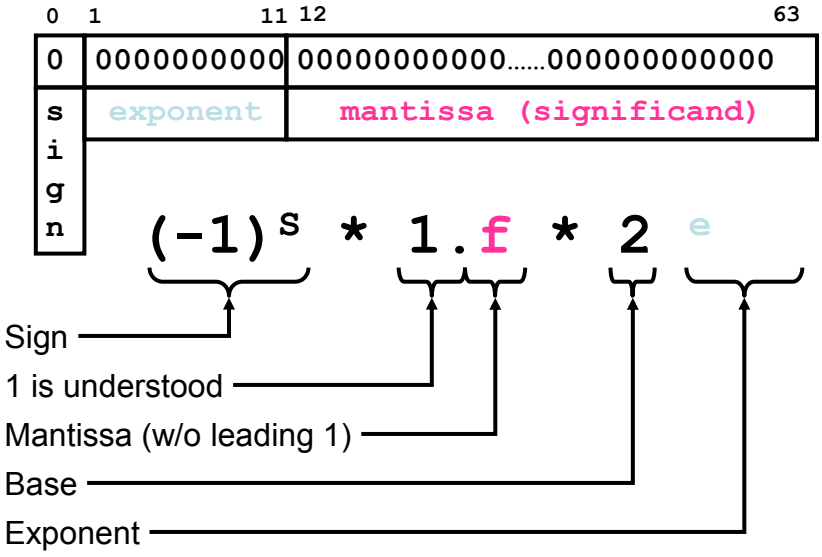
IEEE-754 (double precision) – 64 bits



Effects of floating point

- $x = \pm(1+f) \times 2^e$
- $0 \leq f < 1$
- $f = (\text{integer} < 2^{52}) / 2^{52}$
- $-1022 \leq e \leq 1023$
- $e = \text{integer}$

Finite f implies finite *precision*.

Finite e implies finite *range*

Floating point numbers have discrete spacing, a maximum and a minimum.

Special floating point numbers

- eps *is the distance from* 1 *to the next larger floating-point number.*
- eps = 2^{-52}
- In Matlab

	Binary	Decimal
`eps`	`2^(-52)`	`2.2204e-16`
`realmin`	`2^(-1022)`	`2.2251e-308`
`realmax`	`(2-eps)*2^1023`	`1.7977e+308`

Floating point exceptions

- Underflow
- Overflow
- Division by zero
- NaN: 0/0, ∞×0, or sqrt(−number).

Special (Exceptional) Numbers

0	00000000000	000000000000.....000000000000000
s i g n	exponent	mantissa (significand)

$$(-1)^s * 2^E * 1.f$$

	E+1023 == 0	0 < E+1023 < 2047	E+1023 == 2047
f==0	0	Powers of Two	∞
f~=0	Non-normalized typically underflow	Floating point Numbers	Not A Number

Three Loops

```
x = 1; while 1+x > 1, x = x/2, pause(.02), end

x = 1; while x+x > x, x = 2*x, pause(.02), end

x = 1; while x+x > x, x = x/2, pause(.02), end
```

- Loop 1 executes 52 times -- > corresponds to mantissa size
- Loop 2 executes 1024 times … effectively increasing exponent till it reaches inf
- Loop 3 executes 1075 times

Effects of floating point errors

- Singular equations will only be nearly singular
- Severe cancellation errors can occur

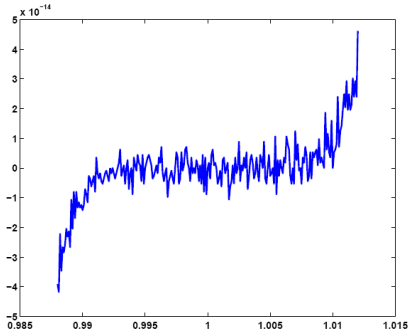
$$17x_1 + 5x_2 = 22$$
$$1.7x_1 + 0.5x_2 = 2.2$$

```
A = [17 5; 1.7 0.5]
b = [22; 2.2]
x = A\b
```

produce

```
x =
  -1.0588
   8.0000
```

```
x = 0.988:.0001:1.012;
y = x.^7-7*x.^6+21*x.^5-35*x.^4+35*x.^3-21*x.^2+7*x-1;
plot(x,y)
```



Error Analysis

- Computations should be as accurate and as error-free as possible
- Sources of error:
  - Poor models of a physical situation
  - Ill-posed problems
  - Errors due to representation of numbers on a computer and successive operations with these
- How do we analyze an algorithm for correctness?
  - Forward error analysis
  - Backward error analysis
- Well posedness

Error definition

- Computation should have result: c
- Actual result is: x
- Absolute error= $|x - c|$
- Relative error = $|x-c|/|x|$ for $x \neq 0$
- $x = (c) \times (1+\text{rel\_err})$

- Usually we do not know c
  - No need to solve the problem if we already knew it!
- Error analysis tries to estimate *abs_err* and *rel_err* using computed result x and knowledge of the algorithm and data

Errors due to round off in addition

- Errors can be magnified during computation.
- Example: 2.003×10^0 (suppose ± .001 or .05% error)
  $- 2.000 \times 10^0$ (suppose ± .001 or .05% error)
- Result of subtraction: 0.003×10^0
- but true answer could be as small as 2.002 - 2.001 = 0.001, or as large as 2.004 - 1.999 = 0.005!
- So error in the answer is as much as (± .002 or 200% error if true answer is 0.001)
- Called: **Catastrophic cancellation**, or "loss of significance"

Addition:

- We could generalize this example to prove a theorem:
- When **adding or subtracting**, the bounds on **absolute errors** add.

Multiplication/Division

- What if we multiply or divide?
- Suppose x and y are the true values, and X and Y are our approximations to them. If

$$X = x(1 - r) \text{ and } Y = y(1 - s)$$

then r is the relative error in x and s is the relative error in y.

Can show that $\left| \frac{xy - XY}{xy} \right| \leq |r| + |s| + |rs|$

- If r and s are small, then we can ignore |rs| term

Rules of thumb

- Addition/subtraction: Bounds on **absolute errors** add
- Multiplication/Division: Bounds on **relative errors** add
- One way to analyze the algorithm is to assume, this error occurs in each arithmetic operation
- Worst case analysis
- Such error bounds (forward error bounds) are often pessimistic

Modeling

- Original mathematical models may be poorly specified or unavailable
  - E.g. Newton's laws work for non relativistic dynamics
  - Turbulence
  - …
- Computing with a poor model will lead to inevitable errors
- Quantities that are measured may be done so with error and bias
  - Using them in computation will lead to errors

Errors are inevtiable

- Each step is characterized by some error

1. Measurement errors:
2. Errors in properties
3. Inexact mathematical models
4. Discretization errors: something continuous is represented discretely
5. Errors in the solution to discrete representations of numbers

Error Analysis

- Design algorithms to minimize errors
- Estimate errors based on knowledge of algorithm
  - Error Analysis – forward and backward

- Two primary techniques of error analysis
  - Forward Error Analysis
    - Floating-point representation of the error is subjected to the same mathematical operations as the data itself.
      - Equation for the error itself
  - Backward Error Analysis
    - Attempt to regenerate the original mathematical problem from previously computed solutions
      - Minimizes error generation and propagation

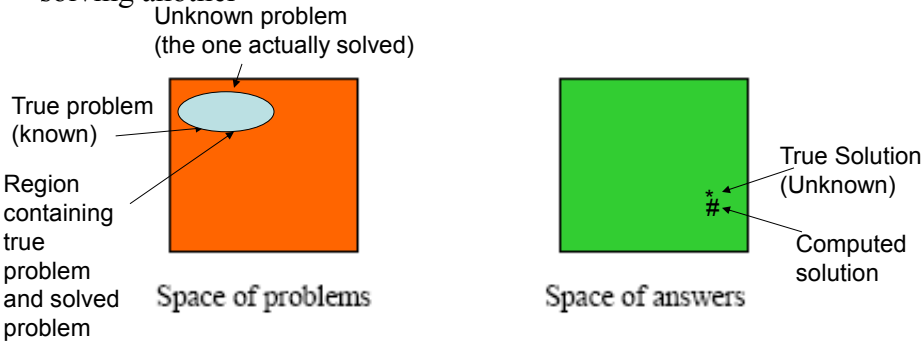
Error Analysis

- Forward and Backward error analysis
- Forward error analysis
  - Assume that the problem we are solving is exactly specified
  - Produce an approximate answer using the algorithm considered

  - Goal of forward error analysis produce region guaranteed to contain true soln.
  - Report region and computed solution

Backward error analysis

- We know that our problem specification itself has error ("error in initial data")
- So while we think we are solving one problem we are actually solving another



- Given an answer, determine how close the problem actually solved is to the given problem.
- Report solution and input region

Testing for Error Propagation

- Use the computed solution in the original problem
- Use Double or Extended Precision rather than Single Precision
- Rerun the problem with slightly modified (incorrect) data and look at the results
  - Sensitivity analysis
  - If problem result changes a lot, perhaps the result is incorrect

Well posed problems

- Hadamard postulated that for a problem to be "well posed"
  1. *Solution must exist*
  2. *It must be unique*
  3. *Small changes to input data should cause small changes to solution*
- Many problems in science result in "ill-posed" problems.
  - Numerically it is common to have condition 3 violated.
  - Problems which violate 3 are also called "ill-conditioned"
- Converting ill-posed problem to well-posed one is called *regularization*.
- Will discuss this later in the course when talking about optimization and Singular Value Decompositon.

Linear Algebra

- Much of scientific computation involves modeling n-tuples of numbers
- Assumed to live in a linear vector space
  - Even non-linear problems are solved by linearization
- Some interpretations of matrices and vectors
- Matrix vector multiplication and complexity
- Identity, Inverse, Singular Matrices
- Permutation, Lower and Upper Triangular Matrices

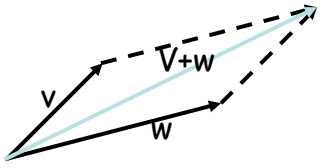
Vectors

- Ordered set of numbers: (1,2,3,4)
- Example: (*x,y,z*) coordinates of a pt in space.
- The 16384 pixels in a 128×128 image of a face
- Vectors usually indicated with bold lower case letters. Scalars with lower case
- Usual operations assumed with vectors:
  - Addition operation $\mathbf{u} + \mathbf{v}$, with:
    - Identity $\mathbf{0}$ $\quad \mathbf{v} + \mathbf{0} = \mathbf{v}$
    - Inverse - $\quad \mathbf{v} + (-\mathbf{v}) = \mathbf{0}$
  - Scalar multiplication:
    - Distributive rule: $\alpha(\mathbf{u} + \mathbf{v}) = \alpha(\mathbf{u}) + \alpha(\mathbf{v})$

      $(\alpha + \beta)\mathbf{u} = \alpha\mathbf{u} + \beta\mathbf{u}$

Vector Addition

$$\mathbf{v} + \mathbf{w} = (x_1, x_2) + (y_1, y_2) = (x_1 + y_1, x_2 + y_2)$$



Vector Spaces

- A *linear combination* of vectors results in a new vector:

$$\mathbf{v} = \alpha_1\mathbf{v}_1 + \alpha_2\mathbf{v}_2 + \ldots + \alpha_n\mathbf{v}_n$$

- If the only set of scalars such that

$$\alpha_1\mathbf{v}_1 + \alpha_2\mathbf{v}_2 + \ldots + \alpha_n\mathbf{v}_n = \mathbf{0}$$

  is $\alpha_1 = \alpha_2 = \ldots = \alpha_3 = \mathbf{0}$

  then we say the vectors are *linearly independent*
- The *dimension* of a space is the greatest number of linearly independent vectors possible in a vector set
- For a vector space of dimension *n*, any set of *n* linearly independent vectors form a *basis*

Vector Spaces: Basis Vectors

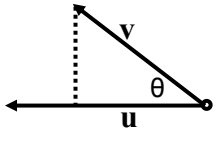
- Given a basis for a vector space:
  - Each vector in the space is a *unique* linear combination of the basis vectors
  - The *coordinates* of a vector are the scalars from this linear combination
  - Best-known example: Cartesian coordinates
    - Example
  - Note that a given vector **v** will have different coordinates for different bases

Dot Product

- The *dot product* or, more generally, *inner product* of two vectors is a scalar:

$$\mathbf{v}_1 \bullet \mathbf{v}_2 = \mathbf{x}_1\mathbf{x}_2 + \mathbf{y}_1\mathbf{y}_2 + \mathbf{z}_1\mathbf{z}_2 \qquad \text{(in 3D)}$$

- Useful for many purposes
  - Computing the length of a vector: $\mathit{length}(\mathbf{v}) = \mathrm{sqrt}(\mathbf{v} \bullet \mathbf{v})$
  - *Normalizing* a vector, making it unit-length
  - Computing the angle between two vectors:
    $\mathbf{u} \bullet \mathbf{v} = |\mathbf{u}|\,|\mathbf{v}| \cos(\theta)$
  - Checking two vectors for orthogonality
  - *Projecting* one vector onto another



Vector norms

$v = (x_1, x_2, \ldots$ n

Two norm (Euclidean norm)

$$\|v\|_2 = \sqrt{\sum_{i=1}^{n} x_i^2}$$

If $\|v\|_2 = 1$, v is a unit vector

Infinity norm

$\|v\|_\infty = \max(|x|_1, |x|_2, \ldots$ n

One norm ("Manhattan distance")

$$\|v\|_2 = \sum_{i=1}^{n} \left| x_i \right|$$

For a 2 dimensional vector, write down the set of vectors with two, one and infinity norm equal to unity

Linear Transformations: Matrices

- A *linear transformation*:
  - Maps one vector to another
  - Preserves linear combinations
- Thus behavior of linear transformation is completely determined by what it does to a basis
- Turns out any linear transform can be represented by a *matrix*