Introduction to Numerical Analysis for Engineers

- Fundamentals of Digital Computing
  - Digital Computer Models
  - Convergence, accuracy and stability
  - Number representation
  - Arithmetic operations
  - Recursion algorithms
- Error Analysis
  - Error propagation – numerical stability
  - Error estimation
  - Error cancellation
  - Condition numbers

Floating Number Representation

$$r = mb^e$$

m	Mantissa
b	Base
e	Exponent

Examples

Decimal $0.00527 = 0.527_{10} \times 10^{-2_{10}}$

Binary $10.1_2 = 0.101_2 \times 2^{2_{10}} = 0.101_2 \times 2^{10_2}$

Convention

Decimal $0.1 \leq m < 1.0$

Binary $0.1_2 = 0.5_{10} \leq m < 1.0$

General $b^{-1} \leq m < b^0$

Max mantissa $0.11\ldots1 = 0.999999$

Min mantissa $0.10\ldots0 = 0.5$

Max exponent $2^7 - 1 = 127 \quad 2^{127} \simeq 1,7 \times 10^{38}$

Min exponent $-2^7 = -128 \quad 2^{-128} \simeq 2.9 \times 10^{-39}$

Error Analysis

Number Representation

Absolute Error

$$\bar{\epsilon} = |\bar{m} - m| \leq \frac{1}{2} b^{-t}$$

Relative Error

$$\bar{\alpha} = \frac{|\bar{m} - m| b^e}{|m| b^e} \leq \frac{\frac{1}{2} b^{-t}}{b^{-1}} \leq \frac{1}{2} b^{1-t}$$

Addition and Subtraction

$$r_1 \pm r_2 = m_1 b^{e_1} \pm m_2 b^{e_2}$$

Shift mantissa of largest number

$$e_1 > e_2$$

Result has exponent of largest number

$$r_1 \pm r_2 = \left(m_1 \pm m_2 b^{e_2 - e_1}\right) b^{e_1} = m b^{e1}$$

Absolute Error

$$\bar{\epsilon} \leq \bar{\epsilon_1} + \bar{\epsilon_2}$$

Relative Error

$$\bar{\alpha} = \frac{|\bar{m} - m|}{|m|}$$

Unbounded

Multiplication and Division

$$r_1 \times r_2 = m_1 m_2 b^{e1+e2}$$

$$m = m1m2 < 1$$

$$0.1_2 \times 0.1_2 = 0.01_2$$

Relative Error

$$\bar{\alpha} \leq \bar{\alpha_1} + \bar{\alpha_2}$$

Bounded

Digital Arithmetics

Finite Mantissa Length

radd.m

```
function c = radd(a,b,n)
%
% function c = radd(a,b,n)
%
% Adds two real numbers a and b simulating an arithmetic unit with
% n significant digits.
%
% First determine sign
sa=sign(a);
sb=sign(b);
if (sa == 0)
    la=-200;
else
la=ceil(log10(sa*a*(1+10^(-(n+1)))));
end
if (sb == 0)
    lb=-200;
else
    lb=ceil(log10(sb*b*(1+10^(-(n+1)))));
end
    lm=max(la,lb);
f=10^(n);
at=sa*round(f*sa*a/10^lm);
bt=sb*round(f*sb*b/10^lm);
ct=at+bt;
sc=sign(ct);
if (sc ~= 0)
if (log10(sc*ct) >= n)
    ct=round(ct/10)*10;
end
end
c=ct*10^lm/f;
```

Limited precision addition in MATLAB

Recursion
Heron's Device

Numerically evaluate square-root

$$\sqrt{s}\ , \quad s > 0$$

Initial guess

$$x_0 \simeq \sqrt{s}$$

Test

$$x_0^2 < s \;\Rightarrow\; x_0 < \sqrt{s} \;\Rightarrow\; \frac{s}{x_0} > \sqrt{s}$$

$$x_0^2 > s \;\Rightarrow\; x_0 > \sqrt{s} \;\Rightarrow\; \frac{s}{x_0} < \sqrt{s}$$

Mean of guess and its reciprocal

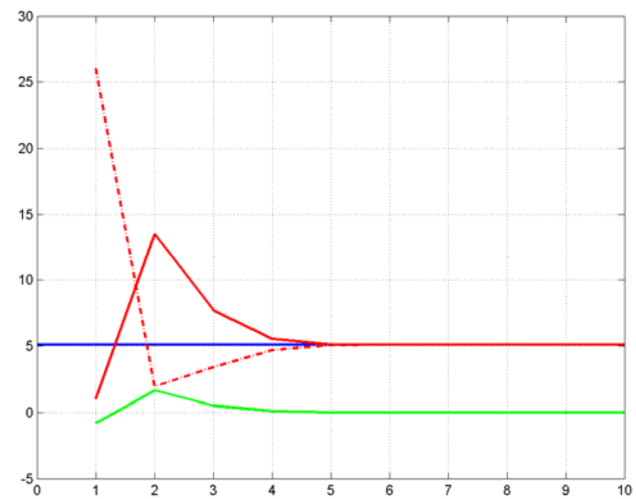
$$x_1 = \frac{1}{2}\left(x_0 + \frac{s}{x_0}\right)$$

Recursion Algorithm

$$x_{n+1} = \frac{1}{2}\left(x_n + \frac{s}{x_n}\right)$$

MATLAB script
heron.m

```
a=26;
n=10;
g=1;
% Number of Digits
dig=5;
     sq(1)=g;
     for i=2:n
       sq(i)= 0.5*radd(sq(i-1),a/sq(i-1),dig);
     end
     hold off
     plot([0 n],[sqrt(a) sqrt(a)],'b')
     hold on
     plot(sq,'r')
     plot(a./sq,'r-.')
     plot((sq-sqrt(a))/sqrt(a),'g')
     grid on
```



Recursion
Horner's Scheme

Evaluate polynomial

$$\begin{aligned} p(z) &= a_0 z^3 + a_1 z^2 + a_2 z + a_3 \\ &= ((a_0 z + a_1) z + a_2)\, z + a_3 \end{aligned}$$

Horner's Scheme

a_0	a_1	a_2	a_3
	zb_0	zb_1	zb_2
b_0	b_1	b_2	b_3

$$p(z) = b_3$$

General order n

$$p(z) = a_0 z^n + a_1 z^{n-1} + \cdots + a_{n-1} z + a_n$$

Recurrence relation

$$b_0 = a_0 \,, \quad b_i = a_i + z b_{i-1} \,, \quad i = 1, \ldots n$$

$$\boxed{p(z) = b_n}$$

horner.m

```
% Horners scheme
% for evaluating polynomials
a=[ 1 2 3 4 5 6 7 8 9 10 ];
n=length(a) -1 ;
z=1;
b=a(1);
% Note index shift for a
for i=1:n
    b=a(i+1)+ z*b;
end
p=b
```

```
>> horner

p =

    55

>>
```

Recursion

Order of Operations Matter

0 1

$$y = f(x) = \sum_{n=1}^{\infty} [x^n + b\sin[\pi/2 - \pi/10n] - c\cos[pi/(10(n+1))]]$$

$$x = 0.5\,,\quad b = 0\,,\quad c = 0 \Rightarrow y = 1.0$$

Result of small, but significant term 'destroyed' by subsequent addition and subtraction of almost equal, large numbers.

Remedy:
Change order of additions

recur.m

```
N=20; sum=0; sumr=0;
b=1; c=1; x=0.5;
xn=1;
% Number of significant digits in computations
dig=2;
ndiv=10;
for i=1:N
a1=sin(pi/2-pi/(ndiv*i));
a2=-cos(pi/(ndiv*(i+1)));
% Full matlab precision
xn=xn*x;
addr=xn+b*a1;
addr=addr+c*a2;
ar(i)=addr;
sumr=sumr+addr;
z(i)=sumr;
% additions with dig significant digits
add=radd(xn,b*a1,dig);
add=radd(add,c*a2,dig);
% add=radd(b*a1,c*a2,dig);
% add=radd(add,xn,dig);
a(i)=add;
sum=radd(sum,add,dig);
y(i)=sum;
end
sumr
'      i       delta      Sum    delta(approx) Sum(approx)'
res=[[1:1:N]' ar' z' a' y']

hold off
a=plot(y,'b'); set(a,'LineWidth',2);
hold on
a=plot(z,'r'); set(a,'LineWidth',2);
a=plot(abs(z-y)./z,'g'); set(a,'LineWidth',2);
legend([ num2str(dig) ' digits'],'Exact','Error');
```

recur.m

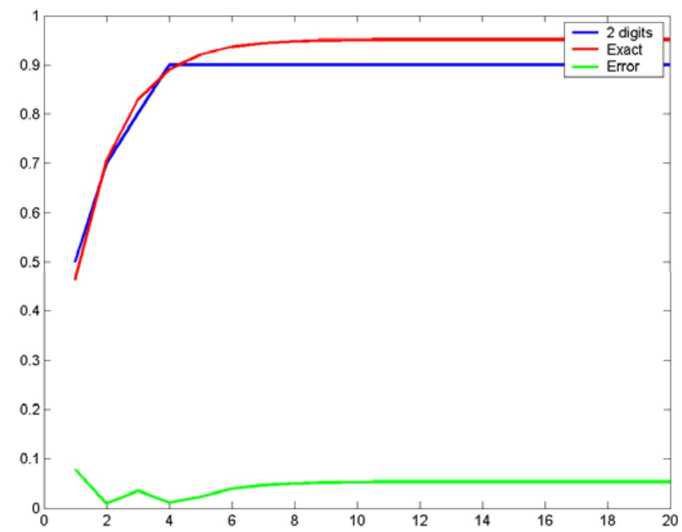
```
>> recur

b = 1; c = 1; x = 0.5;
dig=2

      i        delta       Sum     delta(approx) Sum(approx)


res =

    1.0000    0.4634    0.4634    0.5000    0.5000
    2.0000    0.2432    0.7065    0.2000    0.7000
    3.0000    0.1226    0.8291    0.1000    0.8000
    4.0000    0.0614    0.8905    0.1000    0.9000
    5.0000    0.0306    0.9212         0    0.9000
    6.0000    0.0153    0.9364         0    0.9000
    7.0000    0.0076    0.9440         0    0.9000
    8.0000    0.0037    0.9478         0    0.9000
    9.0000    0.0018    0.9496         0    0.9000
   10.0000    0.0009    0.9505         0    0.9000
   11.0000    0.0004    0.9509         0    0.9000
   12.0000    0.0002    0.9511         0    0.9000
   13.0000    0.0001    0.9512         0    0.9000
   14.0000    0.0000    0.9512         0    0.9000
   15.0000    0.0000    0.9512         0    0.9000
   16.0000   -0.0000    0.9512         0    0.9000
   17.0000   -0.0000    0.9512         0    0.9000
   18.0000   -0.0000    0.9512         0    0.9000
   19.0000   -0.0000    0.9512         0    0.9000
   20.0000   -0.0000    0.9512         0    0.9000
```



Spherical Bessel Functions

Differential Equation

$$x^2\frac{d^2y}{dx^2} + 2x\frac{dy}{dx}(x^2 - n(n+1))y = 0$$

Solutions

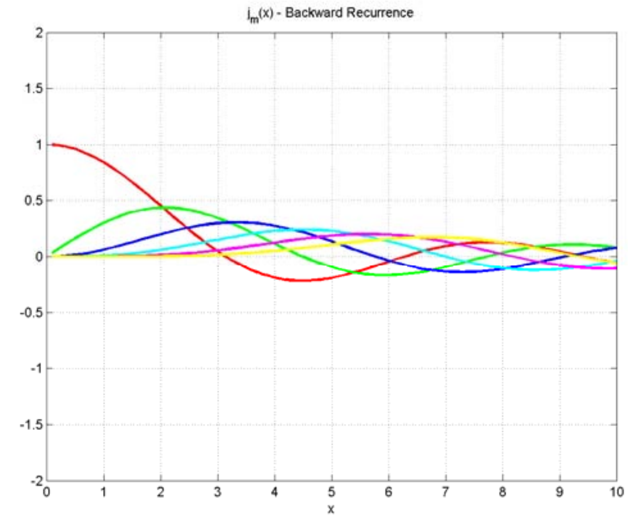
$$j_n(x) y_n(x)$$

n	$j_n(x)$	$y_n(x)$
0	$\frac{\sin x}{x}$	$-\frac{\cos x}{x}$
1	$\frac{\sin x}{x^2} - \frac{\cos x}{x}$	$-\frac{\cos x}{x^2} - \frac{\sin x}{x}$

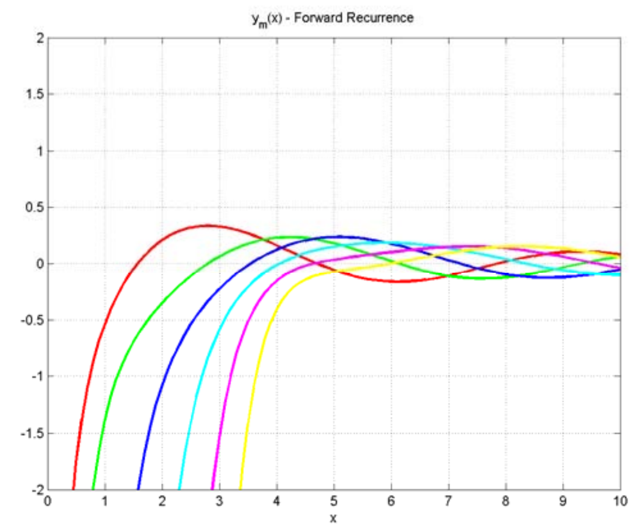
$$j_n(x) \rightarrow 0 \begin{cases} n \rightarrow \infty \\ x \rightarrow 0 \end{cases}$$

$$y_n(x) \rightarrow -\infty \begin{cases} n \rightarrow \infty \\ x \rightarrow 0 \end{cases}$$

$j_n(x)$



$y_n(x)$



Recurrence in MATLAB

Spherical Bessel Functions

sbfj_f.m

```
% Forward recurrence for spherical Bessel
% function j_n(x), n=0...N-1
function[jn]=spfj_f(x,N);
jn(1)=sin(x)/x;
jn(2)=sin(x)/x^2 -cos(x)/x;
for n=2:N
jn(n+1)=((2*n+1)/x)*jn(n) - jn(n-1);
end
```

sbfy.m

```
% Forward recurrence for spherical Bessel
% function y_n(x), n=0...N-1
clear;
x=1.0;
ynm1=-cos(x)/x;;
y(1)=ynm1
yn=-cos(x)/x^2 - sin(x)/x;
y(2)=yn
for n=2:20
ynp1=((2*(n-1)+1)/x)*yn-ynm1;
y(n+1)=ynp1;
ynm1=yn;
yn=ynp1;
end
```

sbfj.m

```
% Backward recurrence for spherical Bessel
% function j_n(x), n=0...N-1
function[j]=sbfj(x,N);
jnp1=0;
jn=1.0;
for n=N+round(x)+20:-1:N+2
jno=jn;
jn=-jnp1+ ((2*n+1)/x)*jno;
jnp1=jno;
end
for n=N+1:-1:1
jno=jn;
jn=-jnp1+ ((2*n+1)/x)*jno;
jnp1=jno;
j(n)=jn;
end
% Normalize
jr=j(1);
j0=sin(x)/x;
for n=N:-1:1
j(n)=j(n)*j0/jr;
end
```

Spherical Bessel Functions

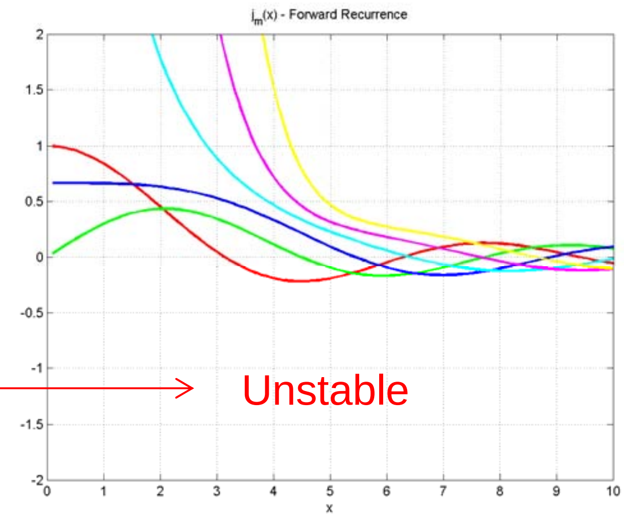
Generation by Recurrence Relations

Forward Recurrence

$$j_{n+1}(x) = \frac{2n+1}{x} j_n(x) - j_{n-1}(x)$$

Forward Recurrence

$$\frac{2n+1}{x} j_n(x) \simeq j_{n-1}(x)$$



Unstable

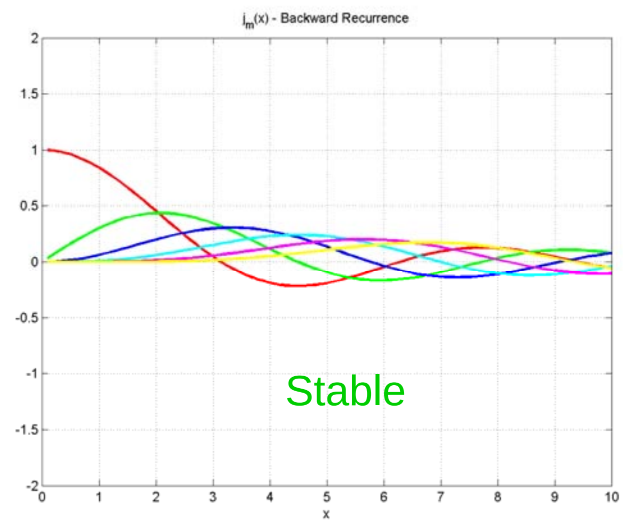
Backward Recurrence

$$j_{n-1}(x) = \frac{2n+1}{x} j_n(x) - j_{n+1}(x)$$

Miller's algorithm

$$j_N(x) = 1 \ , \ j_{N+1}(x) = 0 \ , \ j_0(x) = \frac{\sin x}{x}$$

$N \sim x+20$



Stable

Recurrence Relations

Spherical Bessel Functions

tsbfj.m

```
% test of spherical bessel function generator
x=[0.1:0.1:10];
% Backward recurrence
jf=zeros(length(x),6);
for i=1:length(x)
    j=sbfj(x(i),10);
    jf(i,:)=j(1:size(jf,2));
end
..
% Backward recurrence 3 sign digits
jf=zeros(length(x),6);
for i=1:length(x)
    j=sbfj_3(x(i),10);
    jf(i,:)=j(1:size(jf,2));
end
..
% forward recurrence
jf=zeros(length(x),6);
for i=1:length(x)
    j=sbfj_f(x(i),10);
    jf(i,:)=j(1:size(jf,2));
end
% forward recurrence 3 sign digits
jf=zeros(length(x),6);
for i=1:length(x)
    j=sbfj_f_3(x(i),10);
    jf(i,:)=j(1:size(jf,2));
end
% forward recurrence y
jf=zeros(length(x),6);
for i=1:length(x)
    j=sbfy(x(i),10);
    jf(i,:)=j(1:size(jf,2));
end
```