1 Basic Background on Linear Algebra (August 29 2012)

**Inverse** if matrix $A$ is square and invertible then there exists $A^{-1}$ such that $AA^{-1} = A^{-1}A = I$ where $I$ denotes the identity matrix. If no such $A^{-1}$ exists then $A$ is **singular**. The inverse satisfies

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

Note that $A$ is invertible if and only if the column vectors are **linearly independent**.

**Transpose** $A^T$ is obtained by exchanging the rows and columns of $A$

$$(A^T)_{ij} = (A)_{ji}$$

We use $(A)_{ij}$ to denote the entry in row $i$ and column $j$, also denoted as $a_{ij}$ and $A = (a)$

**Conjugate transpose** $A^H$ is obtained by conjugating the entries of complex matrix $A$ and then taking the transpose. Hence where $\bar{a}$ denotes the complex conjugate of $a$

$$(A^H)_{ij} = (\bar{a})_{ji}$$

**Some important identities for $A$ square**

- **Symmetry** if $A$ is real and $A = A^T$ then $A$ is symmetric.
- **Anti-Symmetry** if $A$ is real and $A = -A^T$ then $A$ is anti-symmetric.
- **Orthogonal** $A^TA = AA^T = I$ - so that $A^T$ is the inverse of $A$.
- **Unitary** For $A$ complex $A^HA = AA^H = I$.
- **Normal** $AA^H = A^HA$.
- **Hermitian** or selfadjoint if for $A^T = \bar{A}$, so $A^H = A$, where $A$ is complex.

Suppose that $A$ is complex and Hermitian then $\text{diag}(A^H) = \text{diag}(A)$, but $\text{diag}(A^H)_{ij} = (\bar{a}_{ii})$ and $\text{diag}(A)_{ij} = a_{ii}$. Thus the diagonal entries are real.

1.0.1 Some important functions for square $A$

**Trace** of matrix $A$ is the sum of the diagonal entries

$$\text{tr}(A) = \sum_{i=1}^{n} a_{ii}.$$  

**Determinant** See the formulae in the book, But note some identities

- $(i)$ $\det(A) = \det(A^T)$
- $(ii)$ $\det(AB) = \det(A)\det(B)$
- $(iii)$ $\det(A^{-1}) = (\det(A))^{-1}$
- $(iv)$ $\det(A^H) = \det(A)$

If two rows or columns of a matrix coincide then $\det(A) = 0$ and the matrix is singular. Exchanging the order of rows or columns changes the sign in the determinant. Each change introduces a change in sign.
1.0.2 Eigenvalues

For matrix $A$ its characteristic equation is

$$p_A(\lambda) = \det(A - \lambda I) = 0$$

where $p_A$ is a polynomial in $\lambda$ of degree $n$ for matrix $A$ of size $n$ and the roots define the eigenvalues of $A$.

$\lambda \in \mathbb{C}$ is an eigenvalue of $A$ if there is an $x \neq 0 \in \mathbb{C}^n$ such that

$$Ax = \lambda x.$$

$(\lambda, x)$ is an eigenvalue - eigenvector pair for matrix $A$.

The set of $\lambda$ is called the spectrum of $A$ denoted by $\sigma(A)$ and we have

$$Ax = \lambda x \quad \text{x is a right eigenvector}$$

$$y^H A = \lambda y^H \quad \text{y is a left eigenvector}$$

The Rayleigh quotient is

$$\lambda = \frac{x^H Ax}{x^H x}$$

Two properties of importance are

(i) $\det(A) = \prod_{i=1}^{n} \lambda_i$ (ii) $\text{tr}(A) = \sum_{i=1}^{n} \lambda_i$

Thus consider finding the eigenvalues of $A^T$. We must have the characteristic equation

$$p_{A^T}(\lambda) = \det(A^T - \lambda I) = 0,$$

and by the properties of the determinant

$$p_{A^T}(\lambda) = \det\left((A^T - \lambda I)^T\right) = \det(A - \lambda I)$$

because $I$ is the diagonal identity matrix. Hence $\sigma(A) = \sigma(A^T)$ and the eigenvalues of $A$ and its transpose are the same.

1.1 Calculating the eigenvalues of a small matrix using Matlab

We now look at an example of calculating the eigenvalues of a small matrix of size $7 \times 7$.

$$A = \begin{bmatrix}
289 & 2064 & 336 & 128 & 80 & 32 & 16 \\
1152 & 30 & 1312 & 512 & 288 & 128 & 32 \\
-29 & -2000 & 756 & 384 & 1008 & 224 & 48 \\
512 & 128 & 640 & 0 & 640 & 512 & 128 \\
1053 & 2256 & -504 & -384 & -756 & 800 & 208 \\
-287 & -16 & 1712 & -128 & 1968 & -30 & 2032 \\
-2176 & -287 & -1565 & -512 & -541 & -1152 & -289 \\
\end{bmatrix}$$

It is real, we also calculate the eigenvalues of $A^T$.
1.2 Script to illustrate the sensitivity to computer arithmetic

First calculating with standard precision in Matlab.

\[ \text{tr}(A), \text{sum}(\lambda), \text{sum}(\lambda^T), \text{digits} \]

\[ 0 \ -0.000000000000245 \ 0.000000000000557 \ 32.0000 \]

It appears that the sum of the eigenvalues is close to zero for both \( A \) and \( A^T \) but that the eigenvalues are complex, see Figure 1.

Now we reduce the precision but use digits = 16 : observe the trace and it agrees with the sum of the eigenvalues. The middle figure in Figure 1 suggests the eigenvalues are real and now the eigenvalues of \( A \) and \( A^T \) agree.

\[ \text{tr}(A), \text{sum}(\lambda), \text{sum}(\lambda^T), \text{digits} \]

\[ = [0, 1.4693679385278593849609206715278e-38, -2.2040519077917890774413810072917e-38, 16] \]

Now we increase the precision and use digits = 64 : observe the trace and it agrees with the sum of the eigenvalues. The final figure in Figure 1 confirms the eigenvalues are real and the eigenvalues of \( A \) and \( A^T \) agree.

\[ \text{tr}(A), \text{sum}(\lambda), \text{sum}(\lambda^T), \text{digits} \]

\[ = [0, 1.4693679385278593849609206715278e-38, -2.2040519077917890774413810072917e-38, 64] \]

Figure 1: Eigenvalues of the matrices \( A \) and \( A^T \) calculated with standard precision in MATLAB - command \texttt{eig}(A), then with lower precision but symbolic calculation and then with high precision and symbolic.
1.3 Similarity Transformations

Matrices $A$ and $L^{-1}AL$ are similar and $A \rightarrow L^{-1}AL$ is a similarity transformation. Suppose that $(\lambda, \mathbf{x})$ is an eigenvalue-eigenvector pair for matrix $A$ then

$$A\mathbf{x} = \lambda \mathbf{x} \quad L^{-1}A\mathbf{x} = \lambda (L^{-1}\mathbf{x})$$
$$L^{-1}A(LL^{-1})\mathbf{x} = \lambda (L^{-1}\mathbf{x}) \quad (L^{-1}AL)L^{-1}\mathbf{x} = \lambda (L^{-1}\mathbf{x})$$

and we see that $(\lambda, L^{-1}\mathbf{x})$ is an eigenvalue-eigenvector pair for matrix $L^{-1}AL$. i.e. Similar matrices have the same set of eigenvalues.

$$\sigma(A) = \sigma(L^{-1}AL).$$

1.4 Returning to the example

In our example we find eigenvalues of $A$ and $A^T$. $A$ is real and the eigenvalues should be the same and sum to the diagonal.

Consider

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

$$B = L*A*\text{inv}(L)$$

$$\begin{bmatrix}
1 & 2048 & 256 & 128 & 64 & 32 & 16 \\
0 & -2 & 1024 & 512 & 256 & 128 & 32 \\
0 & 0 & 4 & 512 & 1024 & 256 & 64 \\
0 & 0 & 0 & 512 & 512 & 128 & \\
0 & 0 & 0 & 0 & -4 & 1024 & 256 \\
0 & 0 & 0 & 0 & 0 & 2 & 2048 \\
0 & 0 & 0 & 0 & 0 & 0 & -1
\end{bmatrix}$$

We see immediately that the eigenvalues of $B$ are real

Eigenvalues

$$\begin{bmatrix}
1 & -2 & 4 & 0 & -4 & 2 & -1
\end{bmatrix}$$
and forming

\[ A = L^{-1}BL, \]

we find \( A \) and \( B \) are similar. Hence \( A \) should have the same set of real eigenvalues, and not the complex eigenvalues we find using standard Matlab precision. We see that the Matlab results are clearly impacted by precision of the algorithm, but when using symbolic algebra we can obtain the correct results.

### 1.5 Conclusions

(i) In study of numerical algorithms we need to understand their sensitivity to floating point implementations.

(ii) We need to determine ways to assess correctness of solutions.

(iii) We need to understand the underlying physical or mathematical model.

First step — understand floating point arithmetic and how it contributes to propagation of errors, and the condition of a problem.
Floating point numbers are represented in the form
\[ x = f \beta^e, \]
where \( f \) is the fraction or mantissa, \( \beta \) is the base and \( e \) is the exponent. The number \( x \) is said to be normalized if \( \beta > 1 \) and \( 1/\beta \leq f < 1 \). In this case we have
\[ f = 0.x_1x_2...x_t, \quad x_1 \neq 0, \quad \& \quad 0 \leq x_i \leq \beta - 1, i \neq 1. \]

In decimal computations we use \( \beta = 10 \) but on most computers \( \beta = 2 \) or 16. We shall emphasize \( \beta = 2 \), binary arithmetic. Note also that zero is also a floating point number, given by both mantissa and exponent set to zero.

### 2.1 The set of numbers for a Given System

The distribution of the numbers is not uniform across the set of floating point numbers given by a given allocation of number of bits to mantissa and exponent. Consider for example the word with 2 bits for the mantissa, giving effectively 3 bit mantissas \( .x_1x_2x_3 \) for which the set number of numbers is
\[ \{(0.100)_2, (0.101)_2, (0.110)_2, (0.111)_2\} = \{1/2, 5/8, 3/4, 7/8\}. \]

To determine the complete set of representable numbers we must scale by the base raised to the available exponents. If there are 3 bits for the exponent, one of which is for sign, then \( e \) takes on the values \( 0, 1, 2, 3, -1, -2, -3 \) and the set of positive floating point numbers is
\[ \{1/16, 5/64, 6/64', 7/64', 1, 5/6, 6/6, 7/6, 1, 5, 6, 7/4, 1, 5/2, 6/2, 7/2, 3/4, 4, 5, 6, 7\}. \]

We also have to include the number 0 in this set of floating point numbers which is a special case in which \( x_1 = 0 \). Observe that the reals are more densely represented at the low end in magnitude than at the upper end.

### 2.2 The General Case

A floating point number is stored as a floating point word. For example a word of length 32 uses 1 bit for the sign of the number, 8 bits for the exponent and 23 bits for the mantissa. Effectively, with normalization \( x_1 \neq 0 \) the mantissa has 24 bits available because \( x_1 \) does not need to be stored. This is the usual wordlength for a single precision number. Note if we have number \( x = 2^e(\cdot x_1x_2\cdots x_t) \) then \( x = 2^e(\cdot 1 + 2^e(\cdot 0x_1x_2\cdots x_t)) = 2^{e-1}(1+f) \). Hence we store \( f \). The precision is determined by number of bits in \( f = .x_1x_2\cdots x_t \) and the number of bits for representing \( e \) determines the range of representable numbers.
Thus \( e \) ranges from 0 to \( 2^8 - 1 = 256 - 1 = 255 \). Hence the smallest number in magnitude has \( f = 0 \) giving

\[
x_{\text{min}} = 2^{-127}.
\]

(1)

The largest number is obtained for the largest \( f \). For \( t \) bits we have \( f = .1 \cdots 1 \) which is equivalent to

\[
f = \sum_{i=1}^{t} \frac{1}{2^i} = \frac{1}{2} \frac{1 - \frac{1}{2^t}}{1 - \frac{1}{2}} = 1 - \frac{1}{2^t}.
\]

Thus we have the largest value

\[
x_{\text{max}} = 2^{255-127} \cdot (1 + (1 - 2^{-23})) = 2^{128} (2 - 2^{-23}) \approx 2^{129}.
\]

The error introduced in representing in this notation for a given exponent \( e \) is \((-1)^s 2^e - 127 (2^{-24})\) because the 23rd bit is obtained by rounding up or down from the 24th bit. (or 25th bit if we use the guard digit). Thus maximum relative error is \(2^{-24}\).

The double precision word in IEEE standard uses 64 bits assigned with 11 bits for the exponent and 52 for the mantissa. The largest number is about \(10^{307}\).

\[
\begin{array}{c|c|c}
\text{sign} & 11 & 52 \\
\hline
\text{exponent} & 8 & \text{fraction}
\end{array}
\]

### 2.3 Overflow and Underflow September 12, 2012

We have seen that the set of floating point numbers represented on our computer is necessarily finite in size: the exponent is of restricted magnitude and the fraction is limited to the range \(1/\beta \ldots 1\). If the result of a given calculation generates an exponent lying outside the acceptable range an exponent exception occurs. If the exponent is too large or too small, this is called overflow or underflow, respectively. For example, in decimal arithmetic with only 2 digits for the exponent, \((10^{60})^2\) causes an overflow.

For the single precision example we therefore have

underflow below \(2^{-127}\)

overflow above \(2^{128}\).

The way that an underflow is dealt with can be used to avoid overflow. For example consider the calculation of

\[
c = \sqrt{a^2 + b^2},
\]
where $a = 10^{60}$ and $b = 1$ in the decimal system with only two digits for the exponent. We know that $c \approx a$, but an overflow will occur because of the calculation of $a^2$. On the other hand, if we first scale by $a$, so that

$$c = a \sqrt{1 + \left(\frac{b}{a}\right)^2},$$

the approximate answer is obtained if, whenever an underflow exception occurs, underflows are set to zero. This can be used to design an effective means to calculate $c$ for any set of $a, b$. First let $s = \max\{|a|, |b|\}$, then

$$c = s \sqrt{(\frac{a}{s})^2 + (\frac{b}{s})^2},$$

and rather than overflow occurring an underflow may occur and the value be set to zero. (Note that if both $a$ and $b$ are of similar magnitude the issue of underflow or overflow will be mute.)

### 2.4 Machine Epsilon or Unit Roundoff

Machine epsilon can be calculated by testing for the value of $\epsilon$ for which $1 + \epsilon = 1$ with the given floating point number system. **Hence machine epsilon, or the unit roundoff, $\epsilon_M$ is defined to be the distance of 1 to the next nearest floating point number.** For $t$ bits in the mantissa the number closest to 1 = .1...0 $\times \beta$ is given by .1000...1 $\times \beta$, where the trailing 1 is in the $t^{th}$ position. Therefore

$$\epsilon_M = \frac{1}{\beta^t} \times \beta = \beta^{1-t}.$$

In some systems we have an additional bit for calculations, a **guard** bit. Then we can float to a mantissa with a larger number of significant bits for a calculation. In this case we would obtain one more degree of precision for the machine epsilon yielding

$$\epsilon_M = \frac{1}{\beta^{t+1}} \times \beta = \beta^{-t}.$$

Note that the calculation of the number of significant digits in a calculation proceeds much in the same way as the calculation of $\epsilon_M$. Specifically, to obtain a certain number of significant bits, we have to assume that the first bit is non-zero, according to the normalization

$$\cdot 1 \times \beta^e,$$

for some $e$, as in (1). Then we determine the smallest number that we can add, which, as above, will be $\frac{1}{\beta^t} \times \beta^e$. Significance then depends on the mantissa, and specifically, in base 2, on

$$\frac{1}{2^t} = \frac{1}{2^{23}} = 1.2 \times 10^{-7},$$

from which we conclude an accuracy of 6 – 7 significant digits.
2.5 Calculation Of Machine Epsilon:

A simple matlab code can be used to find the machine precision. For example the following should find $p$ such that $1 + 2^p = 1$, and provide for you the calculated machine epsilon, calceps, and compare with the built-in machine epsilon for matlab:

```
x=1; p=0; y=1; z=x+y;
while x~=z
    y=y/2; p=p+1; z=x+y;
end
p, calceps=1/2^(p-1), eps
```

Note that in the above ; indicates that the entry should not be displayed, otherwise it is displayed.

2.6 Rounding Error

The set of floating point numbers is finite. Therefore any given number is subject to rounding error. Suppose that we work in decimal rounded to $t$ digits and the floating point representation of $x$ is denoted by $fl(x)$. Then for $x = .XXXX \ldots XY \times 10^e$,

$$fl(x) = \begin{cases} .XXXX \ldots X \times 10^e, & Y < 5 \\ .XXXX \ldots Z \times 10^e, & Y \geq 5, \quad Z = X + 1. \end{cases}$$

Here $Y$ is in the $(t+1)^{st}$ position after the decimal point and

$$|x - fl(x)| < 5 \times 10^{-(t+1)+e} = \frac{1}{2} 10^{-t+e}.$$

Moreover, assuming normalization so that the first digit is non zero,

$$|x| > 10^{e-1}, \quad \frac{1}{|x|} < 10^{1-e}$$

From which the relative error is also bounded

$$\frac{|x - fl(x)|}{|x|} < \frac{1}{2} 10^{-t+e} 10^{1-e} = \frac{1}{2} 10^{-t+1}.$$

On the other hand, if we apply chopping

$$|x - fl(x)| < 9 \times 10^{-(t+1)+e} = 10^{-t+e},$$

and

$$\frac{|x - fl(x)|}{|x|} < 10^{-t+1}.$$

The effect on the relative error of chopping is thus twice that of rounding. In binary arithmetic the same idea can be used to see that

$$\frac{|x - fl(x)|}{|x|} \leq \begin{cases} 2^{-t} & \text{rounding} \\ 2^{-t+1} & \text{chopping}. \end{cases} \quad (2)$$

Now observe that the relative error in (2) is also bounded by $\beta^{-t+1}$. Thus if $(fl(x) - x)/x = \epsilon$, then $fl(x) = x(1 + \epsilon)$ where $|\epsilon| < |\epsilon_M|$ and $\epsilon_M$ is an upper bound on the relative error.
2.7 Example: Decimals

Consider the expression of \(\frac{1}{10} = 0.1\) as a binary fraction. To be expressed as a binary fraction we have to be able to express \(\frac{1}{10}\) as a sum of terms \(\frac{1}{2^k}, k \geq 1\). Notice first

\[
\frac{1}{16} = \frac{1}{2^4} < \frac{1}{10} < \frac{1}{8} = \frac{1}{2^3}
\]

\[
\frac{1}{16} + \frac{1}{32} = \frac{3}{32} < \frac{1}{10} < \frac{7}{64} = \frac{1}{16} + \frac{1}{32} + \frac{1}{64},
\]

which implies at least

\[
\frac{1}{10} = \frac{0}{2} + \frac{0}{4} + \frac{0}{8} + \frac{1}{16} + \frac{1}{32} + \frac{0}{64} + \ldots
\]

If you continue, you find the next terms

\[
\frac{1}{10} = \frac{1}{2^4} + \frac{1}{2^5} + \frac{0}{2^6} + \frac{0}{2^7} + \frac{1}{2^8} + \frac{1}{2^9} + \frac{0}{2^{10}} + \frac{0}{2^{11}} + \frac{1}{2^{12}} \ldots
\]

which suggests the pattern \((.0001100110011...)_2\), and that we can collect terms

\[
\frac{1}{10} = \frac{1}{2^4} + \left( \frac{1}{2^5} + \frac{1}{2^8} \right) + \left( \frac{1}{2^9} + \frac{1}{2^{12}} \right) + \ldots
\]

\[
= \frac{1}{16} \left( 1 + \frac{9}{16} + \frac{9}{16^2} + \ldots \right)
\]

Now we prove that in fact 0.1 is actually the limit of the infinite series just given:

\[
\frac{1}{16} \left( 1 + \frac{9}{16} + \frac{9}{16^2} + \ldots \right) = \frac{1}{16} \left( 1 + \frac{9}{16} \left( 1 + \frac{1}{16} + \frac{1}{16^2} + \ldots \right) \right)
\]

\[
= \frac{1}{16} \left( 1 + \frac{9}{16} \left( \sum_{j=0}^{\infty} \frac{1}{16^j} \right) \right)
\]

Now for a geometric series

\[
\sum_{j=0}^{n} r^j = \frac{1 - r^{n+1}}{1 - r}
\]

and for \(r < 1\) we have the limit

\[
\lim_{n \to \infty} \sum_{j=0}^{n} r^j = \frac{1}{1 - r}.
\]

We can apply this result to obtain

\[
\lim_{n \to \infty} \frac{1}{16} \left( 1 + \frac{9}{16} \left( \sum_{j=0}^{\infty} \frac{1}{16^j} \right) \right) = \frac{1}{2^4} \left( 1 + \frac{9}{16} \left( 1 - \frac{1}{16} \right) \right)
\]

\[
= \frac{1}{16} \left( 1 + \frac{9}{16} \frac{1}{16} \right)
\]

\[
= \frac{1}{10}.
\]

and deduce that relative to binary fractions, the decimal .1 is expressible as the limit of an infinite binary sum, and thus can not be represented exactly in binary arithmetic.
2.8 Floating point operations

Let $\circ$ be one of the $\times + - \div$ operations. We calculate exactly $x \circ y$. In floating point we obtain $fl(x \circ y)$ which is floating point representation of $x \circ y$, and

$$fl(x \circ y) - x \circ y = \text{rounding error}.$$ 

If $fl(x \circ y)$ is the nearest floating point number to $x \circ y$, then we say that the arithmetic rounds correctly, and we can write $fl(x \circ y) = (x \circ y)(1 + \delta)$, $|\delta| < \epsilon$ where $\epsilon$ is the machine epsilon. In IEEE standard we also obtain

$$fl(\sqrt{x}) = \sqrt{x}(1 + \delta).$$

2.9 Guard Digits (Round Digit in the Book)

Ideally an operation on a pair of floating point numbers would yield the floating point representation of the operation on the pair of numbers:

$$fl(x) \circ fl(y) = fl(x \circ y),$$

where here $\circ$ denotes the operations of $+,-,\times,$ and $\div$. But $fl(x \circ y) = (x \circ y)(1 + \epsilon)$ and ideally we therefore want

$$fl(x) \circ fl(y) = (x \circ y)(1 + \epsilon).$$

Is this always the case? Consider again a decimal system, this time with 5 significant digits. Then,

$$fl(1 - .99999) = fl(1) - fl(.99999)$$

$$= .1 \times 10 - .99999 \times 10^0 \quad \text{which floats to}$$

$$= .1 \times 10 - .099999 \times 10$$

$$= .00001 \times 10$$

$$= 10^{-4},$$

whereas in exact arithmetic, we have $10^{-5}$. The ratio $(fl(x) \circ fl(y))/(x \circ y) = 10$ is not at all small as desired. To improve the result a guard digit can be used and we float to one more digit in the calculation to obtain the correct result

$$fl(1 - .99999) = fl(1) - fl(.99999)$$

$$= .1 \times 10 - .99999 \times 10^0 \quad \text{which floats to}$$

$$= .1 \times 10 - .099999 \times 10$$

$$= .000001 \times 10$$

$$= 10^{-5}.$$ 

The guard / round digit is used to provide one extra significant figure for the purposes of computation but is not used for the storage of the results of computations.
2.10 Cancellation or Loss of Significance

Consider now the calculation of \( f(x) = (1 - \cos x)/x^2 \) for small \( x \), \( x = 1.2 \times 10^{-5} \). To 10 significant figures we find that \( y = \cos x = .9999999999 \) and \( 1 - y = .0000000001 \). Therefore we obtain

\[
f(x) = \frac{1 - \cos x}{x^2} = \frac{10^{-10}}{1.44 \times 10^{-10}} = .6944 \ldots.
\]

Does this seem reasonable? For \( x \) small we know that the Taylor series expansion of \( \cos x \) is given by \( \cos x = 1 - x^2/2 + O(x^4) \) so that \( 1 - \cos x \approx x^2/2 + O(x^4) \), where the trailing terms are increasingly smaller in size so that

\[
f(x) \approx \frac{x^2/2 - O(x^4)}{x^2} < \frac{1}{2}.
\]

Clearly the computed answer was not even correct in one significant figure. \( 1 - y \) has only 1 significant figure remaining, even though \( y \) was calculated with 10 significant figures. The importance of the **loss of significant figures in the subtraction** is elevated in the calculation of \( f(x) \) in this way. On the other hand, trigonometric identities can be used to our advantage to provide an effective means for the calculation of \( f(x) \). Using \( \cos x = 1 - 2\sin^2 x/2 \) we obtain

\[
f(x) = \frac{1}{2} \left( \frac{\sin x/2}{x/2} \right)^2,
\]

which for the same value of \( x \) gives .5, accurate to 10 significant figures.

The above example can very easily be explained. Consider \( fl(x) = x(1 + \epsilon_x) \) and \( fl(y) = y(1 + \epsilon_y) \). Then \( fl(x) - fl(y) = (x - y) + x\epsilon_x - y\epsilon_y \) and

\[
\frac{x - y - (fl(x) - fl(y))}{x - y} = \frac{y\epsilon_y - x\epsilon_x}{x - y}.
\]

But \( \epsilon_x \) and \( \epsilon_y \) are bounded by \( \epsilon_M \) and thus

\[
\frac{x - y - (fl(x) - fl(y))}{x - y} \leq \epsilon_M \frac{|1.2 \times 10^{-5}| + |.9999999999|}{|10^{-9}|} \approx \frac{10^{-9}}{10^{-9}} = 1,
\]

where we note the machine epsilon for the given example. In general if

\[
|x - y| << |x| + |y|
\]

the error can be substantially magnified: i.e. subtraction magnifies earlier errors.

2.11 Example: Roots of a Quadratic

Consider the determination of the roots of the quadratic polynomial

\[
x^2 - bx + c = 0.
\]
The roots are given by

\[ x_\pm = \frac{b \pm \sqrt{b^2 - 4c}}{2}, \]

which for \( b = 3.6778 \) and \( c = .0020798 \) are

\[ x_+ = 3.6772344190 \] and \( x_- = .00056558809, \]

calculated in exact arithmetic. Calculated to 5 significant figures \( x_- = .00055 \) is obtained, accurate to only 1 significant figure. The problem occurs in the evaluation of \( b^2 - 4c = 13.526 - .008 \). In effect the value of \( c \) that is used is \( c = .002 \), and critical information about the polynomial is lost. If instead we observe that \( x_+x_- = c \) and that \( x_- = c/x_+ = .0056558 \) then a result correct to 5 significant figures is obtained. Again it is important to consider the calculation before computation and take advantage of any alternative formulations.

2.12 Problem Stability

Suppose we want to calculate \( f(x) \) given \( x + \delta x \) where \( \|\delta x\| \) is bounded in terms of machine precision. Then we calculate \( f(x + \delta x) \) in actuality. Hence we want to estimate whether \( f(x + \delta x) \approx f(x) \). By Taylor series \( f(x + \delta x) - f(x) \approx \delta x f'(x) \) assuming sufficient differentiability in \( f(x) \) near \( x \). Hence \( \|f(x + \delta x) - f(x)\| \approx \|\delta x\|\|f'(x)\| \). Thus size of error depends on \( \|f'(x)\| \), which we call “absolute condition number.” If \( \|f'(x)\| \) is large we say “\( f(x) \) is ill-conditioned at \( x \).” Moreover, relative error

\[
\frac{\|f(x + \delta x) - f(x)\|}{\|f(x)\|} \approx \frac{\|\delta x\|\|f'(x)\|}{\|f(x)\|} = \left( \frac{\|\delta x\|}{\|x\|} \right) \left( \frac{\|f'(x)\|}{\|f(x)\|} \right).
\]

Thus this measures relative error of the calculation in terms of the relative error in \( x \), and \( K(x) = \frac{\|f'(x)\|\|x\|}{\|f(x)\|} \) is the “relative condition number.”

**Condition number** gives information on how the input error (error in \( x \)) affects output error, (error in \( f(x) \)). The condition number is not large, the solution is stable, otherwise it is not stable.

**Well posed** We say a problem is well-posed (stable) if it admits a unique solution that depends continuously on the data. Solutions are stable to perturbations in the data.

**Ill-posed** We say a problem is ill-posed if it is not well-posed. Solutions are not stable to changes in the data.

**Instability** is intrinsic, it cannot always be avoided by designing a numerical method, although some algorithms may be better suited than others to handle a particular instability.
2.13 Returning to the Quadratic Problem

Consider the case
\[ x^2 - 2px + 1 = 0 \]
\[ x_{\pm} = \frac{p \pm \sqrt{4p^2 - 4}}{2} = p \pm \sqrt{p^2 - 1}, \]
for which the solution is \( x = G(p) \), \( G(p) = \{x_+, x_-\} \). To find the condition we consider
\[ G'(p) = 1 \pm \frac{p}{\sqrt{p^2 - 1}}. \]
Thus
\[ K(p) = \frac{\|G'(p)\|\|p\|}{\|G(p)\|} = \frac{\|\sqrt{p^2 - 1} \pm p\|\|p\|}{\sqrt{p^2 - 1}\|p \pm \sqrt{p^2 - 1}\|} = \frac{\|p\|}{\|\sqrt{p^2 - 1}\|}, \quad p > 1 \]
and \( K(p) > 1 \). The size of \( K(p) \) will depend on the relative sizes of numerator and denominator. For example take \( p = \sqrt{2} \) then \( K(p) \approx \sqrt{2} \) which is reasonable and the problem is well-conditioned. But if \( p = 1.00001 \) then \( K(p) \approx 224 \) which is rather large. i.e. the problem is ill-conditioned for \( p \approx 1 \) but well-conditioned away from 1.

Can we deal with the ill-conditioning for this problem? Introduce the parameter \( t = p + \sqrt{p^2 - 1} \), then \( x_+ = t \) and \( 1/t = x_- \), and we obtain the desired solution. Moreover, calculating \( K(t) \) we find \( K(t) = 1 \) so that with this formulation the problem is well-conditioned. Hence the problem itself is not ill-posed.

2.14 Another simple example illustrating stability issues

Consider the solution of the linear system \( Ax = b \) with
\[ A = \begin{pmatrix} .780 & .563 \\ .913 & .659 \end{pmatrix} \quad b = \begin{pmatrix} .217 \\ .254 \end{pmatrix} \quad \text{solution} \quad x = \begin{pmatrix} 1 \\ -1 \end{pmatrix} \]
Suppose we solve base 10 and with precision 3 and we are told that a solution is given by
\[ \bar{x} = \begin{pmatrix} 0.999 \\ -0.999 \end{pmatrix} \]
Then we can calculate how well \( \bar{x} \) solves the set of equations by finding the residual
\[ \bar{r} = b - A\bar{x} = \begin{pmatrix} .000217 \\ .000254 \end{pmatrix}, \]
which appears to small, suggesting that \( \bar{x} \) is a solution. Indeed the difference measured relatively in the 2-norm is given by

\[
\frac{\|x - \bar{x}\|}{\|x\|} = \frac{\|[0.001, -0.001]\|}{\sqrt{2}} = \frac{0.002}{\sqrt{2}} = 10^{-3}.
\]

This raises the question **Does a small residual mean that the solution is a good solution of the system of equations?** We now consider another candidate solution

\[
\hat{x} = \begin{pmatrix} 0.341 \\ -0.0870 \end{pmatrix}
\]

for which the residual in this precision is

\[
\hat{r} = b - A\hat{x} = \begin{pmatrix} .000001 \\ 0 \end{pmatrix},
\]

and

\[
\frac{\|x - \hat{x}\|}{\|x\|} = \frac{\|[0.999999, -1]\|}{\sqrt{2}} = \frac{1.414212855266491}{\sqrt{2}} = .9999995 \approx 1.
\]

Thus the solution \( \hat{x} \) generates a small residual \( \hat{r} \) but is not a good solution of the problem. **In general we do not know the true solution and cannot work out the relative errors to determine whether the solution is reasonable or not, we can evaluate the residual but that does not provide sufficient information to determine whether the solution is acceptable.**

Now consider an alternative formulation of the problem in which we keep the right hand side fixed and perturb the matrix \( A \) by an amount \( E \)

\[
A + E = \begin{pmatrix} .781 & .564 \\ .914 & .658 \end{pmatrix}, \quad E = .001 \begin{pmatrix} 1 & 1 \\ 1 & -1 \end{pmatrix}
\]

and solve in Matlab precision, we obtain

\[
\tilde{x} = \begin{pmatrix} 0.294117647058827 \\ -0.022528160200255 \end{pmatrix} \quad \frac{\|x - \tilde{x}\|}{\|x\|} = \frac{\|[0.706 - 0.977]\|}{\sqrt{2}} \approx .853
\]

so that in this case the small perturbation in \( A \) generated a large relative error in the solution of the system of equations. Equivalently the perturbed system has a very different solution. In both cases the matrices \( A \) and \( A + E \) are highly ill-conditioned

\[
\text{cond}(A) = 2.193e + 06 \quad \text{cond}(A + E) = 1.374e + 03
\]

where for matrices the condition measured in the two norm is given by the ratio of the maximum to minimum singular values of the matrix. (to be discussed later).
2.15 Algorithm stability

Given an algorithm for calculating $f(x)$ we want to know whether the calculated value from the algorithm, say $\text{alg}(x)$ gives the exact answer $f(x + \delta x)$ for some small $\delta x$ i.e. did we solve a nearby problem? If $\forall x$ there exists $\delta x$ sufficiently small such that $\text{alg}(x) = f(x + \delta x)$ we say the algorithm is backward stable and $\delta x$ is the backward error. In the case of a backward stable algorithm we can bound the error by

$$\|\text{alg}(x) - f(x)\| = \|f(x + \delta x) - f(x)\| \approx \|f'(x)\|\|\delta x\|$$

and thus the error is small provided $\|f'(x)\|$ is not large, because necessarily $\|\delta x\|$ is small. Proof of backward stability requires understanding of propagation of floating point errors through the algorithm. We hope our algorithms are backward stable, otherwise $\text{alg}(x)$ may not be the solution of a nearby problem.

On the other hand, forward error analysis seeks to bound the error obtained in a given estimate of $f(x)$, in terms of the perturbations in the data and due to errors that accumulate in the numerical method. Practically we obtain not $f(x)$, but an estimate $\tilde{f}(x)$ and want to estimate the size of $f - \tilde{f}$. 