(r-1)} \end{bmatrix}.$$

5. We then have, after $(n-1)$ stages,

$$\begin{aligned} \mathcal{Q}^T A &= \mathcal{Q}^T A_0 \\ &= [Q_1 Q_2 \cdots Q_{n-1}]^T A_0 \\ &= [Q_{n-1} Q_{n-2} \cdots Q_1] A_0 \\ &= (Q_{n-1}(\cdots(Q_2(\underbrace{Q_1 A_0}_{A_1}))) \cdots) \\ &= A_{n-1} \\ &\equiv R \end{aligned}$$

Since $\mathcal{Q}^T A = R$ and $\mathcal{Q}^T = \mathcal{Q}^{-1}$ we have, after multiplying both sides of this equation by $\mathcal{Q}$,

$$\mathcal{Q}R = \mathcal{Q}(\mathcal{Q}^T A) = (\mathcal{Q}\mathcal{Q}^T)A = A.$$

An alternative decomposition of A which is easy to work with and compute.

Exercise: Show that the operation count for this decomposition is twice that for the LU decomposition. That is, it is $2n^3/3 + O(n^2)$ flops.

3.5 Solving Linear Systems with QR

1. To solve the linear system $Ax = b$ using $\mathcal{Q}R$ we note that $\mathcal{Q}$ need not be explicitly computed – it need only be represented by retaining the vectors $u_1, u_2 \cdots u_{n-1}$ and the scalars $\|u_1\|_2^2, \|u_2\|_2^2, \cdots \|u_{n-1}\|_2^2$. We then observe that,

$$A = \mathcal{Q}R = [Q_1 Q_2 \cdots Q_{n-1} R],$$

allows us to 'solve' $Ax = b$ as

$$\mathcal{Q}Rx = b, \quad \text{or} \quad Rx = \mathcal{Q}^{-1}b.$$

But

$$\mathcal{Q}^{-1} = \mathcal{Q}^T = [Q_{n-1} Q_{n-2} \cdots Q_1]$$

and we have that,

$$Rx = (Q_{n-1}(Q_{n-2}(\cdots \underbrace{(Q_1 b)}_{z_1} \cdots)).$$

This suggests the following efficient algorithm:

```
-set $z = b$ ;
-for $j = 1, 2 \cdots (n-1)$ do
        -set $z = Q_j z$      (ie. 'solve' $Q_j z_j = z_{j-1}$)
-end
-solve the triangular system $Rx = z$ by back substitution.
```

Exercise:

Given that the Q_j are 'represented' by the vectors, u_j and the scalars $\|u_j\|_2^2$, determine the operation count for the above algorithm and compare it with the that for the standard LU algorithm. (Recall that computing $Q_j\ v$ for an arbitrary vector v can be done using fact that $Q_j v = v + \gamma u_j$, where $\gamma = -2\frac{u_j^T v}{\|u_j\|_2^2}$.)

2. Error bounds for the $\mathcal{Q}R$ algorithm.

One can show an analogous result to that we discussed for the LU algorithm (ie., for GE with partial pivoting). If the above $\mathcal{Q}R$ algorithm is implemented in floating point arithmetic, then the computed solution, $\bar{x}$, will satisfy,

$$(A + E)\bar{x} = b \ \text{ where } \ E = (e_{i\,j}),$$

and

$$|e_{i\,j}| \leq 1.02 \max_{r=0,1\cdots n-1} [\|A_r\|_2](2n^2 + n)\|\mathcal{Q}\|_2 \mu.$$

But $\|\mathcal{Q}\|_2 = 1$ and $\|A_r\|_2 \leq \|Q_r\|_2 \|A_{r-1}\|_2 = \|A_{r-1}\|_2$ for $r = 1, 2 \cdots (n-1)$, so

$$\max_{r=0,1\cdots n-1} [\|A_r\|_2] = \|A_0\|_2 = \|A\|_2,$$

and
$$\|e_{i\,j}\| \leq 1.02\|A\|_2(2n^2+2)\mu.$$
Note that this expression does not contain an extra 'growth factor' that was necessary when analysing the LU algorithm. Thus the QR algorithm is more stable than GE with PP since all the intermediate results (that arise in the computation) are bounded by a smaller value than is the case for GE.

3. An alternative procedure (or algorithm) to form the QR decomposition of A is based on the Gram Schmidt technique. It is equivalent in cost and a similar error expression can be derived.

3.6 Iterative Improvement and Conditioning:

As with any numerical method we would like the software to signal if the underlying problem is ill conditioned. Consider our example of an ill conditioned problem:
$$\begin{bmatrix} .780 & .563 \\ .913 & .659 \end{bmatrix}\begin{bmatrix} x_1 \\ x_2 \end{bmatrix} = \begin{bmatrix} .217 \\ .254 \end{bmatrix}.$$

The exact solution is $x = (1,-1)^T$. Two candidate approximate solutions (determined by some unspecified methods) are:
$$\tilde{x} = (.999, -.1001)^T,$$
and
$$\hat{x} = (.341, -.087)^T,$$
with corresponding residuals:
$$\tilde{r} = A\tilde{x} - b = (-.00136, -.00157)^T,$$
and
$$\hat{r} = A\hat{x} - b = (-.000001, .000000)^T.$$

The ill conditioning is reflected by the size of $cond(A)$ which is $\approx 2.6\times 10^6$. Can we get any indication of this ill conditioning as we solve the linear system with GE or QR?

In 3-digit, base 10, FP arithmetic we determine $\bar{L}\bar{U}$ by
$$\begin{aligned} \bar{U}=A_1=(M_1P_{12})A_0 &= M_1(P_{12}A_0) \\ &= M_1\begin{bmatrix} .913 & .659 \\ .780 & .563 \end{bmatrix} \\ &= \begin{bmatrix} 1.0 & 0.0 \\ -.854 & 1.0 \end{bmatrix}\begin{bmatrix} .913 & .659 \\ .780 & .563 \end{bmatrix} \\ &= \begin{bmatrix} .913 & .659 \\ 0.0 & .000214 \end{bmatrix}. \end{aligned}$$

We then have

$$\bar{L} = M_1^{-1} = \begin{bmatrix} 1.0 & 0.0 \\ .854 & 1.0 \end{bmatrix},$$

$$\bar{U} = \begin{bmatrix} .913 & .659 \\ 0.0 & .000214 \end{bmatrix}.$$

In FP arithmetic we can compute

$$\bar{L}\bar{U} = \begin{bmatrix} .913 & .659 \\ .780 & .563 \end{bmatrix},$$

with no indication of trouble. In the SOLVE stage of GE we determine $\bar{x}$ in FP arithmetic using forward and back substitution.

Solving $\bar{L}z = P_{12}b$ we obtain, $z_1 = .254$ and

$$z_2 = (.217 \ominus .854 \otimes .254) = 0.00$$

and solving $\bar{U}x = z$ we obtain, $\bar{x}_2 = z_2 = 0.00$ and

$$\bar{x}_1 = (.254 \ominus .659 \otimes 0.00) / \oslash .913 = .278.$$

Therefore $\bar{x} = (.278,\ 0.00)^T$ with a corresponding residual,

$$\bar{r} = A\bar{x} - b = \begin{bmatrix} -.000160 \\ -.000186 \end{bmatrix}.$$

We will now consider an approach to improve the accuracy of the computed solution as well as a technique for estimating the value of $cond(A)$.

1. Consider the use of the QR method in floating point arithmetic to define $\bar{x}^{(1)}$ satisfying,

   $$(A + E)\bar{x}^{(1)} = b, \ \ \bar{r}^{(1)} = A\bar{x}^{(1)} - b,$$

   with

   $$|e_{i\,j}| \leq 1.02\|A\|(2n^2 + n)\mu \leq C\|A\|\mu.$$

   To 'improve' $\bar{x}^{(1)}$ consider solving,

   $$Az^{(1)} = -\bar{r}^{(1)}.$$

   If $z^{(1)}$ was to be computed in exact arithmetic then,

   $$\begin{aligned} A(\bar{x}^{(1)} + z^{(1)}) &= A\bar{x}^{(1)} + Az^{(1)} \\ &= (\bar{r}^{(1)} + b) - \bar{r}^{(1)} \\ &= b \end{aligned}$$

Therefore to improve the accuracy of $\bar{x}^{(1)}$ we can compute $\bar{r}^{(1)}$ (in higher precision); solve $Az^{(1)} = -\bar{r}^{(1)}$; and use the resulting approximation, $\bar{z}^{(1)}$ to improve $\bar{x}^{(1)}$ (ie., $\bar{x}^{(2)} = \bar{x}^{(1)} + \bar{z}^{(1)}$.)

We can continue in this way until $\bar{x}^{(i)}$ converges to full accuracy. This is called Iterative Improvement.

Note:

(a) The residual, $\bar{r}^{(i)}$ should be computed in higher precision (although the first iteration will always yield a more accurate value for $\bar{x}^{(2)}$ even if this is not the case).

(b) Solving $Az^{(i)} = -\bar{r}^{(i)}$ is inexpensive since the QR factors of A have been computed and stored.

(c) Extra storage is required since, in order to compute the residual with enough accuracy, we need to retain A as well as the L and U.

2. A computable estimate of $cond(A) = \|A\|\|A^{-1}\|$.

Let $\epsilon^{(1)} \equiv (\bar{x}^{(1)} - x) = A^{-1}(A\bar{x}^{(1)} - b) = A^{-1}\bar{r}^{(1)}$. Taking norms we have,

$$\|\epsilon^{(1)}\| \leq \|A^{-1}\|\|\bar{r}^{(1)}\|.$$

In t-digit, base β floating point arithmetic, $\mu \approx \beta^{-t}$, and $\|\bar{r}^{(1)}\| \approx C\|A\|\|\bar{x}^{(1)}\|\beta^{-t}$, and we have,

$$\|\epsilon^{(1)}\| \approx C\|A^{-1}\|\|A\|\|\bar{x}^{(1)}\|\beta^{-t} = C cond(A)\|\bar{x}^{(1)}\|\beta^{-t}.$$

Let $cond(A) \approx \beta^p$ (we only want an order of magnitude estimate). Then, $\|\epsilon^{(1)}\| \approx C\|\bar{x}^{(1)}\|\beta^{p-t}$. Consider two situations:

(a) If $p \geq t$ the problem is badly conditioned and $\|\epsilon^{(1)}\|$ will be larger than $\|\bar{x}^{(1)}\|$ and we are in trouble. (In this case a large value of $\|\bar{z}^{(1)}\|/\|\bar{x}^{(1)}\|$ will signal trouble.)

(b) If $q = t - p > 0$ we see that the relative error in $\bar{x}^{(1)}$ satisfies:

$$\|\epsilon^{(1)}\|/\|\bar{x}^{(1)}\| \approx C\beta^{-q}.$$

This implies that approximately q digits of $\bar{x}^{(1)}$ are correct.

Now in similarly solving $Az^{(1)} = -\bar{r}^{(1)}$ we have $\bar{z}^{(1)}$ satisfies (agrees with $\epsilon^{(1)}$ to q digits):

$$\frac{\|\bar{z}^{(1)}\|}{\|\bar{x}^{(1)}\|} \approx \frac{\|\bar{\epsilon}^{(1)}\|}{\|\bar{x}^{(1)}\|} \approx \beta^{-q} = \beta^{p-t}, \text{ with } \mu \approx \beta^{-t}.$$

Therefore an estimate of $cond(A)$ is given by,

$$cond(A) \approx \beta^p \approx \beta^t \frac{\|\bar{z}^{(1)}\|}{\|\bar{x}^{(1)}\|} = (1/\mu)\frac{\|\bar{z}^{(1)}\|}{\|\bar{x}^{(1)}\|}.$$

3.7 The Linear Least Squares Problem

In many applications the linear systems of equations that arise do not have the same number of equations as unknowns. In such situations, we can 'solve' the equations in a least squares sense. the basic problem and solution technique are presented in this section.

1. Given m linear equations in n unknowns,

$$Ax \approx b, \quad A \in \Re^{m\times n}, \quad x \in \Re^n, \quad b \in \Re^m.$$

The problem is to determine the vector x that minimises,

$$\|Ax - b\|_2$$

Note that other norms ($\|x\|_\infty$, or $\|x\|_1$ for example) are not differentiable and the corresponding minimisation problem is difficult to analyse and the algorithms more complicated.

A Generic Example: – Data Fitting

Let the 'unknown' vector be $c \in \Re^n$, the coefficients defining a polynomial $p_n(s)$ of degree $< n$,

$$p_n(s) = c_1 + c_2 s \cdots c_n s^{n-1}.$$

Assume we wish to approximate the function $f(s)$ by $p_n(s)$ and we 'know' that $f(s_i) \equiv f_i$ for $i = 1, 2 \cdots m$. Let $A \in \Re^{m\times n}$ be defined by

$$a_{i\,j} = s_i^{j-1}, \; i = 1, 2 \cdots m, \;\; j = 1, 2 \cdots n.$$

Then it is easy to see that if,

$$s = \begin{bmatrix} s_1 \\ s_2 \\ \vdots \\ s_m \end{bmatrix}, \quad f = \begin{bmatrix} f_1 \\ f_2 \\ \vdots \\ f_m \end{bmatrix},$$

then

$$Ac = \begin{bmatrix} p_n(s_1) \\ p_n(s_2) \\ \vdots \\ p_n(s_m) \end{bmatrix},$$

and therefore our task is to determine c such that,

$$\|Ac - f\|_2$$

is minimised. This is an example of a <u>Discrete Least Squares</u> problem.

Note that:

(a) If the s_i are distinct then A is always full rank.

(b) A is a Vandermonde Matrix.

(c) If $m = n$, A is nonsingular but it can be badly conditioned (ie., cond(A) may be very large).

For the general LLSQ problem (including this data-fitting example), there will be three cases to consider:

$m = n$ Standard case with a unique solution if A has full rank. In this case the solution can be determined using either of the standard algorithms we have analysed and discussed.

$m > n$ Overdetermined case (where there are more equations than unkowns). Such problems can have full rank or they can be rank deficient.

$m < n$ Underdetermined case (fewer equations than unknowns). Such problems can have full rank or they can be rank deficient.

2. Overdetermined Problems/Normal Equations

$$\begin{bmatrix} \times & \times & \cdots & \times \\ \times & \times & \cdots & \times \\ \vdots & \vdots & \vdots & \vdots \\ \vdots & \vdots & \vdots & \vdots \\ \times & \times & \cdots & \times \end{bmatrix} \begin{bmatrix} x_1 \\ x_2 \\ \vdots \\ x_n \end{bmatrix} = \begin{bmatrix} b_1 \\ b_2 \\ \vdots \\ \vdots \\ b_m \end{bmatrix}.$$

Let

$$\begin{aligned} \Phi(x) &= \|Ax - b\|_2^2 \\ &= (Ax-b)^T(Ax-b) \equiv r^T(x)r(x) \\ &= \sum_{i=1}^{m} r_i^2(x). \end{aligned}$$

From standard results in calculus we know that $\Phi(x)$ is a minimum when,

$$\frac{\partial \Phi}{\partial x_j} = 0 \text{ for } \quad j = 1, 2, \cdots n.$$

But since $\Phi(x) = \sum_{i=1}^{m} r_i^2(x)$ we have,

$$\begin{aligned} \frac{\partial \Phi}{\partial x_j} &= \frac{\partial}{\partial x_j}\left[\sum_{i=1}^{m} r_i^2(x)\right] &(1)\\ &= \sum_{i=1}^{m} \frac{\partial}{\partial x_j}(r_i^2(x)), &(2)\\ &= 2\sum_{i=1}^{m} r_i(x)\frac{\partial r_i(x)}{\partial x_j}. &(3) \end{aligned}$$

But $r_i(x) = -b_i + a_{i\,1}x_1 + a_{i\,2}x_2 \cdots + a_{i\,n}x_n$ ($= -b_i +$ inner product of i^{th} row of A with x, which $= -b_i +$inner product of i^{th} column of A^T with x).

Therefore we have,

$$\begin{aligned} \frac{\partial r_i(x)}{\partial x_j} &= \frac{\partial}{\partial x_j}\left[Ax - b\right]_i, \\ &= (a_{i\,j}), \end{aligned}$$

for $i = 1, 2, \cdots m; j = 1, 2, \cdots n$.

It then follows that
$$\frac{\partial \Phi}{\partial x_j} = 2\sum_{i=1}^{m} r_i(x) a_{i\,j} = 2\left(A^T r\right)_j.$$

Therefore to achieve $\frac{\partial \Phi}{\partial x_j} = 0$ for $j = 1, 2, \cdots n$ we must have,

$$\left(A^T \underline{r}\right)_j = 0, \text{ for } j = 1, 2, \cdots m.$$

This is equivalent to asking that,

$$A^T r = 0 \text{ or } A^T (Ax - b) = 0.$$

Note that the matrix $A^T A$ is a square $n \times n$ nonsingular matrix and we can therefore solve our LLSQ problem with $m > n$ by solving the linear system,

$$\boxed{\boldsymbol{A^T A x = A^T b}}$$

These linear equations are called the Normal Equations.

We have shown that any solution to the LLSQ problem must be a solution to the Normal Equations. The converse is also true (exercise).

Note:

(a) The $(i, j)^{th}$ entry of $A^T A = a_i^T a_j = a_j^T a_i$.

(b) $A^T A$ is symmetric (since $(A^T A)^T = A^T (A^T)^T = A^T A$).

(c) The cost to determine $A^T A$ is $[n + (n^2 - n)/2]m = \frac{mn^2+mn}{2}$ flops.

It can be shown that, once $A^T A$ has been formed, solving $A^T Ax = b$ can be accomplished in $\frac{n^3}{6} + O(n^2)$ flops (for nonsymmetric matrices it would be $\frac{n^3}{3} + O(n^2)$).

3. QR based Algorithms for LLSQs

We will introduce a QR based algorithm that doesn't require the explicit computation of $A^T A$.

Consider forming the $\mathcal{QR}$ factorization (or Schur decomposition) of the $m \times n$ matrix A,

$$A = \mathcal{QR} = (Q_1 Q_2 \cdots Q_n)\mathcal{R},$$

where $\mathcal{Q}$ is an orthogonal matrix and $\mathcal{R}$ is an upper triangular matrix. This is a standard factorization in numerical linear algebra and is usually accomplished using a sequence of Householder reflections.

That is, we will determine $\mathcal{Q}$ as a product of n Householder reflections:

$$\mathcal{Q}A = \mathcal{R} \Leftrightarrow Q_n^T(Q_{n-1}^T \cdots Q_1^T A)) \cdots) = \mathcal{R},$$

where each $Q_i = Q_i^T$ is an $m \times m$ Householder reflection and $\mathcal{R}$ is an $m \times n$ upper triangular matrix,

$$\mathcal{R} = \begin{bmatrix} \times & \times & \cdots & \times \\ 0 & \times & \cdots & \times \\ 0 & 0 & \cdots & \times \\ \vdots & \vdots & \vdots & \times \\ 0 & 0 & \cdots & 0 \\ \vdots & \vdots & \vdots & \vdots \\ 0 & 0 & \cdots & 0 \end{bmatrix} \equiv \begin{bmatrix} R \\ 0 \end{bmatrix},$$

and R is a square $n \times n$ upper triangular matrix.

With such a factorization of A we have,

$$A^T A = (\mathcal{QR})^T \mathcal{QR} = \mathcal{R}^T \mathcal{Q}^T \mathcal{QR} = \mathcal{R}^T \mathcal{R},$$

where

$$\mathcal{R}^T\mathcal{R} = \begin{bmatrix} \times & 0 & 0 & \cdots & 0 \\ \times & \times & 0 & \cdots & 0 \\ \vdots & \vdots & \vdots & \cdots & 0 \\ \times & \times & \times & \cdots & 0 \end{bmatrix} \begin{bmatrix} \times & \times & \cdots & \times \\ 0 & \times & \cdots & \times \\ 0 & 0 & \cdots & \times \\ \vdots & \vdots & \cdots & \times \\ \vdots & \vdots & \vdots & \vdots \\ 0 & 0 & \cdots & 0 \end{bmatrix} = \begin{bmatrix} \times & \times & \cdots & \times \\ \times & \times & \cdots & \times \\ \vdots & \vdots & \cdots & \vdots \\ \times & \times & \cdots & \times \end{bmatrix}$$

$$= \begin{bmatrix} R^T & 0 \end{bmatrix} \begin{bmatrix} R \\ 0 \end{bmatrix} = R^T R.$$

Now solving the Normal Equations to determine x can be done by solving the equivalent linear system,

$$R^T R x = A^T b.$$

This requires the computation of $A^T b$ (at a cost of nm flops) and two triangular linear systems (at a cost of n^2 flops). The cost of determining the $\mathcal{QR}$ factorization of A is $n^2m + O(nm)$ and therefore the total cost of this algorithm to determine x is $n^2m + O(nm)$ flops.

Note that in most applications m is much larger than n. With this approach we have computed the LU (or Cholesky) decomposition of A^TA $(= R^TR)$ <u>without</u> forming A^TA.

4. (Optional) Underdetermined Problems
   In the case $m < n$ we do not have enough constraints to uniquely determine x from $Ax = b$,

$$\begin{bmatrix} \times & \times & \cdots & \times \\ \times & \times & \cdots & \times \\ \vdots & \vdots & \vdots & \vdots \\ \times & \times & \cdots & \times \end{bmatrix} \begin{bmatrix} x_1 \\ x_2 \\ \vdots \\ \vdots \\ x_n \end{bmatrix} = \begin{bmatrix} b_1 \\ b_2 \\ \vdots \\ b_m \end{bmatrix}.$$

   Before discussing algorithms for this case we will first review some key results and definitions from Linear Algebra:

   - Mathematical Preliminaries:
     (a) For an $m \times n$ matrix, A, the <u>null space</u> of A, $\mathcal{N}(A)$, is the <u>subspace</u> of vectors, $z \in \Re^n$ such that $Az = 0$.
     (b) If $rank(A) = m$ (ie. full rank), the <u>dimension</u> of the null space of $A = n - m$. That is, there exists $n - m$ linearly independent vectors in $\Re^n$ that span all of $\mathcal{N}(A)$.
     (c) The <u>range</u> of A, $\mathcal{R}(A)$, is the subspace of $\Re^m$ spanned by the <u>columns</u> of A (or the rows of A^T).
   - Properties:
     (a) If $y \in \Re^n$ is <u>any</u> solution of $Ax = b$ and if $z \in \mathcal{N}(A)$ then $y + z$ is also a solution of $Ax = b$. This follows since $A(y + z) = Ay + Az = b + 0 = b$.
     (b) If y_1 and y_2 are two solutions of $Ax = b$ then $y_1 - y_2 \in \mathcal{N}(A)$. This follows since $A(y_1 - y_2) = Ay_1 - Ay_2 = b - b = 0$.
   - To make the solution to an underdetermined problem well posed we usually ask for the particular solution of smallest $\|\cdot\|_2$. That is, find x such that $Ax = b$ and, for any other solution $y \in \Re^n$, $\|x\|_2 \le \|y\|_2$. Such a solution is unique.
   - **Theorem**: If w is a solution of $Ax = b$ and $w^Tz = 0$ for all $z \in \mathcal{N}(A)$ (ie. w is $\perp$ to $\mathcal{N}(A)$) then w is the unique solution of smallest $\|\cdot\|_2$.
     Proof: Let u be any solution of $Ax = b$.
     RTP: $\|w\|_2 < \|u\|_2$ unless $u = w$

We can write u as $u = w + (u - w)$ where $(u - w) \in \mathcal{N}(A)$,

$$\begin{aligned}
\|u\|_2^2 &= [w + (u-w)]^T[w + (u-w)] \\
&= w^T w + w^T(u-w) + (u-w)^T w + (u-w)^T(u-w) \\
&= \|w\|_2^2 + 2w^T(u-w) + \|u-w\|_2^2 \\
&= \|w\|_2^2 + \|u-w\|_2^2 \\
&> \|w\|_2^2 \text{ for } u \neq w.
\end{aligned}$$

- The rows of A are orthogonal to $\mathcal{N}(A)$.
  Proof: We can write A in terms of its rows as:
  $$A = \begin{bmatrix} r_1^T \\ r_2^T \\ \vdots \\ r_m \end{bmatrix},$$
  where the rows of A are also the columns of A^T,
  $$A^T = \begin{bmatrix} r_1 & r_2 & \cdots & r_m \end{bmatrix}.$$
  Now the j^{th} row of $A = j^{th}$ column of $A^T = A^T e_j$, where $e_j \in \Re^m$ is defined by,
  $$e_j^T \equiv (\underbrace{0, 0 \cdots 0}_{j}, 1, 0 \cdots 0).$$
  Now for any $z \in \mathcal{N}(A)$ we have,
  $$z^T r_j = z^T(A^T e_j) = (z^T A^T)e_j = (Az)^T e_j = \underline{0}^T e_j = 0.$$
- For the full rank case we know that the rows of A span an m-dimensional subspace of $\Re^n \equiv$ all of $[\mathcal{N}(A)]^\perp$. We can then view the subspace of all $n - vectors$ as a direct sum of the two subspaces, the null space of A and the orthogonal complement of the null space of A. That is,
  $$\Re^n \equiv \mathcal{N}(A) \oplus [\mathcal{N}(A)]^\perp.$$
  Therefore any vector $w \in [\mathcal{N}(A)]^\perp$ can be written as,
  $$\begin{aligned}
  w &= \sum_{i=1}^m t_i r_i, \\
  &= A^T \begin{bmatrix} t_1 \\ t_2 \\ \vdots \\ t_m \end{bmatrix}, \\
  &= A^T \underline{t},
  \end{aligned}$$

where $\underline{t} \in \Re^m$.

But we have seen that if $w \in \mathcal{N}(A)^\perp$ and $Aw = b$, then w is the unique solution of minimum $\|\cdot\|_2$. Now $w \in \mathcal{N}(A)^\perp$ implies (from above),

$$w = A^T\underline{t}, \text{ for some } \underline{t} \in \Re^m.$$

Therefore the solution we are seeking satisfies,

$$Aw = AA^T\underline{t} = b,$$

and a square $m \times m$ nonsingular system of linear equations determines $\underline{t} \in \Re^m$. Once $\underline{t}$ is known we can recover $w = A^T\underline{t}$ at a cost of $O(m \times n)$ flops.

The total cost of this full-rank underdetermined algorithm is $[m^2n/2+m^3/6+m^2] + O(mn)$ flops. (We must form AA^T, find the Cholesky factorization of AA^T, and solve two triangular systems.)

Summary: We have shown that the Normal Equations approach can be applied in both the overdetermined case ($m > n$):

$$x = (A^TA)^{-1}A^Tb, \text{ or } (A^TA)x = A^Tb,$$

and in the underdetermined case ($m < n$):

$$x = A^T(AA^T)^{-1}b, \text{ or } (AA^T)\underline{t} = b, \ x = A^T\underline{t}.$$

An alternative algorithm, based on a QR decomposition of A^T can also be developed for underdetermined problems. To see this, first determine,

$$A^T = \mathcal{QR} = [Q_1Q_2\cdots Q_m]\mathcal{R},$$

where each Q_i is an $n \times n$ Householder reflection,

$$\mathcal{R} = \begin{bmatrix} R \\ \underline{0} \end{bmatrix},$$

where R is an $m \times m$ upper triangular matrix and $\underline{0}$ is an $(m-n) \times n$ zero matrix.

With this decomposition we have,

$$A^T = [Q_1Q_2\cdots Q_m]\begin{bmatrix} R \\ \underline{0} \end{bmatrix},$$

and therefore,

$$A = [R^T : 0][Q_mQ_{m-1}\cdots Q_1].$$

The matrix AA^T can then be written as,

$$AA^T = [R^T : 0][Q_m Q_{m-1} \cdots Q_1][Q_1 Q_2 \cdots Q_m] \begin{bmatrix} R \\ .. \\ 0 \end{bmatrix},$$

$$= [R^T : 0] \begin{bmatrix} R \\ .. \\ 0 \end{bmatrix} = R^T R.$$

Therefore solving $AA^T\underline{t} = b$ is equivalent to solving the two triangular systems,

$$R^T R\underline{t} = b,$$

and setting $x = A^T\underline{t}$ completes the algorithm at a cost of one QR decomposition (of A^T), two triangular systems, and one matrix-vector multiply.(Note that a slightly different algorithm with the same cost is presented in the text.)

Exercise:

Carry out a detailed operation count for this algorithm and compare the results with the cost of explicitly forming the normal equation matrix, AA^T.

5. (Optional) Rank Defficient Problems:

   In the case that the LLSQ problem does not have full rank, the QR based algorithms we have discussed can still be applied with caution. Note that defficient rank implies that for overdetermined problems $rank(A) = rank(A^T) = rank(A^T A) < n$, while for underdetermined problems $rank(A) = rank(A^T) = rank(AA^T) < m$. In either case the $\mathcal{R}$ we obtain from the QR algorithm will have the same defficient rank as that of the LLSQ problem. That is, for the overdetermined case, $rank(\mathcal{R}) < n$ and for underdetermined problems $rank(\mathcal{R}) < m$. In exact arithmetic the rank of a triangular matrix is the number of non-zeros on the diagonal. In floating point arithmetic we don't expect to see an exact zero but a reliable indication of rank defficiency is to observe a large value for the ratio of the largest to smallest magnitudes of the diagonal entries. That is since the diagonal entries are the eigenvalues of a triangular matrix, a large value for the ratio of largest magnitude to smallest magnitude eigenvalue is an indication of <u>Numerical rank defficiency</u>.

   In problems where rank deffficiency is detected (using this idea) the algorithm can either exit with a warning or attempt to produce a solution to a nearby exactly rank defficient problem.

3.8 The Eigenvalue Problem

1. **A Review**

   [Note that this background review material is discussed in chapter 4 of the text.]

The Basic Problem:

For a given matrix $A \in \Re^{n \times n}$ determine $\lambda \in C$ and $x \in \Re^n, x \neq 0$ such that:

$$Ax = \lambda x.$$

λ is an <u>eigenvalue</u> and x is an <u>eigenvector</u> of A.

(a) An eigenvalue and corresponding eigenvector, (λ, x) is called an <u>eigenpair</u>.

(b) The <u>spectrum</u> of A is the set of all eigenvalues of A.

(c) To make the definition of a eigenvector precise we will often <u>normalize</u> the vector so it has $\|x\|_2 = 1$. (As we have defined it, any multiple of x is also an eigenvector.)

Note that the definition of eigenvalue is equivalent to finding λ and $x \neq 0$ such that,

$$(A - \lambda I)x = 0.$$

But the linear system $Bx = 0$ has a nontrivial solution iff B is singular. Therefore we have that λ is an eigenvalue of A iff $[A - \lambda I]$ is singular iff $det(A - \lambda I) = 0$.

Properties from Linear Algebra:

(a) For $A \in \Re^{n \times n}, det(A - \lambda I)$ is a polynomial of degree $\leq n$ in λ – it is called the <u>characteristic polynomial</u> of A. The roots of this polynomial are the eigenvalues of A.(This follows from the definition of determinate and from the above observation.)

(b) For a triangular matrix, $L = (l_{i\,j})$ or $U = (u_{i\,j})$ we have,

$$det(L) = \prod_{i=1}^{n} l_{i\,i}, \quad det(U) = \prod_{i=1}^{n} u_{i\,i}.$$

Therefore the eigenvalues of a triangular matrix are the diagonal entries of the matrix (since the determinate of a triangular matrix is the product of the diagonal entries and therefore $det(L - \lambda I) = \prod_{i=1}^{n}(l_{i\,i} - \lambda)$ has only the roots $l_{1\,1}, l_{2\,2} \cdots l_{n\,n}$.

(c) For an upper triangular matrix, U, an eigenvector corresponding to the eigenvalue, $u_{i\,i}$, can be determined by solving the linear system of equations,

$$[U - u_{i\,i}I]y = 0,$$

That is,

$$\begin{bmatrix} (u_{1\,1} - u_{i\,i}) & u_{1\,2} & \cdots & u_{1\,n} \\ 0 & (u_{2\,2} - u_{i\,i}) & \cdots & u_{2\,n} \\ \vdots & \vdots & \vdots & \vdots \\ 0 & 0 & 0 & (u_{n\,n} - u_{i\,i}) \end{bmatrix} \begin{bmatrix} y_1 \\ y_2 \\ \vdots \\ y_n \end{bmatrix} = \begin{bmatrix} 0 \\ 0 \\ \vdots \\ 0 \end{bmatrix}.$$

A solution to this system can be determined by a modified back substitution algorithm:

```
-set $y_n = y_{n-1} = \cdots y_{i+1} = 0$;
-set $y_i = 1$;
-for $j = (i-1), (i-2) \cdots 1$,
    $y_j = -[\sum_{r=j+1}^{i} u_{j\,r} y_r]/(u_{j\,j} - u_{i\,i})$;
-end
-normalize by setting $x = y/\|y\|_2$;
```

Note that this algorithm must be modified for multiple eigenvalues (we will consider this case later). A similar procedure works for lower triangular matrices (exercise).

We have shown that the eigenvalue problem is easy for all triangular matrices and the eigenvector problem is easy when the eigenvalues are distinct. We will now consider algorithms for the case of general matrices. One approach is to transform the general problem to an equivalent 'easy' problem (ie., an equivalent triangular eigenproblem). Before we consider this approach we will consider an alternative technique that is particularly appropriate if only the largest (or smallest) magnitude eigenvalue is desired.

2. The Power Method

Assume that $A \in \Re^{n\times n}$ has n eigenvalues $\lambda_1, \lambda_2 \cdots \lambda_n$, satisfying $|\lambda_1| \geq |\lambda_2| \cdots \geq |\lambda_n|$. Also assume that A has a complete set of normalized eigenvectors, $(v_1, v_2 \cdots v_n)$, (ie., A is non-defective). From linear algebra we have the result that these eigenvectors are linearly independent and any $x \in \Re^n$ can be expressed as,

$$x = \sum_{j=1}^{n} \alpha_j v_j.$$

If we are given an initial vector $x_0 \in \Re^n$ we define the normalized sequence $x_j, j = 1, 2, \cdots$ by,

$$\begin{aligned} y_j &= Ax_{j-1}, \\ x_j &= \frac{y_j}{\|y_j\|}, \end{aligned}$$

for $j = 1, 2 \cdots$.

It is straightforward to show that when $|\lambda_1| > |\lambda_2|$, we have,

$$x_j \rightarrow v_1,$$

and the rate of convergence is $O(\rho^j)$ where $\rho = \frac{|\lambda_2|}{|\lambda_1|}$.

Since $\|x_j\| = 1$ and $y_j \rightarrow \lambda_1 x_j$ we must also have,

$$\|y_j\| \rightarrow |\lambda_1|,$$

We then have that the eigenvalue can be determined from the observation that $\lambda_1 \in \Re$ (since $|\lambda_1| > |\lambda_2|$ and non-real eigenvalues must appear as conjugate pairs). This implies,

$$\lambda_1 = \pm \lim_{j \to \infty} \|y_j\|,$$

where the correct sign can be determined by comparing the first non-zero components of x_j and y_j.

The choice of norm used in the definion of x_j and y_j leads to different sequences but the term Power Method is used to refer to any method based on such a sequence. The text uses the l_∞ norm which is efficient but makes the discussion more difficult to follow. In many cases the l_2 norm is used for discussion but is slightly more expensive to implement since it requires one compute a sum of squares and a square root to determine $\|y_j\|$.

Exercise:

For the three norms, l_1, l_2 and l_∞ implement the power method in MATLAB and verify that for various choices of A and x_0 satisfying our assumptions, the resulting sequences are different but all three converge with the same rate of convergence.

3. Transformational Methods for General Matrices:

Recall that, for Linear Equations, triangular systems $Rx = b$ are easy and the LU and QR algorithms are based on transforming a given general problem, $Ax = b$, onto an equivalent triangular system,

$$Ux = \tilde{b}.$$

A similar approach will be developed for the eigenproblem.

For the general eigenvalue problem, we are given an $n \times n$ matrix, A, and we introduce a sequence of transformations that transform the eigenproblem for A onto equivalent eigenproblems for matrices A_i, where $A_i \to U$ (U upper triangular) as $i \to \infty$. This is an Iterative method. We will focus on developing an Iterative QR method, where $(n-1)$ Householder reflections are used to define the transformation on each iteration (defining A_i from A_{i-1}).

The Key Result from linear algebra that justifies this approach is the Theorem that similarity transformations preserve eigenvalues and allow us to recover eigenvectors. That is, Given any nonsingular matrix, M, the eigenproblem,

$$Ax = \lambda x,$$

has a solution (λ, x) iff the eigenproblem,

$$[MAM^{-1}]y = \lambda y,$$

has a solution (λ, y) where $y = Mx$.

Proof:

Let (λx) be a solution of $Ax = \lambda x$ and consider $B = MAM^{-1}, \;\; y = Mx$. We then have,

$$\begin{aligned} By &= [MAM^{-1}](Mx), \\ &= MA(M^{-1}M)x, \\ &= MAx, \\ &= M\lambda x, \\ &= \lambda(Mx), \\ &= \lambda y. \end{aligned}$$

To see the converse, let (λ, y) be an eigenpair for $B = MAM^{-1}$, with x the solution to $Mx = y$, (ie., $x = M^{-1}y$). If we now let $w = Ax = AM^{-1}y$ we observe (since (λ, y) is an eigenpair for MAM^{-1}),

$$\begin{aligned} Mw &= MAx, \\ &= MAM^{-1}y, \\ &= \lambda y, \\ &= \lambda Mx, \end{aligned}$$

or, after multiplying both sides by M^{-1},

$$Ax = \lambda x,$$

which, by definition means that (λ, x) is an eigenpair for A.

The 'trick' then is to choose the nonsingular sequence, $M_1, M_2 \cdots M_i$ such that,

$$\begin{aligned} A_0 &= A, \\ A_1 &= M_1 A_0 M_1^{-1}, \\ \vdots \;\; &\vdots \;\; \vdots \\ A_i &= M_i A_{i-1} M_i^{-1}, \end{aligned}$$

for $i = 1, 2 \cdots$, and $A_i \rightarrow$ a triangular matrix. One such choice leads to the QR Algorithm for eigenproblems.

4. QR Based Methods for Eigenproblems:

   This is a stable and efficient technique first introduced and analyzed by Rutishauser and Francis in the late 1950's. The basic idea is,

   (a) Factor $A_i = \mathcal{Q}_i \mathcal{R}_i$, where $\mathcal{Q}_i$ is orthogonal and $\mathcal{R}_i$ is upper triangular. Recall that the cost of this decomposition is $2/3n^3$ flops.

(b) Set $A_{i+1} = \mathcal{R}_i \mathcal{Q}_i$. This can be accomplished by forming $\mathcal{Q}_i^T \mathcal{R}_i^T$ as a product of $n-1$ Householder reflections and then taking the transpose to recover $\mathcal{R}_i \mathcal{Q}_i$ at a cost of $1/6n^3$ flops.

Note that we can make the following observations:

- A_{i+1} is similar to A_i since,

$$\mathcal{Q}_i^{-1} A_i \mathcal{Q}_i = \mathcal{Q}_i^T (\mathcal{Q}_i \mathcal{R}_i \mathcal{Q}_i) = (\mathcal{Q}_i^T \mathcal{Q}_i) \mathcal{R}_i \mathcal{Q}_i = A_{i+1}.$$

  To recover the eigenvector we must 'remember' each $\mathcal{Q}_i$ and each is a product of $n-1$ Householder reflections.

- Let $\overline{\mathcal{Q}_i} = \mathcal{Q}_1 \mathcal{Q}_2 \cdots \mathcal{Q}_i$ and $\overline{\mathcal{R}_i} = \mathcal{R}_i \mathcal{R}_{i-1} \cdots \mathcal{R}_1$ then we have,

$$\begin{aligned} A_{i+1} &= (\mathcal{Q}_1 \mathcal{Q}_2 \cdots \mathcal{Q}_i)^T A \mathcal{Q}_1 \mathcal{Q}_2 \cdots \mathcal{Q}_i, \\ &= \overline{\mathcal{Q}_i}^T A \overline{\mathcal{Q}_i}. \end{aligned}$$

  This result follows from the first observation using induction.

Rutishauser proved that with this iteration the A_i converge to an upper triangular matrix. For insight into why this is true consider $\overline{\mathcal{Q}}_i \overline{\mathcal{R}}_i = \overline{\mathcal{Q}}_{i-1} (\mathcal{Q}_i \mathcal{R}_i) \overline{\mathcal{R}}_{i-1} = \overline{\mathcal{Q}}_{i-1} (A_i) \overline{\mathcal{R}}_{i-1}$. From the second observation above,

$$\overline{\mathcal{Q}}_{i-1}^T A \overline{\mathcal{Q}}_{i-1} = A_i \quad \text{or} \quad \overline{\mathcal{Q}}_{i-1} A_i = A \overline{\mathcal{Q}}_{i-1}.$$

We then have,

$$\overline{\mathcal{Q}}_i \overline{\mathcal{R}}_i = \overline{\mathcal{Q}}_{i-1} A_i \overline{\mathcal{R}}_{i-1} = A \overline{\mathcal{Q}}_{i-1} \overline{\mathcal{R}}_{i-1},$$

which by induction implies the key observation,

$$\overline{\mathcal{Q}}_i \overline{\mathcal{R}}_i = A^i.$$

That is we have the QR decomposition of the i^{th} power of A. There is then a strong relationship then between the sequence A_i and the power method [section 9.2 of the text]. As the power method is known to converge, under some mild assumptions, it can be shown that this QR iteration will also converge.

5. Convergence and Implementation Issues:

Rate of Convergence: The rate of convergence depends on ratios $(\lambda_j/\lambda_r)^i$ for $j \neq r$, where i is the iteration number and λ_j and λ_r are the j^{th} and r^{th} eigenvalues of A. Thus we will observe slow convergence for complex eigenvalues since such eigenvalues appear as complex conjugate pairs and have equal magnitudes.

If the magnitudes of the largest eigenvalues are not well separated one can apply a 'shifted QR' to accelerate convergence.

The Shifted QR: To improve the rate of convergence consider the iteration:

$$(A_i - k_i I) = \mathcal{Q}_i \mathcal{R}_i,$$

where,

$$A_{i+1} = \mathcal{R}_i \mathcal{Q}_i + k_i I.$$

Note:

(a) A_{i+1} is similar to A_i since,

$$\begin{aligned} \mathcal{Q}_i^T A_i \mathcal{Q}_i &= \mathcal{Q}_i(\mathcal{Q}_i\mathcal{R}_i + k_i I)\mathcal{Q}_i \\ &= \mathcal{Q}_i^T \mathcal{Q}_i \mathcal{R}_i \mathcal{Q}_i + k_i \mathcal{Q}_i^T I \mathcal{Q}_i \\ &= \mathcal{R}_i \mathcal{Q}_i + k_i I \\ &= A_{i+1} \end{aligned}$$

and the rate of convergence will depend on ratios,

$$\frac{|\lambda_j - k_i|}{|\lambda_r - k_i|}.$$

(b) Choosing the 'shift' (k_i) as an approximation to the largest eigenvalue, $k_i \approx \lambda_n$, should ensure that the last row of $A_i \to [0, 0 \cdots 0, \lambda_n]$ rapidly (2 or 3 iterations).
The usual choice of k_i is the largest eigenvalue of the lower right 2×2 block of A_i. If k_i is complex a 'double shift' is applied (avoiding complex arithmetic).

(c) The cost of this version of the QR iteration is still $O(n^3)$ per iteration with $O(n)$ iterations. This results in an $O(n^4)$ method which is too expensive and which can be improved.

6. (Optional) A Two Stage QR Iteration:

A More efficient QR algorithm can be developed by viewing the convergence to an equivalent Triangular (or block triangular) matrix as consisting of two stages. The first stage consists of applying $n-2$ Householder-reflection, similarity transformations to reduce A to an equivalent upper Hessenberg matrix $A_{n-2} = H$.

The second stage applies the shifted QR iteration to H, $H_{i+1} = \mathcal{R}_i \mathcal{Q}_i + k_i I$, preserving the upper Hessenberg form for each H_{i+1}. The total cost is then $O(n^2)$ flops per iteration or $O(n^3)$ flops to find all the eigenvalues of H.

Stage 1: Reduction to Hessenberg Form

(a) On first step (of stage 1) choose the Householder reflection,

$$H_1 = [\, I - 2\frac{u_1 u_1^T}{\|u_1\|_2^2}\,]\ ,$$

so that the first column of $A = A_0$,

$$\underline{a}_1^{(0)} = \begin{bmatrix} a_{11}^{(0)} \\ a_{21}^{(0)} \\ \vdots \\ \vdots \\ a_{n1}^{(0)} \end{bmatrix}, \text{ is mapped onto } \underline{a}_1^{(1)} = \begin{bmatrix} a_{11}^{(0)} \\ \pm s \\ 0 \\ \vdots \\ 0 \end{bmatrix}.$$

Therefore we have that u_1 satisfies,

$$u_1 = (\underline{a}_1^{(1)} - \underline{a}_1^{(0)}) = \begin{bmatrix} 0 \\ -a_{21}^{(0)} \pm s \\ -a_{31}^{(0)} \\ \vdots \\ -a_{n1}^{(0)} \end{bmatrix},$$

where, to preserve the l_2 norm, we must have $s^2 = (a_{21}^{(0)})^2 + (a_{31}^{(0)})^2 \cdots + (a_{n1}^{(0)})^2)$.

(b) The first step of stage 1 is then defined by the associated similarity transformation,

$$\begin{aligned} A_0 &= A \\ A_1 &= H_1^T A_0 H_1 \, . \end{aligned}$$

With this definition we observe that the first column of A_1 is in upper Hessenberg form since $(H_1 A_0)$ clearly has its first column in this form and forming $(H_1 A_0)H_1$ doesn't change the first column. (That is, multiplying on the right by H_1 adds multiples of u_1^T to the rows of $(H_1 A_0)$ which doesn't change the first element of each row.)

Therefore we have,

$$A_1 = \begin{bmatrix} \times & \times & \times & \cdots & \times \\ \times & \times & \times & \cdots & \times \\ 0 & \times & \times & \cdots & \times \\ 0 & \times & \times & \cdots & \times \\ \vdots & \vdots & \vdots & \vdots & \vdots \\ 0 & \times & \times & \cdots & \times \end{bmatrix}.$$

(c) On the j^{th} step of stage 1 ($j = 1, 2 \cdots (n-2)$) we have

$$A_{j-1} = \begin{bmatrix} \times & \times & \cdots & \times & \times & \cdots & \times \\ \times & \times & \cdots & \times & \times & \cdots & \times \\ 0 & \times & \cdots & \times & \times & \cdots & \times \\ \vdots & \vdots & \vdots & \times & \times & \cdots & \times \\ \vdots & \vdots & \vdots & \times & \times & \cdots & \times \\ \vdots & \vdots & \vdots & 0 & \times & \cdots & \times \\ \vdots & \vdots & \vdots & 0 & \times & \cdots & \times \\ \vdots & \vdots & \vdots & \vdots & \vdots & \vdots & \vdots \\ 0 & 0 & \cdots & 0 & \times & \cdots & \times \end{bmatrix},$$

with the first $j-1$ columns in upper Hessenberg form (ie. zeros below the first subdiagonal). We then choose The Householder reflection, H_j so that,

$$H_j \underline{a}_j^{(j-1)} = \begin{bmatrix} a_{1j}^{(j-1)} \\ a_{2j}^{(j-1)} \\ \vdots \\ a_{jj}^{(j-1)} \\ \pm s \\ 0 \\ \vdots \\ 0 \end{bmatrix} = \underline{a}_j^{(j)}.$$

We then set

$$A_j = H_j A_{j-1} H_j,$$

which will have its first j columns in upper Hessenberg form. (This follows since multiplication on the right by H_j doesn't affect the first j columns of $(H_j A_{j-1})$.

To summarize, on step j:

$$u_j \equiv (\underline{a}_j^{(j)} - \underline{a}_j^{(j-1)}) = \begin{bmatrix} 0 \\ 0 \\ \vdots \\ -a_{j+1\,j}^{(j-1)} \pm s \\ -a_{j+2\,j}^{(j-1)} \\ \vdots \\ -a_{n\,j}^{(j-1)} \end{bmatrix}, \quad s^2 = \sum_{i=j+1}^{n} (a_{ij}^{(j-1)})^2.$$

With this definition of u_j we have

$$H_j = [I - \frac{2u_j u_j^T}{\|u\|_2^2}] = [I - vv^T],$$

where $v = \frac{\sqrt{2}}{\|u_j\|_2} u_j$. Therefore

$$\begin{aligned}
A_j &= H_j A_{j-1} H_j \\
&= [I - vv^T] A_{j-1} [I - vv^T] \\
&= A_{j-1} - A_{j-1} v v^T - v v^T A_{j-1} - v \underbrace{(v^T A_{j-1} v)}_{\alpha} v^T \\
&= A_{j-1} - A_{j-1} v v^T - v v^T A_{j-1} - \alpha v v^T,
\end{aligned}$$

with $\alpha \in \Re$.
Now if we let $x^T = \frac{\alpha}{2} v^T - v^T A_{j-1}$ (having its first $j-1$ elements equal to zero) and we let $y = \frac{\alpha}{2} v - A_{j-1} v$ then A_j can be determined from the expression,

$$A_j = A_{j-1} + v x^T + y v^T.$$

Note:

i. The first $j-1$ columns of vx^T and yv^T are in upper Hessenberg form which implies (from this expression) that the first $j-1$ columns of A_j are in upper Hessenberg form. Our choice of u_j ensures that the j^{th} column of A_j is in upper Hessenberg form.

ii. The cost of computing x^T and y is $2(n-j)^2$ flops, and the cost of computing,
$$a_{ir}^{(j)} = a_{ir}^{(j-1)} + v_i x_r + y_i v_r,$$
for $i = j, j+1 \cdots n$; $r = j+1, j+2 \cdots n$ is $2(n-j)^2$ flops.

iii. The total cost of Stage 1 is then $\frac{2}{3} n^3 + O(n^2)$ flops ($= 4 \sum_{j=1}^{n-2} (n-j)^2$).

Stage 2: The Shifted QR iteration applied to $H = A_{n-2}$.

- If H is upper Hessenberg then the QR iterates, $H_1, H_2 \cdots H_r \cdots$ will all also be upper Hessenberg. This follows since for an upper Hessenberg, H, the standard QR algorithm we have discussed produces a Q that is symmetric and tri-diagonal (and therefore upper Hessenberg). Furthermore since $H_1 = RQ$ is a product of an upper triangular matrix and an upper Hessenberg matrix it is upper Hessenberg.
- The QR iteration will converge (with real shifts) for any real matrix A_0 to an 'almost' upper triangular matrix, $\hat{U}$,

$$\hat{U} = \begin{bmatrix}
\times & \times & \times & \times & \times & \cdots & \times & \times & \times \\
\times & \times & \times & \times & \times & \cdots & \times & \times & \times \\
0 & 0 & \times & \times & \times & \cdots & \times & \times & \times \\
0 & 0 & \times & \times & \times & \cdots & \times & \times & \times \\
0 & 0 & 0 & 0 & \times & \cdots & \times & \times & \times \\
\vdots & \vdots & \vdots & \vdots & \vdots & \vdots & \vdots & \vdots & \vdots \\
0 & 0 & 0 & 0 & 0 & \cdots & 0 & \times & \times \\
0 & 0 & 0 & 0 & 0 & \cdots & 0 & \times & \times
\end{bmatrix}.$$

Note that $\hat{U}$ is an upper Hessenberg matrix with every second element of the subdiagonal zero. This implies (from linear algebra) that the eigenvalues of $\hat{U}$ are the union of the eigenvalues of all the diagonal blocks (2×2 and 1×1 blocks).

- Recall that the eigenvalues of a 2×2 matrix can be computed explicitly as the eigenvalues of

$$M = \begin{bmatrix} a & b \\ c & d \end{bmatrix}$$

are the roots of the polynomial $det(M - \lambda I) = 0$. That is, the roots of the equation $(a - \lambda)(d - \lambda) - cb = 0$. Using the quadradic formula to solve for the roots of this equation we obtain,

$$\begin{aligned} \lambda &= \frac{(a+d) \pm \sqrt{(a+d)^2 - 4(ad - cb)}}{2}, \\ &= \frac{(a+d) \pm \sqrt{(a-d)^2 - 4cb}}{2}. \end{aligned}$$

If $(a - d)^2 - 4cb < 0$, this gives a pair of complex conjugate eigenvalues. Otherwise the two eigenvalues are real.

4 Nonlinear Systems and Optimization

Recall that the basic problem of finding the solution of one equation in one unknown can be formally defined as:

Given $f(x)$, $f : \Re \rightarrow \Re$, find a real root (or zero), α, such that $f(\alpha) = 0$. We must be satisfied with an $\bar{\alpha}$ in our FP system, such that $|\bar{\alpha} - \alpha|$ is small.

The best known method to solve this class of problems is Newton's Method, which for real numbers x_0, a, b and $f(x) \in C^1[a, b]$ with $x_0 \in [a, b]$, is defined by,

```
-for $r = 1, 2 \cdots$ until satisfied do:
    $x_{r+1} = x_r - \frac{f(x_r)}{f'(x_r)}$
-end
```

Note that this iteration can also be written as,

$$f'(x_r)(x_{r+1} - x_r) = -f'(x_r).$$

Recall also that to analyze methods for solving this class of problems we need to introduce two important definitions:

- Definition: A sequence x_r converges to α iff $|x_r - \alpha| \rightarrow 0$ as $r \rightarrow \infty$.

- Definition: If $x_r \to \alpha$ and $\rho \geq 1$ is the smallest real number such that,

$$\lim_{r\to\infty} \frac{|x_{r+1}-\alpha|}{|x_r-\alpha|^\rho} \leq C \neq 0,$$

for some $C > 0$, then the convergence is order ρ. (Note that in this case, if x_r is accurate to k digits then x_{r+1} can be expected to be accurate to $\rho\ k$ digits. In particular, with order 2 convergence we should observe a doubling of the number of digits of accuracy on each iteration.)

Note that the order of convergence quantifies the rate or speed of convergence and can be very important in real computation.

The key properties for Newton's Method are:

- If $f(x) \in C^1[a,b]$ and $|x_0 - \alpha|$, is sufficiently small, then x_r will converge to α.
- If Newton's Method converges and $f'(\alpha) \neq 0$, then the order of convergence is 2.

4.1 Systems of Nonlinear Equations

- The Basic Problem: find $x \in \Re^n$ such that $F(x) = \underline{0}$ where,

$$\underline{0} = \begin{bmatrix} 0 \\ 0 \\ \vdots \\ 0 \end{bmatrix}, \ F(x) = \begin{bmatrix} f_1(x) \\ f_2(x) \\ \vdots \\ f_n(x) \end{bmatrix}.$$

Note that, in this section, n is the dimension of x (the number of unknowns) and r will be the iteration number.

- Analogous with the scalar case, the vector sequence, $x^{(0)}, x^{(1)} \cdots x^{(r)}$ converges to $\underline{\alpha} \in \Re^n$ with order p if there exists $c > 0$ such that

$$\lim_{r\to\infty} \frac{\|x^{(r+1)} - \underline{\alpha}\|}{\|x^{(r)} - \underline{\alpha}\|^p} = c.$$

The three most common values for p are $p = 1$ (linear convergence), $p = 2$ (quadratic convergence) and $p = 3$ (cubic convergence).

- Of the widely used methods for scalar problems ($n = 1$) only Newton's Method extends directly to higher dimensional problems ($n > 1$).

Newtons Method for Systems:
Given an initial guess, $x^{(0)} \in \Re^n$ we define the sequence $x^{(r)}$ for $r = 0, 1 \cdots$ by solving the linear system,

$$\frac{\partial F}{\partial x}|_{x=x^{(r)}} (x^{(r+1)} - x^{(r)}) = -F(x^{(r)},$$

or with $\Delta^r = x^{(r+1)} - x^{(r)}$, (the Newton Correction),

$$W_r \Delta^r = -F(x^{(r)}),$$

where $W_r = \frac{\partial F}{\partial x}|_{x=x^{(r)}}$ is the $n \times n$ matrix whose $(i\ j)^{th}$ element is $(\frac{\partial f_i}{\partial x_j})$.

Note that:

1. One can re-write this equation as,
$$x^{(r+1)} = x^{(r)} - W_r^{-1} F(x^{(r)}).$$
2. The matrix W_r must be recomputed and a new LU or QR decomposition computed on each iteration.
3. If $\frac{\partial F}{\partial x}|_{x=\underline{\alpha}}$ is nonsingular and the iterates converge, then the order of convergence is 2.

- Approximate versions of Newtons Method have been developed for systems of nonlinear equations that avoid the $O(n^3)$ flops per iteration or are more efficient to implement for other reasons.

  1. If we hold an approximation to W_r constant for several iterations we have a Modified Newton method. For example one can use $W_r = W_1$ for all r or one can re-evaluate W once every k iterations. In either case one looses quadratic convergence.
  2. If $\frac{\partial F}{\partial x}$ is difficult to compute we can use divided differences to define a Quasi Newton method,
  $$\frac{\partial f_i}{\partial x_j} \approx \frac{f_i(x + \delta e_j) - f_i(x)}{\delta},$$
  where $\delta \approx \sqrt{\mu}$, and e_j is the vector in $\Re^n$ whose components are all $= 0$ except for the j^{th} component which is equal to 1.
  3. We can approximate W_r by a 'nearby' matrix with special structure. This is called pre-conditioning and can involve approximating $W_r$by a diagonal, banded, or triangular matrix.
  4. If the approximation $W_r \approx I$ is used the method is called Functional iteration.

These approximate versions of Newtons method often work in special cases and can be readily analysed and justified only for these cases.

In the 'very' special case that $F(x)$ is linear, that is, $F(x) = Ax - b$, Newton method will converge in one iteration since,

$$\begin{aligned}
\Delta^r &= (x^{(r+1)} - x^{(r)}) \\
&= -W_r^{-1} F(x^{(r)}) \\
&= -A^{-1}(Ax^{(r)} - b) \\
&= A^{-1}b - x^{(r)} \\
&= \alpha - x^{(r)}
\end{aligned}$$

In summary, we have that the various versions of Newtons Method that are used in practice can be viewed as,

```
-guess $x^{(0)} = (x_1^{(0)}, x_2^{(0)} \cdots x_n^{(0)})^T$
-for $r = 0, 1 \cdots$ until satisfied do:
        -Solve $W_r \Delta^r = -F(x^{(r)}$
        $x^{(r+1)} = x^{(r)} + \Delta^r$
-end
```

where $W_r \approx \left(\frac{\partial F}{\partial x}\right)|_{x=x^{(r)}}$.

4.2 Optimization Problems

A special class of nonlinear systems are associated with optimization problems which arise in a wide variety of application areas.

- **The Basic Problem**:
  These problems have the form, Find $x \in \Re^n$ such that $f(x)$ is a minimum (or maximum) where
  $$f : \Re^n \to \Re.$$
  From calculus we know that $f(x)$ will have a minimum (or maximum) value at $x = (x_1, x_2 \cdots x_n)^T$ when,
  $$\nabla f(x) \equiv \begin{bmatrix} \frac{\partial f}{\partial x_1} \\ \frac{\partial f}{\partial x_2} \\ \vdots \\ \frac{\partial f}{\partial x_n} \end{bmatrix} = 0.$$

  Therefore an optimization problem can be 'reduced' to solving an associated system of nonlinear equations with $F(x) \equiv \nabla f(x)$. $\nabla f(x)$ has some very special structure which we can exploit when we apply a version of Newton's Method to solve the associated set of nonlinear equations.

  For a given $f(x), x^{(r)} \in \Re^n$ and an arbitrary vector $u \in \Re^n$ define $g(t),\ \ g : \Re \to \Re$ by,
  $$g(t) \equiv f(x^{(r)}\ +\ tu).$$
  It then follows that,
  $$\begin{aligned} \frac{dg}{dt} &\equiv g'(t) \\ &= (\frac{\partial f}{\partial x_1}u_1 + \frac{\partial f}{\partial x_2}u_2 \cdots + \frac{\partial f}{\partial x_n}u_n) \\ &= (\nabla f)^T u. \end{aligned}$$

This expression describes how f changes in the 'direction' u. In particular, if $(\nabla f)^T u < 0$ then f decreases in that direction and u is called a 'descent direction'. This observation leads to the method of steepest descent: where we choose $u = -\nabla f$ (to obtain $g'(t) = \nabla f^T u = -\|\nabla f\|_2^2$), and determine $\bar{t} \in \Re$ to minimise $g(t) = f(x^{(r)} - t\nabla f)$.

With this approach we have the following method for optimization problems:

```
-guess $x^{(0)} = (x_1^{(0)}, x_2^{(0)} \cdots x_n^{(0)})^T$
-for $r = 0, 1 \cdots$ until satisfied do:
    $u^r = \nabla f(x^{(r)})$  $(\equiv F(x^{(r)}))$
    -if $u^r \approx \underline{0}$ then signal convergence
    -else
        -find $\bar{t}$ such that $g(t) \equiv f(x^{(r)} - tu_r)$ is minimum
        -$x^{(r+1)} = x^{(r)} - \bar{t}u^r$
    -end
end
```

With this approach there is only a $1D$ line search on each iteration and any scalar nonlinear equation solver can be used. For example this line search can be done with Bisection, Newton or Secant. We will always observe a descent, that is

$$f(x^{(r+1)}) < f(x^{(r)}) \cdots < f(x^{(0)}).$$

One can prove that the sequence of iterates, $x^{(r)}$, will always converge but convergence may be slow.

- Just as we have shown how any Nonlinear equation method can be used to solve optimization problems, the converse is also true. That is we can interpret a system of nonlinear equations as a special case of an optimization problem. To see this, consider the nonlinear system $F(x) = 0$ and define $h : \Re^n \to \Re$ by,

$$h(x) \equiv \|F(x)\|_2^2 = \sum_{i=1}^{n} f_i^2(x).$$

Clearly,

$$F(\alpha) = \underline{0} \Leftrightarrow h(\alpha) \text{ is minimum.}$$

In this case,

$$\nabla h(x) = (\frac{\partial h}{\partial x_1}, \frac{\partial h}{\partial x_2} \cdots \frac{\partial h}{\partial x_n})^T.$$

where

$$\begin{aligned} \frac{\partial h}{\partial x_j} &= \frac{\partial}{\partial x_j} \sum_{i=1}^{n} f_i^2(x) \\ &= 2\sum_{i=1}^{n} \frac{\partial f_i}{\partial x_j} f_i \\ &= 2\left(W^T f\right)_j, \end{aligned}$$

where W is the Jacobian matrix defined by,

$$W \equiv \frac{\partial F}{\partial x} \text{ whose}(i\ j)^{th}\text{entry is}\frac{\partial f_i}{\partial x_j}.$$

Therefore we have that $\nabla h(x) = 2W^T F$ and this is the zero vector only when all components of F are zero.

5 Numerical Solution of ODEs

1. **Mathematical Preliminaries:**

   - Definition: A first-order ordinary differential equation is specified by:

     $$y' = f(x, y),$$

     over a finite interval $[a, b]$.

     Note that a solution of this ODE, $y(x)$, is a function of one variable (this is the reason for the term 'ordinary' as opposed to 'partial'). When the solution depends on more than one variable (ie a multivariate function) it is called a partial differential equation – PDE). The term first-order refers to the highest derivative that appears in the equation. We will consider higher-order equations later. For ODEs the variable x is called the independent variable while y (which depends on x) is called the dependent variable. 'Solving' the ODE is interpreted as determining a technique for expressing y as a function of x in some explicit way.

   - A function $\Phi(x)$ is a solution of this ODE if $\Phi(x) \in C^1[a, b]$ and $\forall x \in [a, b]$ we have $\Phi^{'}(x) = f(x, \Phi(x))$. (Note that this condition is often easy to check or verify).

     –An Example:
     $y' = \lambda y$, has solutions $\Phi(x) = c\ e^{\lambda x}$ for any constant c. In particular this ODE does not have a unique solution but rather a whole family of solutions (characterized by the parameter c).

   - To determine a unique mathematical solution we must add an additional constraint as part of the problem specification. This can be done in many ways. The most common is to prescribe the value of the solution at the initial point of the interval. That is we specify,

     $$y(a) = y_0.$$

     –Definition: An ODE together with the initial conditions specifies an initial value problem for an ordinary differential equation (IVP for an ODE).

- Before we can attempt to approximate a solution to an IVP we must consider some essential mathematical questions:
  (a) Does a solution exist?
  (b) If a solution exists, is it unique?
  (c) Can the problem be solved analytically? (If so, is it worth it?)

- Definition: The function $f(x,y)$ (a function of two variables that defines the ODE) satisfies a Lipschitz condition in y (ie, wrt its second argument) if $\exists L > 0$ such that $\forall x \in [a,b]$ and $\forall\, u, v$ we have

$$|f(x,u) - f(x,v)| \le L|u - v|.$$

  In particular, if $f(x,y)$ has a continuous partial derivative with respect to y and this derivative is bounded for all y, then f satisfies a Lipschitz condition in y since,

$$|f(x,u) - f(x,v)| = |\frac{\partial f}{\partial y}(x,\eta)|\;|u - v|,$$

  for some η between u and v.

- Theorem:
  Let $f(x,y)$ be continuous for $x \in [a,b]$ and $\forall y$ and satisfy a Lipschitz condition in y, then for any initial condition y_0 the IVP,

$$y' = f(x,y), \;\; y(a) = y_0, \;\; \text{over } [a,b],$$

  has a unique solution, $y(x)$ defined for all $x \in [a,b]$.

- Extension to systems of equations:
  Often one must deal with a system of n ‘unknown’ dependent variables of the form:

$$\begin{aligned} y_1' &= f_1(x, y_1, y_2, \cdots y_n), \\ y_2' &= f_2(x, y_1, y_2, \cdots y_n), \\ \vdots\; &\;\vdots\; \vdots \\ y_n' &= f_n(x, y_1, y_2, \cdots y_n), \end{aligned}$$

  with initial conditions all specified at the same point,

$$\begin{aligned} y_1(a) &= c_1, \\ y_2(a) &= c_2, \\ \vdots\; &\;\vdots\; \vdots \\ y_n(a) &= c_n, \end{aligned}$$

In vector notation, this system of IVPs in ODEs can be written

$$Y' = F(x, Y), \;\; Y(a) = Y_0,$$

where $Y(x) = [y_1(x), y_2(x), \cdots y_n(x)]^T$, $Y_0 = [c_1, c_2, \cdots c_n]^T$ and $F(x, Y)$ is a vector-valued function,

$$F(x, Y) = \begin{bmatrix} f_1(x, Y) \\ f_2(x, Y) \\ \vdots \\ f_n(x, Y) \end{bmatrix}.$$

The theory and the investigation of numerical methods that we present will be the same for systems as for scalar IVPs. In particular, the main mathematical Theorem quoted above holds for systems.

–Examples of systems:

(a) From Biology:
A predator-prey relationship can be modeled by the IVP:

$$y_1' = y_1 - 0.1 y_1 y_2 + 0.02x$$

$$y_2' = -y_2 + 0.02 y_1 y_2 + 0.008x$$

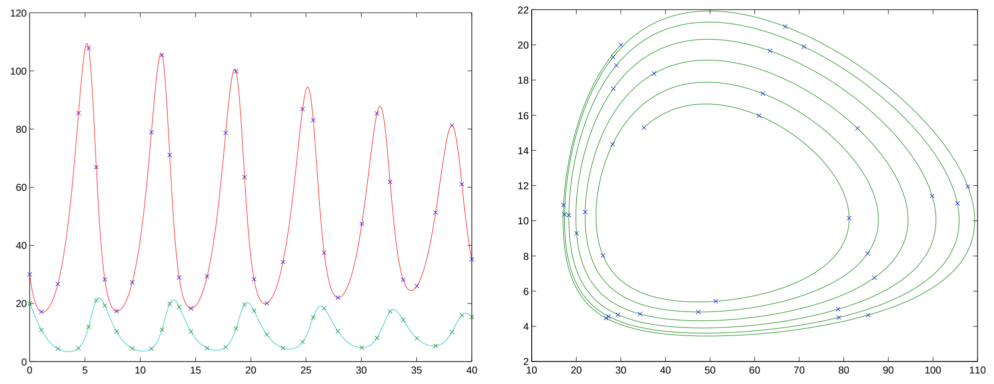
with

$$y_1(0) = 30, \;\; y_2(0) = 20.$$

Here $y_1(x)$ represents the 'prey' population at time x and $y_2(x)$ represents the 'predator' population at time x. The solution can then be visualized as a standard x/y solution plot or by a 'phase plane' plot. Figure 1 illustrates the solution to this system. We know that for different initial conditions solutions to this problem exhibit oscillatory behaviour as x increases.
A biologist may be interested in whether the solutions to this equation are 'almost periodic' (in the sense that the difference between successive maximum is constant) and whether the local maxima approach a steady state exponentially. (See Figure 2).

(b) From Chemistry:
The chemical reaction involving the combination of two chemicals C_1 and C_2, to yield a product C_3 is represented (in chemistry) by the mechanism (or notation)

$$C_1 + C_2 \;\underset{K_1}{\overset{K_2}{\rightleftharpoons}}\; C_3 \;.$$



(a) Solution Plot (b) Phase Plane Plot

Figure 1: Solutions to the predator prey problem for x in [0, 20]

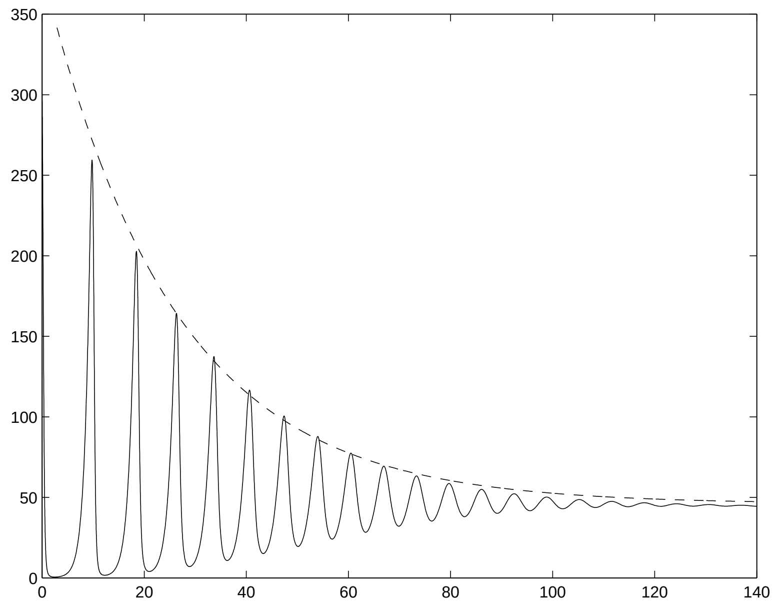


Figure 2: Typical behaviour of prey or predator population and decay to steady state

We can model this chemical reaction with the system of 3 ODEs, where $y_1(x) = [C_1]$ the concentration of the chemical C_1 at time x, $y_2(x) = [C_2]$ and $y_3(x) = [C_3]$. The resulting system of IVPs whose solution for $x \in [a, b]$ describes the change in concentrations over time as the reaction takes place (possibly approaching a steady state) is,

$$\begin{aligned} y_1' &= K_1 y_3 - K_2 y_1 y_2, \\ y_2' &= K_1 y_3 - K_2 y_1 y_2, \\ y_3' &= K_2 y_1 y_2 - K_1 y_3. \end{aligned}$$

- Extension to Second (and higher) order ODEs:

  Often physical or biological systems are best described by second or higher-order ODEs. That is, second or higher order derivatives appear in the mathematical model of the system. For example, from physics we know that Newtons laws of motion describe trajectory or gravitational problems in terms of relationships between velocities, accelerations and positions. These can often be described as IVPs where the ODE has the form $y''(x) = f(x, y)$ or $y''(x) = f(x, y, y')$.

  (a) A Second-order ODE can be reduced to an equivalent system of first-order ODEs as follows: With $y^{''} = f(x, y, y')$ we let $Z(x)$ be defined by,

  $$Z(x) = [z_1(x), z_2(x)]^T,$$

  where $z_1(x) = y(x)$ and $z_2(x) = y'(x)$. It is then clear that $Z(x)$ is the solution of the first order system of IVPs:

  $$\begin{aligned} Z' &= \begin{bmatrix} z_1'(x) \\ z_2'(x) \end{bmatrix}, \\ &= \begin{bmatrix} y'(x) \\ y''(x) \end{bmatrix}, \\ &= \begin{bmatrix} z_2(x) \\ f(x, y, y') \end{bmatrix}, \\ &= \begin{bmatrix} z_2(x) \\ f(x, z_1, z_2) \end{bmatrix}, \\ &= F(x, Z). \end{aligned}$$

  - Note that in solving this ‘equivalent’ system for $Z(x)$, we actually determine an approximation to $y'(x)$ as well as to $y(x)$. This has implications for numerical methods as, when working with this equivalent system, we will actually be trying to accurately approximate $y'(x)$ and this may be a more difficult problem than just approximating $y(x)$.
  - Note also that to determine a unique solution to our problem we must prescribe initial conditions for $Z(a)$, that is for both $y(a)$ and $y'(a)$.
  - Higher order (greater than second order) equations can be reduced to first order systems in a similar way.

2. **Taylor Series Methods**:

- If $f(x, y)$ is sufficiently differentiable wrt x and y then we can determine the Taylor series expansion of the unique solution $y(x)$ to

$$y' = f(x, y), \quad y(a) = y_0,$$

by differentiating the ODE at the point $x_0 = a$. That is, for x near $x_0 = a$ we have,

$$y(x) = y(x_0) + (x - x_0)y'(x_0) + \frac{(x - x_0)^2}{2}y''(x_0) + \cdots,$$

- To generate the TS coefficients, $y^{(n)}(x_0)/n!$, we differentiate the ODE and evaluate at $x = x_0 = a$. The first few terms are computed from the expressions,

$$\begin{aligned} y'(x) = f(x, y) &= f, \\ y''(x) = \frac{d}{dx}f(x, y) &= f_x + f_y y' \\ &= f_x + f_y f. \\ y'''(x) = \frac{d}{dx}[y''(x)] &= (f_{xx} + f_{xy}f) + (f_{yx} + f_{yy}f)f + f_y(f_x + f_y f), \\ &= f_{xx} + 2f_{xy}f + f_{yy}f^2 + f_y f_x + f_y^2 f. \end{aligned}$$

- In general, if $f(x, y)$ is sufficiently differentiable, we can use the first $(k + 1)$ terms of the Taylor series as an approximation to $y(x)$ for $|(x - x_0)|$ 'small'. That is, we can approximate $y(x)$ by $\hat{z}_{k,0}(x)$,

$$\hat{z}_{k,0}(x) \equiv y_0 + (x - x_0)y_0' + \cdots + \frac{(x - x_0)^k}{k!}y_0^{(k)}.$$

  Note that the derivatives of y become quite complicated so one usually chooses a small value of k (for example $k \leq 6$).

- One can use $\hat{z}_{k,0}(x_1)$ as an approximation, y_1, to $y(x_1)$. We can then evaluate the derivatives of $y(x)$ at $x = x_1$ to define a new polynomial $\hat{z}_{k,1}(x)$ as an approximation to $y(x)$ for $|(x - x_1)|$ 'small' and repeat the procedure.

  Note:

  (a) The resulting $\hat{z}_{k,j}(x)$ for $j = 0, 1, \cdots$ define a piecewise polynomial approximation to $y(x)$ that is continuous on $[a, b]$.

  (b) How do we effectively choose $h_j = (x_j - x_{j-1})$ and k?

- Let $T_k(x, y_{j-1})$ denote the first $k + 1$ terms of the Taylor series expanded about the discrete approximation, (x_{j-1}, y_{j-1}), and $\hat{z}_{k,j}(x)$ be the polynomial approximation (to $y(x)$) associated with this truncated Taylor series. That is,

$$\begin{aligned} \hat{z}_{k,j}(x) &= y_{j-1} + h\, T_k(x, y_{j-1}), \\ T_k(x, y_{j-1}) &\equiv f(x_{j-1}, y_{j-1}) + \frac{h}{2}f'(x_{j-1}, y_{j-1}) + \cdots \frac{h^{k-1}}{k!}f^{(k-1)}(x_{j-1}, y_{j-1}), \end{aligned}$$

  where $h = (x - x_{j-1})$.

  A simple, constant stepsize (fixed h) numerical method is then given by:

```
-Set $h = (b-a)/N$;
-for $j = 1, 2, \cdots N$
        $x_j = x_{j-1} + h$;
        $y_j = y_{j-1} + h\ T_k(x_{j-1}, y_{j-1})$;
-end
```

3. **Local and Global Errors:**

   Note that, strictly speaking, $z_{k,j}(x)$ is not a direct approximation to $y(x)$ but to the solution of the 'local' IVP:

   $$z_j^{'} = f(x, z_j), \quad z_j(x_{j-1}) = y_{j-1}.$$

   Since y_{j-1} will not be equal to $y(x_{j-1})$ in general, the solution to this local problem, $z_j(x)$, will not then be the same as $y(x)$.

   To understand and appreciate the implications of this observation we distinguish between the 'local' and 'global' errors.

   Definitions:

   - The local error associated with step j is $z_j(x_j) - y_j$.
   - The global error at x_j is $y(x_j) - y_j$.

4. **The Classical Approach:**

   A Classical (pre 1965) numerical method approximates $y(x)$ by dividing $[a, b]$ into equally spaced subintervals, $x_j = a + j\ h$, where $h = (b-a)/N$ and (proceeding in a step-by-step fashion), generates y_j after $y_1, y_2, \cdots y_{j-1}$ have been determined.

   - If the Taylor series method is used in this way, then the TS theorem with remainder shows that the local error on step j (for the TS method of order k) is:
     $$E_j = \frac{h^{k+1} f^{(k)}(\eta_j, z_j(\eta_j))}{(k+1))!} = \frac{h^{k+1} z_j^{(k+1)}(\eta_j)}{(k+1)!}.$$
   - An Example:
     If $k = 1$ we have Eulers Method where
     $$y_j = y_{j-1} + h\ f(x_{j-1}, y_{j-1}),$$
     and the associated local error satisfies,
     $$LE_j = \frac{h^2}{2} y^{''}(\eta_j).$$

- Classical convergence result (for a fixed-step method):
  Definition: A method is said to converge if and only if,

$$\lim_{h\to 0,(N\to\infty)} \left\{ \max_{j=1,2,\cdots N} |y(x_j) - y_j| \right\} \to 0.$$

- **Theorem:** (typical of classical convergence results)
  Let $[x_j, y_j]_{j=0}^N$ be the approximate solution of the IVP, $y' = f(x,y),\ y(a) = y_0$ over $[a,b]$ generated by Euler's method with constant stepsize $h = (b-a)/N$. If the exact solution, $y(x)$, has a continuous second derivative and $|f_y| < L,\ |y''(x)| < Y$ then the associated global error, $e_j = y(x_j) - y_j$, at the points $x_j = a + j\ h$ satisfies,

$$\begin{aligned} |e_j| &\leq \frac{hY}{2L}(e^{(x_j - x_0)L} - 1) + e^{(x_j-x_0)L}|e_0|, \\ &\leq \frac{hY}{2L}(e^{(b-a)L} - 1) + e^{(b-a)L}|e_0|. \end{aligned}$$

Note:

(a) e_0 will usually be equal to zero.

(b) This bound is generally pessimistic as it is exponential in $(b-a)$ where linear error growth is often observed.

(c) In the general case one can show that when local error is $O(h^{p+1})$ the global error is $O(h^p)$.

Proof of this Theorem (Outline only): Eulers Method satisfies,

$$y_j = y_{j-1} + hf(x_{j-1}, y_{j-1}).$$

A Taylor series expansion of $y(x)$ about $x = x_{j-1}$ implies

$$y(x_j) = y(x_{j-1}) + hf(x_{j-1}, y(x_{j-1})) + \frac{h^2}{2}y''(\eta_j).$$

Subtracting the first equation from the second we obtain,

$$y(x_j) - y_j = y(x_{j-1}) - y_{j-1} + h\left[f(x_{j-1}, y(x_{j-1})) - f(x_{j-1}, y_{j-1})\right] + \frac{h^2}{2}y''(\eta_j).$$

If $Y = \max_{x\in[a,b]} |y''(x)|$ and $|f_y| \leq L$, then, from the definition of e_j and the observation that $f(x,y)$ satisfies a Lipschitz condition with respect to y, we have

$$\begin{aligned} |e_j| &\leq |e_{j-1}| + hL|y(x_{j-1}) - y_{j-1}| + |\frac{h^2}{2}y''(\eta_j)|, \\ &\leq |e_{j-1}| + hL|e_{j-1}| + \frac{h^2}{2}Y, \\ &= |e_{j-1}|(1 + hL) + \frac{h^2}{2}Y. \end{aligned}$$

This is a linear recurrence relation (or inequality) which after some work (straightforward) can be shown to imply our desired result,

$$|e_j| \leq \frac{hY}{2L}(e^{(b-a)L} - 1) + e^{(b-a)L}|e_0|.$$

Note that this is only an upper bound on the global error and it may not be sharp.

5. **An Example:**

   Consider the following equation,

   $$y' = y, \;\; y(0) = 1, \;\; \text{on } [0, 1].$$

   Now since $\frac{\partial f}{\partial y} = 1$, $L = 1$ and since $y(x) = e^x$, we have $Y = e$ and $e_0 = 0$.

   Applying our error bound with $h = 1/N$ and $y_N \approx y(1) = e$ we obtain,

   $$|GE_N| = |Y_N - e| \leq \frac{he}{2}(e - 1) < 2.4h.$$

   But for $h = .1$ we observe that $y_{10} = 2.5937..$ with an associated true error of .1246... This error bound is .24.

6. **Limitations and Difficulties with Classical Approach:**

   - Analysis is valid only in the limit as $h \to 0$.
   - Bounds are usually very pessimistic (can overestimate the error by several orders of magnitude).
   - Analysis does not consider the affect of f.p. arithmetic.

     To extend the analysis, consider applying Eulers method with roundoff error: Assume $fl(f(x_{j-1}, y_{j-1})) = f(x_{j-1}, y_{j-1}) + \epsilon_j$ and

     $$\begin{aligned} y_j &= y_{j-1} \oplus h \otimes fl(f(x_{j-1}, y_{j-1})), \\ &= y_{j-1} + hf(x_{j-1}, y_{j-1}) + h\epsilon_j + \rho_j, \end{aligned}$$

     where $|\epsilon_j|, \; |\rho_j| \; < \; \mu$.

     Then, proceeding as before we obtain,

     $$|e_j| < |e_{j-1}|(1 + hL) + \frac{h^2}{2}\bar{M},$$

     where $\bar{M} = Y + \mu/h + \mu/(h^2)$.

     Therefore the revised error bound becomes:

     $$\begin{aligned} |e_j| &\leq e^{(b-a)L}|e_0| + \frac{h\bar{M}}{2L}(e^{(b-a)L} - 1), \\ &= e^{(b-a)L}|e_0| + (e^{(b-a)L} - 1)(\frac{hY}{2L} + \frac{\mu}{2L} + \frac{\mu}{2hL}). \end{aligned}$$

     So, as $h \to 0$, the term $\frac{\mu}{2hL}$ will become unbounded (unless the precision changes) and we will not observe convergence.

- Special difficulties with fixed-h Euler:
  - The low order results in requiring a small stepsize, which leads to a large number of derivative evaluations and excessive amount of computer time.
  - The use of a constant stepsize can be inappropriate if the solution behaves differently on parts of the interval of interest. For example in integrating satellite orbits 'close approaches' typically requires a smaller stepsize to ensure accuracy.

7. **Runge-Kutta Methods:**

   (a) We will consider a general class of one-step formulas of the form:

   $$y_j = y_{j-1} + h\Phi(x_{j-1}, y_{j-1}). \tag{4}$$

   where Φ satisfies a Lipschitz condition with respect to y. That is,

   $$|\Phi(x,u) - \Phi(x,v)| \leq \mathcal{L}|u - v|.$$

   We will consider a variety of choices for Φ and will observe that, in each case considered, Φ will be Lipschitz if f is.

   Two examples of such formulas are:

   Euler: $\Phi \equiv f$.

   Taylor Series: $\Phi \equiv T_k(x,y)$.

   (b) **Definition:** A formula (4) is of <u>order p</u> if for all sufficiently differentiable functions $y(x)$ we have,

   $$y(x_j) - y(x_{j-1}) - h\Phi(x_{j-1}, y(x_{j-1})) = O(h^{p+1}). \tag{5}$$

   Note that:

   i. The LHS of (5) is defined to be the <u>Local Truncation Error (LTE)</u> of the formula.

   ii. Order p implies that both the LE and the LTE are $O(h^{p+1})$. (This follows by substituting $z_j(x)$ for $y(x)$ in the definition.)

   (c) **Main Result:**

   <u>Theorem:</u> A p^{th} order formula applied to an initial value problem with constant stepsize h satisfies,

   $$|y(x_j) - y_j| \leq |e_0|e^{\mathcal{L}(b-a)} + \frac{Ch^p}{\mathcal{L}}(e^{\mathcal{L}(b-a)} - 1).$$

   This result can be proved using a similar argument to that used in the Euler convergence theorem.

(d) We wish to consider formulas Φ that are less 'expensive' than higher order Taylor Series and yet are higher order than Euler's formula. Consider a formula Φ based on $\underline{2}$ derivative evaluations. That is,

$$\Phi(x_{j-1}, y_{j-1}) = \omega_1 k_1 + \omega_2 k_2,$$

where,

$$\begin{aligned} k_1 &= f(x_{j-1}, y_{j-1}), \\ k_2 &= f(x_{j-1} + \alpha h, y_{j-1} + h\beta k_1). \end{aligned}$$

We determine the parameters $\omega_1, \omega_2, \alpha, \beta$ to obtain as high an order formula as possible. From the definition of order we have order p if

$$y(x_j) = y(x_{j-1}) + h(\omega_1 k_1 + \omega_2 k_2) + O(h^{p+1}) \tag{6}$$

for all sufficiently differentiable functions $y(x)$. To derive such a formula we expand $y(x_j), k_1, k_2$ in Taylor Series about the point (x_{j-1}, y_{j-1}), equate like powers of h on both sides of (6), and set $\alpha, \beta, \omega_1, \omega_2$ accordingly.

In what follows we omit arguments when they are evaluated at the point (x_{j-1}, y_{j-1}). The expansion of the LHS of (6) is:

$$\begin{aligned} LHS &= y(x_j), \\ &= y(x_{j-1}) + hy'(x_{j-1}) + \frac{h^2}{2}y''(x_{j-1}) + \frac{h^3}{6}y'''(x_{j-1}) + O(h^4), \\ &= y(x_{j-1}) + hf + \frac{h^2}{2}(f_x + f_y f) \\ &\quad + \frac{h^3}{6}(f_{xx} + 2f_{xy}f + f_{yy}f^2 + f_y f_x + f_y^2 f) + O(h^4). \end{aligned}$$

The expansion of the RHS of (6) is more complicated and first requires the expansions of k_1 and k_2,

$$\begin{aligned} k_1 &= f, \\ k_2 &= f(x_{j-1} + \alpha h, y(x_{j-1}) + \beta h k_1), \\ &= f(x_{j-1}, y(x_{j-1}) + \beta h f) + (\alpha h) f_x(x_{j-1}, y(x_{j-1}) + \beta h f) \\ &\quad + \frac{\alpha^2 h^2}{2} f_{xx}(x_{j-1}, y(x_{j-1}) + \beta h f) + O(h^3), \\ &= \left[f + \beta h f f_y + \frac{(\beta h f)^2}{2} f_{yy} + O(h^3)\right] \\ &\quad + \left[\alpha h f_x + \alpha\beta h^2 f f_{xy} + O(h^3)\right] + \left[\frac{\alpha^2 h^2}{2} f_{xx} + O(h^3)\right], \\ &= f + (\beta f f_y + \alpha f_x)\, h + (\frac{\beta^2}{2} f^2 f_{yy} + \alpha\beta f f_{xy} + \frac{\alpha^2}{2} f_{xx})\, h^2 + O(h^3). \end{aligned}$$

The expansion of the RHS of (6) then is (with these substitutions for k_1 and k_2)

$$\begin{aligned}
RHS &= y(x_{j-1}) + h(\omega_1 k_1 + \omega_2 k_2), \\
&= y(x_{j-1}) + h\omega_1 f + h\omega_2 [\cdots] + O(h^4), \\
&= y(x_{j-1}) + [(\omega_1 + \omega_2) f] \; h + [\omega_2(\beta f f_y + \alpha f_x)] \; h^2 \\
&\quad + \left[\omega_2(\frac{\beta^2}{2} f^2 f_{yy} + \alpha\beta f f_{xy} + \frac{\alpha^2}{2} f_{xx})\right] \; h^3 + O(h^4).
\end{aligned}$$

Finally these expansions are true for all values of h, so equating like powers of h, in the LHS and RHS expansions, we observe the following:

For order 0 : The coefficients of h^0 always agree and we have order at least zero for any choice of the parameters.

For order 1: If $\omega_1 + \omega_2 = 1$ the coefficients of h^1 agree and we have at least order 1.

For order 2: In addition to satisfying the order 1 constraints we must have the coefficient of h^2 the same. That is $\alpha\omega_2 = 1/2$ and $\beta\omega_2 = 1/2$.

For order 3: In addition to satisfying the order 2 constraints we must have the coefficients of h^3 the same. That is we must satisfy the equations,

$$\begin{aligned}
\omega_2 \alpha^2 &= \frac{1}{3}, \\
\omega_2 \alpha\beta &= \frac{1}{3}, \\
\omega_2 \beta^2 &= \frac{1}{3}, \\
\frac{1}{6} f_{xy} &= ?, \\
\frac{1}{6} f_y^2 &= ?.
\end{aligned}$$

Note that there are not enough terms in the coefficient of h^3 in the expansion of the RHS to match the expansion of the LHS. We cannot therefore equate the coefficients of h^3 and the maximum order we can obtain is order 2. Our formula will be order 2 for any choice of $\omega_2 \neq 0$, with $\omega_1 = 1 - \omega_2$ and $\alpha = \beta = \frac{1}{2\omega_2}$. This is a one-parameter family of 2^{nd}-order Runge-Kutta formulas.

Three popular choices from this family are:

Modified Euler: $\omega_2 = 1/2$

$$\begin{aligned}
k_1 &= f(x_{j-1}, y_{j-1}), \\
k_2 &= f(x_{j-1} + h, y_{j-1} + hk_1), \\
y_j &= y_{j-1} + \frac{h}{2}(k_1 + k_2).
\end{aligned}$$

Midpoint: $\omega_2 = 1$

$$\begin{aligned} k_1 &= f(x_{j-1}, y_{j-1}), \\ k_2 &= f(x_{j-1} + \frac{h}{2}, y_{j-1} + \frac{h}{2}k_1), \\ y_j &= y_{j-1} + hk_2. \end{aligned}$$

Heun's Formula: $\omega_2 = 3/4$

$$\begin{aligned} k_1 &= f(x_{j-1}, y_{j-1}), \\ k_2 &= f(x_{j-1} + \frac{2}{3}h, y_{j-1} + \frac{2}{3}hk_1), \\ y_j &= y_{j-1} + \frac{h}{4}(k_1 + 3k_2). \end{aligned}$$

8. **Higher-order Runge-Kutta formulas:**

An s-stage explicit Runge-Kutta formula uses s derivative evaluations and has the form:

$$y_j = y_{j-1} + h(\omega_1 k_1 + \omega_2 k_2 \cdots + \omega_s k_s),$$

where

$$\begin{aligned} k_1 &= f(x_{j-1}, y_{j-1}), \\ k_2 &= f(x_{j-1} + \alpha_2 h, y_{j-1} + h\beta_{21}k_1), \\ &\vdots \\ k_s &= f(x_{j-1} + \alpha_s h, y_{j-1} + h\sum_{r=1}^{s-1}\beta_{sr}k_r). \end{aligned}$$

This formula is represented by the tableau,

$$\begin{array}{c|ccccc} - & - & & & & \\ \alpha_2 & \beta_{21} & - & & & \\ \vdots & \vdots & & & & \\ \alpha_s & \beta_{s1} & \beta_{s2} & \ldots & \beta_{s-1,s} & - \\ \hline & \omega_1 & \omega_2 & \ldots & & \omega_s \end{array}$$

These $\frac{s(s-1)}{2} + (s-1) + s$ parameters are usually chosen to maximise the order of the formula.

The maximum attainable order for an s-stage Runge-Kutta formula is given by the following table:

s	1	2	3	4	5	6
max order	1	2	3	4	4	5

Note that the derivations of these maximal order formulas can be very messy and tedious, but essentially they follow (as outlined above for the case $s = 2$) by expanding each of the k_r in a 2-dimensional Taylor series.

An Example – The Classical Fourth Order Runge Formula (1895)

$$\begin{array}{c|cccc}
- & - & & & \\
1/2 & 1/2 & - & & \\
1/2 & 0 & 1/2 & - & \\
1 & 0 & 0 & 1 & - \\
\hline
 & 1/6 & 1/3 & 1/3 & 1/6
\end{array}$$

9. **Error Estimates:**

(a) Ideally a method would estimate a bound on the global error and adjust the stepsize, h, to keep the magnitude of the global error less than a tolerance. Such computable bounds are possible but are usually pessimistic and inefficient to implement.

(b) On the other hand, <u>local errors</u> can be reliably and efficiently estimated and controlled. Consider a method which keeps the magnitude of the local error less than $h\,TOL$ on each step.

That is, if $z_{j-1}(x)$ is the local solution on step j,

$$z_{j-1}^{'} = f(x, z_{j-1}),\ z_{j-1}(x_{j-1}) = y_{j-1},$$

then a method will adjust $h = x_j - x_{j-1}$ to ensure that $|z_{j-1}(x_j) - y_j| \leq h\,TOL$, for $j = 1, 2 \cdots N_{TOL}$.

(c) With this type of error control one can show that, for the resulting $(x_j, y_j)_{j=0}^{N_{TOL}}$ there exists a piecewise polynomial, $Z(x) \in C^1[a, b]$ such that $Z(x_j) = y_j$ for $j = 0, 1, \cdots N_{TOL}$ and for $x \in [a, b]$,

$$|Z^{'}(x) - f(x, Z)| \leq TOL.$$

We can also show that,

$$|y(x_j) - y_j| \leq \frac{TOL}{L}(e^{L(x_j - a)} - 1).$$

(d) Derivation of Local Error Estimates for Runge-Kutta formulas:

Consider the Modified Euler Formula:

$$\begin{array}{c|cc}
- & - & \\
1 & 1 & - \\
\hline
 & 1/2 & 1/2
\end{array}$$

We have shown

$$\begin{aligned}
z_{j-1}(x_j) &= y_{j-1} + \frac{h}{2}(k_1 + k_2) \\
&\quad + \left[\frac{1}{4}f^2 f_{yy} + \frac{1}{2}f f_{xy} + \frac{1}{4}f_{xx} - y^{'''}(x_j)\right] h^3 + O(h^4), \\
&= y_j + \left[\frac{1}{12}f_{yy}f^2 + \frac{1}{6}f f_{xy} + \frac{1}{12}f_{xx} - f_{xy} - f_y^2 f\right] h^3 + O(h^4), \\
&\equiv y_j + c(f)h^3 + O(h^4).
\end{aligned}$$

It then follows that the local error, LE, satisfies

$$LE = c(f)h^3 + O(h^4),$$

where $c(f)$ is a complicated function of f. There are two general strategies for estimating $c(f)$ – the use of "step halving" and the use of a 3rd order "companion formula".

Step Halving: let $\hat{y}_j$ be the approximation to $z_{j-1}(x_j)$ computed with two steps of size $h/2$. If $c(f)$ is almost constant the we can show

$$z_{j-1}(x_j) = \hat{y}_j + 2c(f)(\frac{h}{2})^3 + O(h^4)$$

and from above

$$z_{j-1}(x_j) = y_j + c(f)h^3 + O(h^4).$$

Therefore the local error associated with $\hat{y}_j$, $\widehat{LE}$, is

$$\begin{aligned}
\widehat{LE} &= 2c(f)(\frac{h}{2})^3 + O(h^4), \\
&= \frac{1}{3}(y_j - \hat{y}_j) + O(h^4).
\end{aligned}$$

The method could then compute $\hat{y}_j, y_j$ and accept $\hat{y}_j$ only if $\frac{1}{3}|y_j - \hat{y}_j| < h\,TOL$.

Note that this strategy requires five derivative evaluations on each step and assumes that each of the components of $c(f)$ is $\underline{\text{slowly}}$ varying.

Third Order Companion Formula: To estimate the local error associated with the Modified Euler formula consider the use of a 3-stage, 3rd order Runge-Kutta formula,

$$\begin{aligned}
\hat{y}_j &= y_{j-1} + h(\hat{\omega}_1 k_1 + \hat{\omega}_2 k_2 + \hat{\omega}_3 k_3), \\
&= z_{j-1}(x_j) + O(h^4),
\end{aligned}$$

We also have

$$\begin{aligned}
y_j &= y_{j-1} + \frac{h}{2}(k_1 + k_2), \\
&= z_{j-1}(x_j) - c(f)h^3 + O(h^4).
\end{aligned}$$

Subtracting these two equations we have the local error estimate,

$$est_j \equiv (\hat{y}_j - y_j) = c(f)h^3 + O(h^4).$$

Note that, for any 3rd order formula, $k_1 = \hat{k}_1$, so we require at most 4 derivative evaluations per step to compute both y_j and est_j. Furthermore, if $\hat{\alpha}_2 = \alpha_2 = 1$ and $\hat{\beta}_{21} = \beta_{21} = 1$, we have $\hat{k}_2 = k_2$ and the cost is only three derivative evaluations per step. The obvious question is can one derive such a 3-stage 3rd order Runge-Kutta formula ? The answer is yes and the following tableau with $\hat{\alpha}_3 \neq 1$ defines a one-parameter family of such "companion formulas" for the Modified Euler formula:

$$\begin{array}{c|ccc} - & - & & \\ 1 & 1 & - & \\ \hat{\alpha}_3 & \hat{\beta}_{31} & \hat{\beta}_{32} & - \\ \hline & \hat{\omega}_1 & \hat{\omega}_2 & \hat{\omega}_3 \end{array}$$

with

$$\hat{\beta}_{31} = \hat{\alpha}_3^2, \hat{\beta}_{32} = \hat{\alpha}_3 - \hat{\alpha}_3^2,\ \hat{\omega}_2 = \frac{1}{6(\hat{\alpha}_3 - 1)},\ \hat{\omega}_3 = \frac{-1}{6(\hat{\alpha}_3 - 1)},\ \hat{\omega}_1 = 1 - (\frac{1 + 3\hat{\alpha}_3}{6\hat{\alpha}_3}).$$

Generalization to Higher Order: This idea of using a "companion formula" of order $p + 1$ to estimate the local error of a p^{th} order formula leads to the derivation of s-stage, order $(p, p + 1)$ formula pairs with the fewest number of stages. Such formula pairs can be characterized by the tableau:

$$\begin{array}{c|ccccc} - & - & & & & \\ \alpha_2 & \beta_{21} & - & & & \\ \vdots & \vdots & & & & \\ \alpha_s & \beta_{s1} & \dots & & \beta_{s-1,s} & - \\ \hline & \omega_1 & \omega_2 & \dots & & \omega_s \\ & \hat{\omega}_1 & \hat{\omega}_2 & \dots & & \hat{\omega}_s \end{array}$$

where

$$\begin{aligned} y_j &= y_{j-1} + h\sum_{r=1}^{s} \omega_r k_r \\ &= z_{j-1}(x_j) - c(f)h^{p+1} + O(h^{p+2}), \\ \hat{y}_j &= y_{j-1} + h\sum_{r=1}^{s} \hat{\omega}_r k_r \\ &= z_{j-1}(x_j) + O(h^{p+2}), \\ est_j &= (\hat{y}_j - y_j) \\ &= c(f)h^{p+1} + O(h^{p+2}). \end{aligned}$$

Note that the error estimate is a reliable estimate of the local error associated with the lower order (order p) formula. The following table gives the fewest number of stages required to generate formula pairs of a given order.

order pair	(2,3)	(3,4)	(4,5)	(5,6)	(6,7)
fewest stages	3	4	6	8	10

10. **Stepsize Control:**

- Step is accepted only if $|est_j| < hTOL$.
- If h is too large, the step will be rejected and the derivative evaluations will be wasted.
- If h is too small, there will be many steps and more function evaluations than necessary.

The usual strategy for choosing the attempted stepsize, h, for the next step is based on 'aiming' at the largest h which will result in an accepted step on the current step. If we assume that $c(f)$ is slowly varying then,

$$|est_j| = |c(f)|h_j^{p+1} + O(h^{p+2}),$$

and on the next step attempted step, $h_{j+1} = \gamma h_j$, we want

$$|est_{j+1}| \approx TOL\ h_{j+1}.$$

But

$$\begin{aligned} |est_{j+1}| &\approx |c(f)|(\gamma h_j)^{p+1}, \\ &= \gamma^{p+1}|est_j|. \end{aligned}$$

We can then expect

$$|est_{j+1}| \approx TOL\ h_{j+1},$$

if

$$\gamma^{p+1}|est_j| \approx TOL\ (\gamma h_j),$$

which is equivalent to

$$\gamma^p|est_j| \approx TOL\ h_j.$$

The choice of γ to satisfy this heuristic is then,

$$\gamma = \left(\frac{TOL\ h_j}{|est_j|}\right)^{1/p}.$$

A typical step-choosing heuristic, justified by the above discussion, is to use the formula,

$$h_{j+1} = .9\left(\frac{TOL\ h_j}{|est_j|}\right)^{1/p} h_j,$$

where .9 is a 'safety factor'. The formula works for use after a rejected step as well but must be modified slightly when round-off errors are significant.

6 Gauss Quadrature and Multidimensional Quadrature

1. **The basic Problem – approximation of integrals:**

   We will consider how best to compute an approximation to the definite integral:

   $$I(f) \equiv \int_a^b f(x)dx.$$

   The obvious generic approach is to approximate the integrand $f(x)$ on the interval $[a,b]$ by a function that can be integrated exactly (such as a polynomial) and then take the integral of the approximating function to be an approximation to $I(f)$.

2. **Interpolatory Rules:**

   When the approximating function is an interpolating polynomial, $P_n(x)$, the corresponding approximation $I(P_n(x))$ is called an interpolatory rule. You have aleady investigated several widely used interpolatory rules. In this secion we will investigate and justify a special class of interpolatory rules.

   Consider writing $P_n(x)$ in Lagrange form,

   $$P_n(x) = \sum_{i=0}^{n} f(x_i) l_i(x),$$

   where $l_i(x)$ is defined by

   $$l_i(x) = \prod_{j=0, j\neq i}^{n} \left(\frac{x - x_j}{x_i - x_j} \right).$$

   We then have

   $$\begin{aligned} \int_a^b P_n(x)dx &= \int_a^b \sum_{i=0}^{n} f(x_i) l_i(x) dx \\ &= \sum_{i=0}^{n} f(x_i) \int_a^b l_i(x) dx \\ &= \sum_{i=0}^{n} \omega_i f(x_i). \end{aligned}$$

   Note:

   - The 'weights' (the ω_i's) depend only on the interval (the value of a and b) and on the x_i's. In particular these weights are independent of the integrand.
   - The interpolatory rules then approximate $I(f)$ by a linear combination of sampled integrand evaluations.

3. **Errors in Interpolatory Rules:**

The error associated with an interpolatory rule is $E(f) = I(f) - I(P_n)$ and satisfies,

$$\begin{aligned} E(f) &= \int_a^b f(x)dx - \int_a^b P_n(x)dx = \int_a^b [f(x) - P_n(x)]dx, \\ &= \int_a^b E_n(x)dx, \end{aligned}$$

where $E_n(x)$ is the error in polynomial interpolation and satisfies,

$$\begin{aligned} E_n(x) &= (x - x_0)(x - x_1)\cdots(x - x_n)f[x_0, x_1, \cdots x_n, x], \\ &= \Pi_n(x)f[x_0, x_1, \cdots x_n, x], \end{aligned}$$

with $\Pi_n(x) \equiv \prod_{j=0}^n (x - x_j)$.

This expression for the error is valid for all interpolatory rules. In some special cases we can simplify this expression to obtain estimates and/or more insight into the behaviour of the error.

- First special case – If $\Pi_n(x)$ is of one sign (on $[a, b]$) then the Mean Value Theorem for Integrals implies,

$$\begin{aligned} E(f) &= \int_a^b f[x_0, x_1, \cdots x_n, x]\Pi_n(x)dx, \\ &= f[x_0, x_1, \cdots x_n, \xi] \int_a^b \Pi_n(x)dx, \end{aligned}$$

  for some $\xi \in [a, b]$. Also since $f[x_0, x_1, \cdots x_n, \xi] = \frac{f^{(n+1)}(\eta)}{(n+1)!}$ for some $\eta \in (a, b)$, we have shown that if $\Pi_n(x)$ is of one sign then,

$$\boxed{E(f) = \frac{1}{(n+1)!} f^{(n+1)}(\eta) \int_a^b \Pi_n(x)dx}$$

- Second special case – If $\int_a^b \Pi_n(x)dx = 0$ we have, for arbitrary x_{n+1},

$$f[x_0, x_1, \cdots x_n, x] = f[x_0, x_1, \cdots x_n, x_{n+1}] + f[x_0, x_1, \cdots x_{n+1}, x](x - x_{n+1}),$$

  and therefore,

$$\begin{aligned} E(F) &= \int_a^b f[x_0, x_1, \cdots x_n, x]\Pi_n(x)dx, \\ &= \int_a^b f[x_0, x_1, \cdots x_{n+1}]\Pi_n(x)dx + \int_a^b f[x_0, x_1, \cdots x_{n+1}, x]\Pi_{n+1}(x)dx, \\ &= \int_a^b f[x_0, x_1, \cdots x_{n+1}, x]\Pi_{n+1}(x)dx. \end{aligned}$$

  As a result, if $\int_a^b \Pi_n(x)dx = 0$ and we can choose x_{n+1} so that $\Pi_{n+1}(x)$ is of one sign, then using a similar argument to that presented in the first special case, it follows that, if $\int_a^b \Pi_n(x)dx = 0$ and $\Pi_{n+1}(x)$ is of one sign,

$$\boxed{E(f) = \frac{1}{(n+2)!} f^{(n+2)}(\eta) \int_a^b \Pi_{n+1}(x)dx}$$

4. **Examples of Interpolatory Rules:**

(a) Trapezoidal Rule (an example of the first special case):

$$T(f) \equiv \int_a^b P_1(x)dx,$$

where $x_0 = a$ and $x_1 = b$. We then have,

$$P_1(x) = l_0(x)f_0 + l_1(x)f_1 = \frac{x - x_1}{x_0 - x_1} f_0 + \frac{x - x_0}{x_1 - x_0} f_1.$$

Therefore we have

$$\begin{aligned} T(f) &= \int_a^b \frac{x-b}{a-b} dx f(a) + \int_a^b \frac{x-a}{b-a} dx f(b), \\ &= \left(\frac{b-a}{2}\right) f(a) + \left(\frac{b-a}{2}\right) f(b), \\ &= \left(\frac{b-a}{2}\right) [f(a) + f(b)]. \end{aligned}$$

We also have that $\Pi_1(x) = (x-a)(x-b)$ is negative for $x \in [a,b]$ and $\int_a^b \Pi_1(x)dx = -\frac{(b-a)^3}{6}$. We therefore have satisfied the conditions of the first special case and this implies,

$$\boxed{T(f) = (\tfrac{b-a}{2})[f(a) + f(b)], \;\; E^T(f) = \tfrac{-f''(\eta)}{12}(b-a)^3.}$$

(b) Simpsons Rule (an example of the second special case):

$$S(f) \equiv \int_a^b P_2(x)dx,$$

with $x_0 = a, x_1 = \frac{a+b}{2}, x_2 = b$.
Exercise: Using

$$P_2(x) = l_0(x)f(a) + l_1(x)f\left(\frac{a+b}{2}\right) + l_2(x)f(b),$$

where

$$\begin{aligned} l_0(x) &= \frac{(x - \frac{a+b}{2})(x-b)}{(a - \frac{a+b}{2})(a-b)}, \\ l_1(x) &= \frac{(x-a)(x-b)}{(\frac{a+b}{2} - a)(\frac{a+b}{2} - b)}, \\ l_2(x) &= \frac{(x-a)(x - \frac{a+b}{2})}{(b-a)(b - \frac{a+b}{2})}. \end{aligned}$$

Simplify and verify (after some tedious algebra) that,

$$\begin{aligned}\int_a^b P_2(x)dx &= [\int_a^b l_0(x)dx]f(a) + [\int_a^b l_1(x)dx]f(\frac{a+b}{2}) + [\int_a^b l_2(x)dx]f(b), \\ \vdots & \ \vdots \\ &= \left(\frac{b-a}{6}\right)\left[f(a) + 4f(\frac{a+b}{2}) + f(b)\right].\end{aligned}$$

Note that for $x \in [a,b]$, $\Pi_2(x)$ is antisymmetric about $\frac{a+b}{2}$ and this implies $\int_a^b \Pi_2(x)dx = 0$. Furthermore by choosing $x_3 = \frac{a+b}{2}$ we have

$$\Pi_3(x) = (x-a)(x-\frac{a+b}{2})^2(x-b),$$

is of one sign and this implies,

$$E^S(f) = I(f) - S(F) = \frac{1}{4!}f^{(4)}(\eta)\int_a^b \Pi_3(x)dx.$$

But $\int_a^b \Pi_3(x)dx = -\frac{4}{15}(\frac{b-a}{2})^5$ so we have,

$$\boxed{S(f) = (\frac{b-a}{6})[f(a) + 4f(\frac{a+b}{2}) + f(b)], \quad E^S(f) = \frac{-f^{(4)}(\eta)}{90}(\frac{b-a}{2})^5}$$

5. **Gauss Quadrature (choosing the x_i's to "optimize" the error bound):**

   (a) Recall that the error in interpolatory rules satisfies,

   $$E(f) = \int_a^b f[x_0, x_1, \cdots x_n, x]\Pi_n(x)dx,$$

   and if $\int_a^b \Pi_n(x)dx = 0$ we have,

   $$E(f) = \int_a^b f[x_0, x_1, \cdots x_{n+1}, x]\Pi_{n+1}(x)dx,$$

   for $\underline{\text{any}}$ choice of x_{n+1}.
   Now if $\int_a^b \Pi_{n+1}(x) = 0$ as well we can repeat this argument and obtain,

   $$E(f) = \int_a^b f[x_0, x_1, \cdots x_{n+2}, x]\Pi_{n+2}(x)dx.$$

   For the general case, let $q_0(x) \equiv 1$ and $q_i(x) \equiv (x - x_{n+1})(x - x_{n+2})\cdots(x - x_{n+i})$ for $i = 1, 2, \cdots (m-1)$. We can then show that if $\int_a^b \Pi_n(x)q_i(x)dx = 0$, for $i = 0, 1, \cdots (m-1)$ then,

   $$E(f) = \int_a^b f[x_0, x_1, \cdots x_{n+m}, x]\Pi_{n+m}(x)dx.$$

(b) The key idea of Gauss Quadrature is to choose the nodes or interpolation points, $(x_0, x_1, \cdots x_n)$ such that $\int_a^b \Pi_n(x)q(x)dx = 0$ for all polynomials, $q(x)$, of degree at most n. Therefore, in particular for the choice $q(x) = q_i(x)$ for $i = 0, 1, \cdots n$ we have $\int_a^b \Pi_n(x)q_i(x)dx = 0$, and from the above observation,

$$E(f) = \int_a^b f[x_0, x_1, \cdots x_{2n+1}, x]\Pi_{2n+1}(x)dx.$$

To ensure that $\Pi_{2n+1}(x)$ is of one sign for $x \in [a\ b]$ we can choose $x_{n+i} = x_i$ for $i = 1, 2, \cdots n+1$ and we then have $\Pi_{2n+1}(x) = \Pi_n^2(x)$ with the corresponding error expression,

$$\begin{aligned} E(f) &= f[x_0, x_1, \cdots x_{2n+1}, \xi] \int_a^b \Pi_n^2(x)dx, \\ &= \frac{1}{(2n+2)!} f^{(2n+2)}(\eta) s_{n+1}, \end{aligned}$$

where $s_{n+1} = \int_a^b \Pi_n^2(x)dx$.

Note that such rules will be exact for all polynomials of degree at most $2n+1$. That is, if the integrand is a polynomial of degree less than $2n+2$ the corresponding Gauss Quadrature interpolatory rule (based on the n carefully chosen points) will give the exact answer.

(c) How do we choose the x_i's to ensure that $\int_a^b \Pi_n(x)q(x)dx = 0$ for all polynomials, $q(x)$ of degree at most n ?

This question leads to the study of orthogonal polynomials.

- Definition: The set of polynomials $\{r_0(x), r_1(x), \cdots r_k(x)\}$ is orthogonal on $[-1, 1]$ iff the following two conditions are satisfied:
  i. $\int_{-1}^1 r_i(x)r_j(x)dx = 0$, for $i \neq j$,
  ii. The degree of $r_i(x)$ is i for $i = 0, 1, \cdots k$.
- Properties of orthogonal polynomials:
  i. Any polynomial $q_s(x)$ of degree $s \leq k$ can be expressed as.
  $$q_s(x) = \sum_{j=0}^{s} c_j r_j(x).$$
  ii. $r_k(x)$ is orthogonal to all polynomials of degree less than k. (This follows from the previous property.)
  iii. $r_k(x)$ has k simple zeros all in the interval $[-1, 1]$.

  Proof:

  For $r_k(x)$, let $\{\mu_1, \mu_2, \cdots \mu_m\}$ be the set of points in $[-1, 1]$ where $r_k(x)$ changes sign. It is clear that each μ_j is a zero of $r_k(x)$ and all simple zeros of $r_k(x)$ in $[-1, 1]$ must be in this set.

  We then have $m \leq k$ as the maximum number of zeros of a polynomial of degree k is k. To show that $m \geq k$ (and hence $m = k$) assume the contrary, ie. $m < k$. With this assumption we have that

$$\hat{q}_m(x) \equiv \prod_{i=1}^{m}(x - \mu_i),$$

is a polynomial of degree $m < k$ that changes sign at each μ_i and,

$$\int_{-1}^{1} \hat{q}_m(x) r_k(x) dx = 0,$$

but $\hat{q}_m(x)$ and $r_k(x)$ have the same sign for all x in $[-1, 1]$ (they change sign at the same locations) and this implies a contradiction (the integrand is of one sign but the integral is zero) and therefore our assumption must be false and $m \geq k$.

iv. The $r_k(x)$ satisfy a 3-term recurrence,

$$r_{s+1}(x) = a_s(x - b_s)r_s(x) - c_s r_{s-1}(x),$$

for $s = 1, 2, \cdots k$, where the a_s are normalization constants, $r_{-1}(x) = 0$, and if $t_s = \int_{-1}^{1} r_s^2(x)dx$ then,

$$\begin{aligned} b_s &= \frac{1}{t_s}\int_{-1}^{1} x r_s^2(x)dx, \\ c_s &= \frac{a_s t_s}{a_{s-1} t_{s-1}}. \end{aligned}$$

For example, we obtain the classical Legendre polynomials if we normalise so $r_s(-1) = 1$. This leads to,

$$a_s = \frac{2s+1}{s+1}, \quad b_s = 0, \quad c_s = \frac{s}{s+1}.$$

- Orthogonal Polynomials on arbitrary intervals $[a, b]$.
To transform orthogonal polynomials defined on $[-1, 1]$ to $[a, b]$ consider the linear mapping from $[-1, 1] \rightarrow [a, b]$ defined by $x = \frac{b-a}{2}y + \frac{a+b}{2}$. The corresponding inverse mapping is $y = \frac{1}{b-a}[2x - b - a]$ and from calculus we know,

$$\int_a^b g(x)dx = (\frac{b-a}{2})\int_{-1}^{1} g(\frac{b-a}{2}y + \frac{a+b}{2})dy.$$

This relationship, combined with the properties of Legendre polynomials (that we have identified above) give a prescription for the selection of the set of interpolation points (the x_i's) that define Gauss Quadrature:
For $i = 0, 1, \cdots n$, set y_i to the i^{th} zero of the Legendre Polynomial, $r_{n+1}(y)$. With this choice we note that $\prod_{j=0}^{n}(y - y_j) = K\ r_{n+1}(y)$ for

some constant $K \neq 0$. Then with $x_i = \frac{b-a}{2}y_i + \frac{b+a}{2}$ we have,

$$\begin{aligned}
\Pi_n(\frac{b-a}{2}y + \frac{a+b}{2}) &= \prod_{j=0}^{n}(\frac{b-a}{2}y + \frac{a+b}{2} - x_j), \\
&= \prod_{j=0}^{n}(\frac{b-a}{2}y + \frac{a+b}{2} - (\frac{b-a}{2}y_j + \frac{a+b}{2})), \\
&= \prod_{j=0}^{n}\left[\frac{b-a}{2}(y - y_j)\right], \\
&= (\frac{b-a}{2})^{n+1}\prod_{j=0}^{n}(y - y_j), \\
&= (\frac{b-a}{2})^{n+1}K\, r_{n+1}(y),
\end{aligned}$$

and therefore for any polynomial, $q(x)$ of degree at most n,

$$\begin{aligned}
\int_a^b \Pi_n(x)q(x)dx &= (\frac{b-a}{2})\int_{-1}^{1}\Pi_n(\frac{b-a}{2}y + \frac{b+a}{2})q(\frac{b-a}{2}y + \frac{b+a}{2})dy, \\
&= (\frac{b-a}{2})\int_{-1}^{1}\Pi_n(\frac{b-a}{2}y + \frac{b+a}{2})\hat{q}(y)dy,
\end{aligned}$$

(where $\hat{q}(y)$ is a polynomial of degree at most n since the degree of $q(x)$ is at most n)

$$\begin{aligned}
&= (\frac{b-a}{2})^{n+2}K\int_{-1}^{1} r_{n+1}(y)\hat{q}(y)dy, \\
&= 0.
\end{aligned}$$

That is with the x_i's chosen as the 'transformed zeros' of the Legendre polynomial, $r_{n+1}(y)$, we have the interpolation points satisfying our desired property.

6. **Composite Gauss Quadrature Rules:**

Approximating the integrand with a piecewise polynomial leads to the class of Composite Rules. Let $a = x_0 < x_1 < \cdots x_M = b$ and $S(x)$ be a piecewise polynomial approximation to $f(x)$ for $x \in [a, b]$. We can then use $\int_a^b S(x)dx$ as the approximation to $I(f) = \int_a^b f(x)dx$. Recall that $S(x) \equiv p_{i,n}(x)$ for $x \in [x_{i-1}, x_i]$ $i = 1, 2, \cdots M$. From calculus we have,

$$\begin{aligned}
\int_a^b S(x)dx &= \sum_{i=1}^{M}\int_{x_{i-1}}^{x_i} S(x)dx, \\
&= \sum_{i=1}^{M}\int_{x_{i-1}}^{x_i} p_{i,n}(x)dx,
\end{aligned}$$

–A sum of basic interpolatory rules. When these interpolatory rules correspond to Guassian rules the resulting Composite Gauss rules are very effective and are widely used.

7. **Error estimates for Gauss Quadrature Rules:**

- Let $G_n(f) = \sum_{i=0}^{n} \omega_i f(x_i)$ denote the $(n+1)$ – point Gauss quadrature rule.
  (a) We have shown,
  $$I(f) - G_n(f) = O(b-a)^{2n+3}, \text{ as } (b-a) \to 0,$$
  and this is optimal.
  (b) The rules $G_{n+1}, G_{n+2}, \cdots$, are higher order and therefore asymptotically more accurate (as $(b-a) \to 0$) so we could form an error estimate from one of these. That is, we could use,
  $$\widehat{EST}_{G_n} \equiv G_{n+k}(f) - G_n(f) = E_{G_n} + O(b-a)^{2(n+k)+3}.$$
  (c) The rules G_{n+k} and G_n have at most one common interpolation point so the computation of this error estimate more than <u>doubles</u> the cost $(2n+k+2$ integrand evaluations).

- An alternative (to forming an error estimate based on G_{n+k}) is is to use the integrand evaluations already available (for the computation of $G_n(f)$) and introduce only the minimum number of extra evaluations required to obtain an effective error estimate. This approach leads to a class of quadrature rules called Kronrod quadrature rules, $K_{n+k}(f)$. The error estimate for $G_n(f)$, is then $K_{n+k}(f) - G_n(f)$, where $K_{n+k}(f)$ is more accurate <u>and</u> less expensive to compute than is $G_{n+k}(f)$. Kronrod proposed a particularly effective class of such rules where $k = n+1$,
  $$K_{2n+1}(f) \equiv \sum_{i=0}^{n} a_i f(x_i) + \sum_{j=0}^{n+1} b_j f(y_j),$$
  where the $x_i's$ are the interpolation points associated with $G_n(f)$, and the y_i's are the extra interpolation points necessary to define an accurate approximation to $I(f)$. Kronrod derived these weights (the a_i's and the b_i's) and the extra interpolation points $(y_0, y_1, \cdots y_n)$ so that the resulting rule is order $3n+3$. The resulting error estimate is then,
  $$EST_{G_n} \equiv K_{2n+1}(f) - G_n(f),$$
  with an associated cost of $2n+3$ integrand evaluations and an order of accuracy of $O((b-a)^{3n+4})$.

- The resulting Gauss-Kronrod pairs of rules can be the basis for composite quadrature rules and adaptive methods. These methods are widely used and implemented in numerical libraries.

8. **Two Dimensional Quadrature:**

Consider the problem of approximating integrals in two dimensions,

$$I(f) = \int\int_D f(x,y)dxdy,$$

This problem is more complicated than the one dimensional case since D can take many forms.

- One can develop the analogs of Gauss rules or interpolatory rules but the weights and nodes will depend on the region D. Such rules can be determined and tabulated for simple regions such as rectangles, triangles and circles. An arbitrary region must then be transformed onto one of these simple regions before the rule can be used. Such a transformation will generally be nonlinear and may introduce an approximation error as well.
- One can apply a 'product rule' where one reduces the $2D$-integral to a sequence of two $1D$-integrals:

  $$\int_a^b \int_{\alpha(y)}^{\beta(y)} f(x,y)dxdy = \int_a^b g(y)dy,$$

  where

  $$g(y) \equiv \int_{\alpha(y)}^{\beta(y)} f(x,y)dx$$

  is approximated, for a fixed value of y, by a standard method (for example, $\approx \sum_{j=0}^{M} \omega_j f(x_j, y)$, and $\int_a^b g(y)dy$ is also approximated by a standard (possibly different) standard method. That is

  $$\begin{aligned} \int_a^b g(y)dy &\approx \sum_{r=0}^{M'} \hat{\omega}_r g(y_r), \\ &\approx \sum_{r=0}^{M'} \hat{\omega}_r \left(\sum_{j=0}^{M} \omega_j f(x_j, y_r) \right), \\ &= \sum_{r=0}^{M'} \sum_{j=0}^{M} (\hat{\omega}_r \omega_j) f(x_j, y_r). \end{aligned}$$

  Note that error estimates for product rules are not easy to develop since the function $g(y) \approx \sum_{j=0}^{M} \omega_j f(x_j, y)$ will not be a 'smooth' function of y unless M and the x_j's are fixed. In particular this 'inner rule' cannot be adaptive.