Introduction to Multiple-precision arithmetic on MATLAB

FUJIWARA, Hiroshi
藤原 宏志

Graduate School of Informatics, Kyoto University
京都大学 情報学研究科

This file is upated on 15th March 2016.

Floating-Point Arithmetic

Real numbers are approximated by floating-point arithmetic.

- Double Precision (MATLAB default)

$$\pi \approx (-1)^0 \times \mathbf{3.141592653589793} \times \mathbf{10}^0.$$

Exercise in MATLAB

Print $\frac{\mathbf{1}}{\mathbf{3}}$ with 30 decimal digits, and check accuracy.

```
> fprintf('%.30f %.30e\n', 1/3, 1/3);
```

Gaps between Mathematics and Computer Arithmetic.

Exercise in MATLAB

(1) Compute the sequence $\{a_n\}$ defined by

$$a_1 = 1, \quad a_2 = \frac{1}{11},$$
$$a_n = \frac{34}{11} a_{n-1} - \frac{3}{11} a_{n-2}, \quad 3 \leq n \leq 50.$$

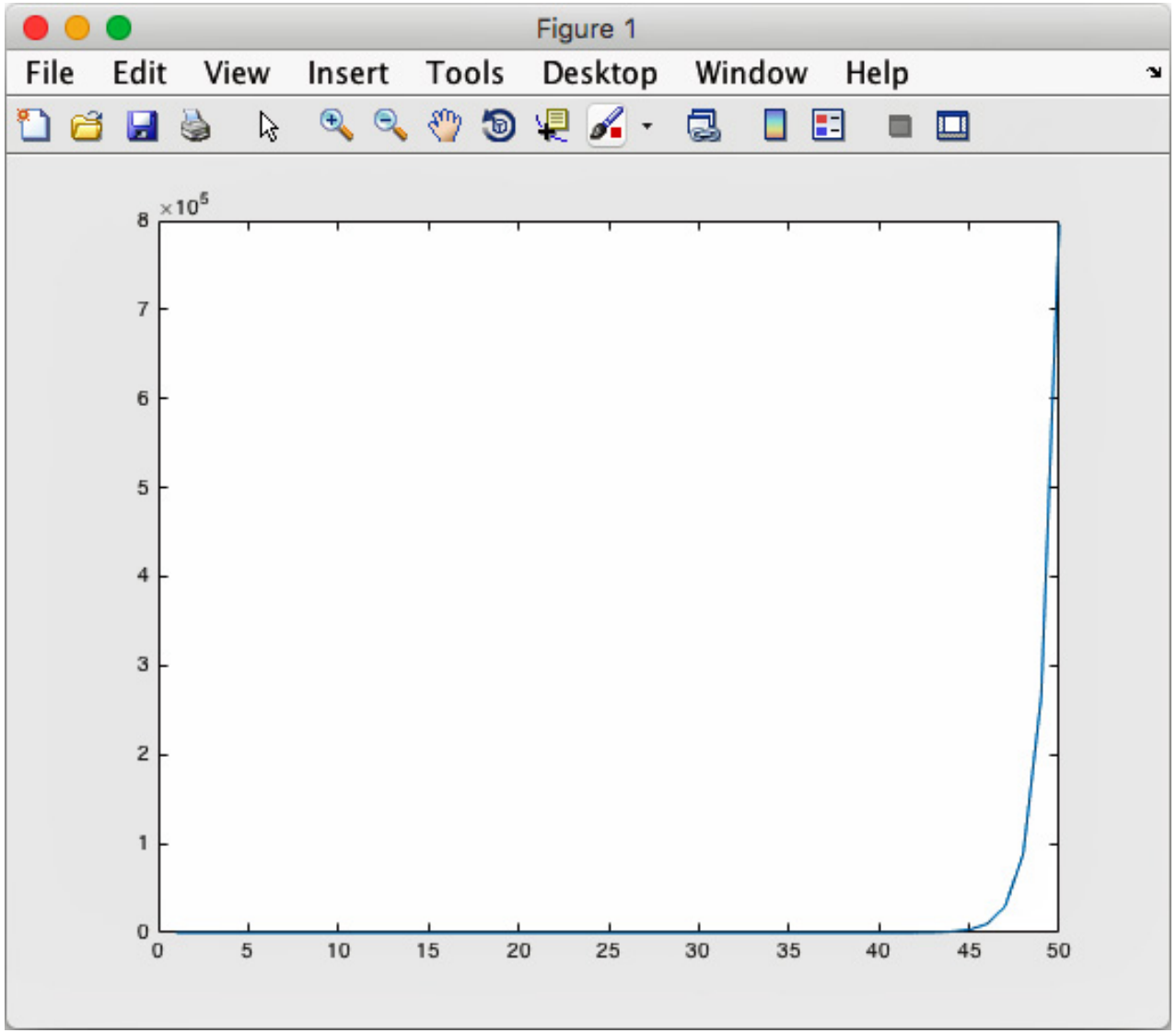
(2) Plot $\{a_n \; ; \; 1 \leq n \leq 50\}$, in semi-logarithmic graph.

Later we visit this exercise again.

Hint

```
a(1) = ...;
a(2) = ...;

for n=3:50
  a(n) = ....;
end

plot( a )
```

Answer



$a_n \to \infty$ as $n \to \infty$?

Floating-Point Arithmetic

Real numbers are approximated by floating-point arithmetic.

- Double Precision (MATLAB default)

$$\pi \approx (-1)^0 \times 3.141592653589793 \times 10^0.$$

- 100 decimal digits

$$\begin{aligned} &3.1415926535\ 8979323846\ 2643383279\ 5028841971 \\ &\quad 6939937510\ 5820974944\ 5923078164\ 0628620899 \\ &\quad 8628034825\ 3421170679 \end{aligned}$$

100 decimal digits arithmetic is not implemented in Hardwares, C, Fortran, and MATLAB.

exflib : extended precision floating-point arithmetic library

`http://www-an.acs.i.kyoto-u.ac.jp/~fujiwara/exflib`

Aimed at Scientific Computing

- Fast and large scale computation in 100–1000 decimal digits

  Arithmetic are designed and optimized in assemlby languages.
- Built-in functions

- PC : Opteron, Xeon, Core
- Solaris, Linux, MacOSX, Windows
- C++, FORTRAN90, MATLAB

Getting Started

Current version:

```
http://www-an.acs.i.kyoto-u.ac.jp/~fujiwara/exflib/
exflib-MATLAB-20160309-distribute.zip
```

(1) Unzip the archive. A new folder "`exflib-MATLAB-20160309-distribute`" is created.

(2) Move the new folder to MATLAB document root.
It is "`Documents/MATLAB`" by default.

(3) Change directory into the folder in MATLAB.

```
>> cd exflib-MATLAB-20160309-distribute
```

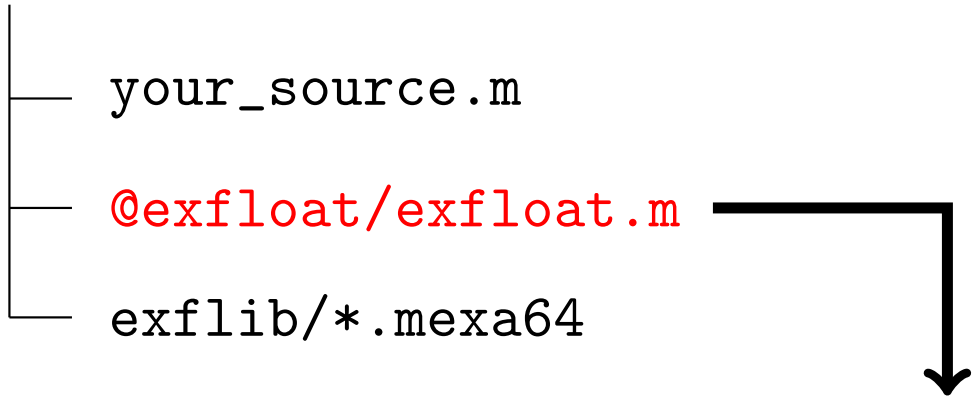
Demonstration

```
f = 1;
for n=1:100
  f = f * n;
  fprintf('fact %d = %f\n', n, f );
end
```

```
f = 1;
for n=1:100
  f = f * n;
  fprintf('fact %d = %f\n', n, f );
end
```

```
clear classes;
addpath('/Users/.../MATLAB/exflib');
f = exfloat( 1 );
for n=1:100
  f = f * n;
  fprintf('fact %d = %s\n', n, num2str(f, '%f') );
end
```

Specification of required precision




```
classdef exfloat

  properties (Constant, Access=private)

    precision10 = 1000;     % required precision
    ...
  end
  ...
end
```

§ How to Use

Substitution, Arithmetic

```
x = exfloat( 1 );              % explicit conversion
x = exfloat( i );
y = x;

x = exfloat( '0.2' );          % by String : valid
x = exfloat( '1/10' );
x = exfloat( '#PI/2+#E*2' );

x = 1;              % Invalid : INTEGER.
x = 0.1;            % Invalid : DOUBLE.
x = '#PI/2';        % Invalid : STRING.
```

```
y = exp( x );    % y is exfloat type
z = x + y;
w = z * 2;
```

Array (Matrix)

```
for i=1:N
  for j=1:N
    A(i,j) = exfloat( 1 ) / (i+j-1);
  end
  B(i) = exfloat( 1 )
end

x = A \ B';     % solve Ax = B
```

Output

num2str( ) : Converting exfloat to string

```
x                                       % without semi-colon
num2str(x)                              % without semi-colon

disp( num2str(x) );
fprintf('x=%s\n', num2str(x) );

% specify output format and precision
disp( num2str(x, '%.20f') );    % fixed-point
disp( num2str(x, '%.30e') );    % exponential

fprintf('x=%s\n', num2str(x, '%.40f');
```

Output — Array

`num2str( )` is available for an array

```
% 1-D array
a(1:N) = exfloat;

% all entries are printed in the specified format
disp( num2str(a, '%.20f') );

% partially
disp( num2str(a(2:N-1), '%.20f') );

fprintf('a=%s\n', num2str(a,  '%e') ); % row-vector
fprintf('a=%s\n', num2str(a', '%e') ); % column-vector
```

exflib on MATLAB — Comparison

```
if ( x == y )
  ...

if ( x ~= y )
  ...

if ( x < y )
  ...

% valid, comparison with integer or double types
if ( x <= 0.1 )
  ...
```

Exercise (revisit)

Using $\mathbf{100}$ decimal digits.

(1) Compute $\{a_n\}$ defined by

$$a_1 = 1, \quad a_2 = \frac{1}{11},$$
$$a_n = \frac{34}{11} a_{n-1} - \frac{3}{11} a_{n-2}, \quad 3 \leq n \leq 50.$$

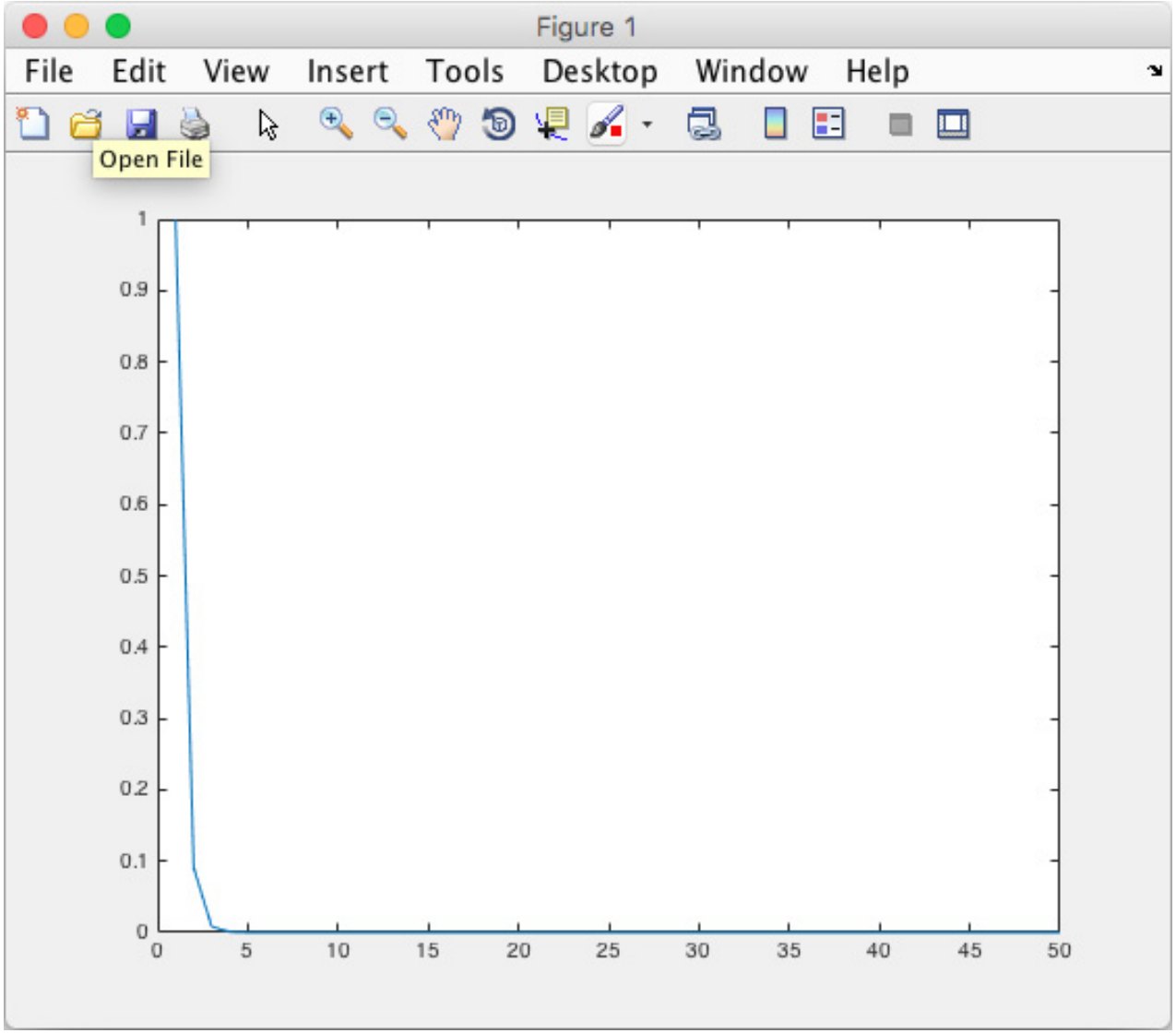
(2) Plot $\{a_n \; ; \; 1 \leq n \leq 50\}$, in semi-logarithmic graph.

(3) Compare double precision results with multiple-precision ones.

Answer

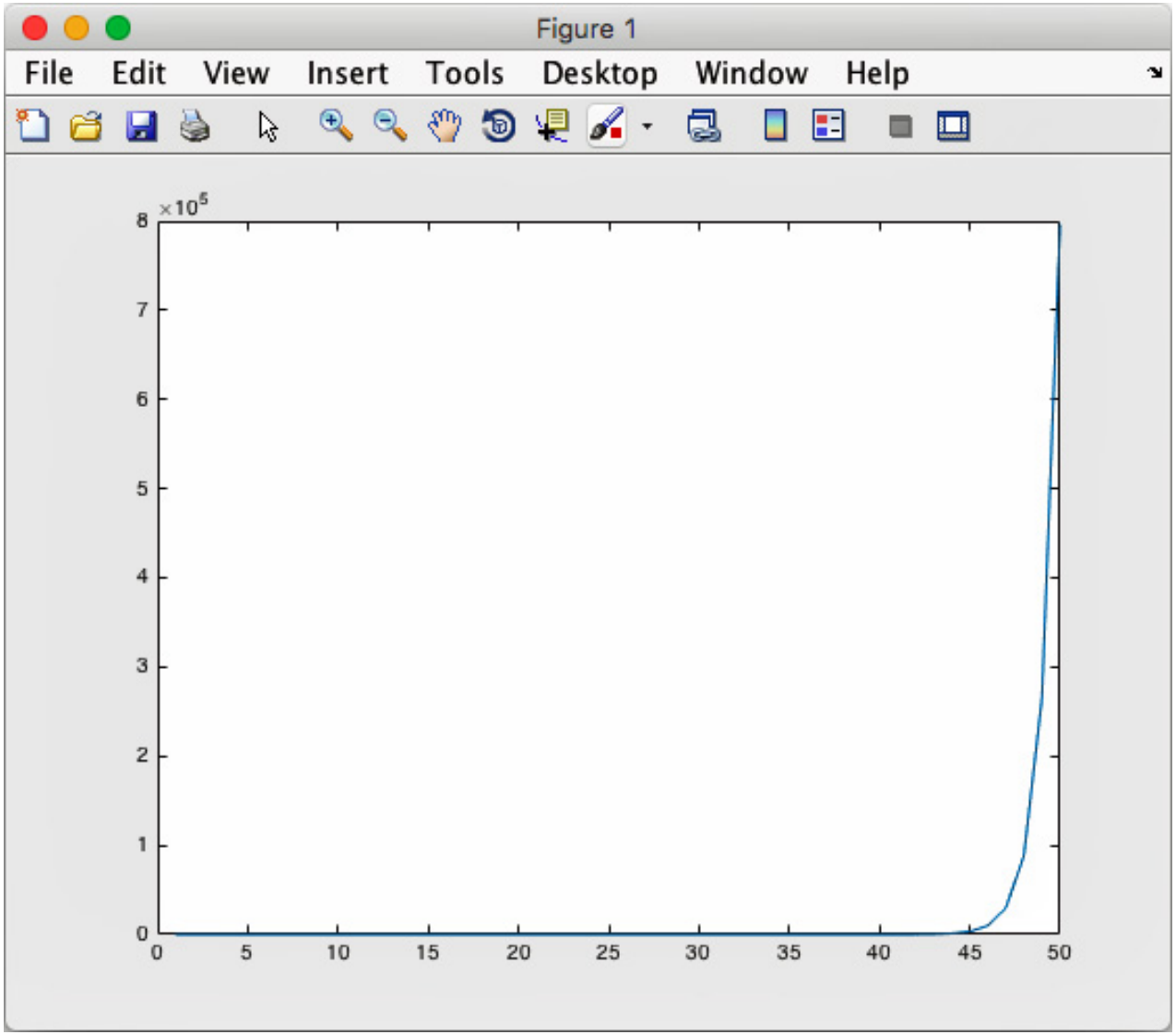
```
clear classes;

addpath('/Users/.../MATLAB/exflib');

a(1) = exfloat( 1 );
a(2) = exfloat( '1/11' );

for n=3:50
  a(n) = exfloat('34/11') * a(n-1)
           + exfloat('3/11') * a(n-2);
end

plot( a )
```

Answer (100 decimal digits)



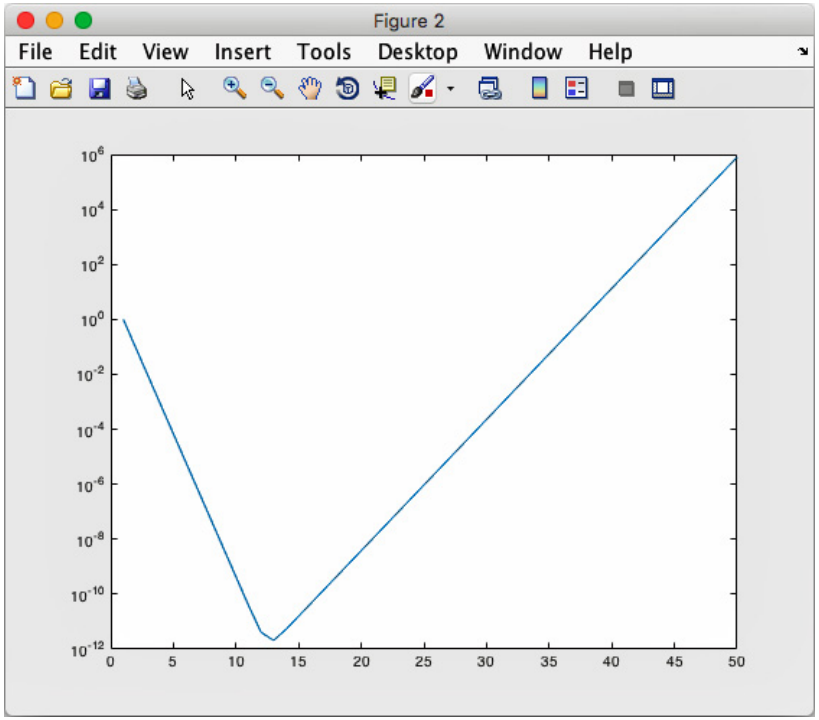
$a_n \to 0$ as $n \to \infty$?

Answer (double precision)

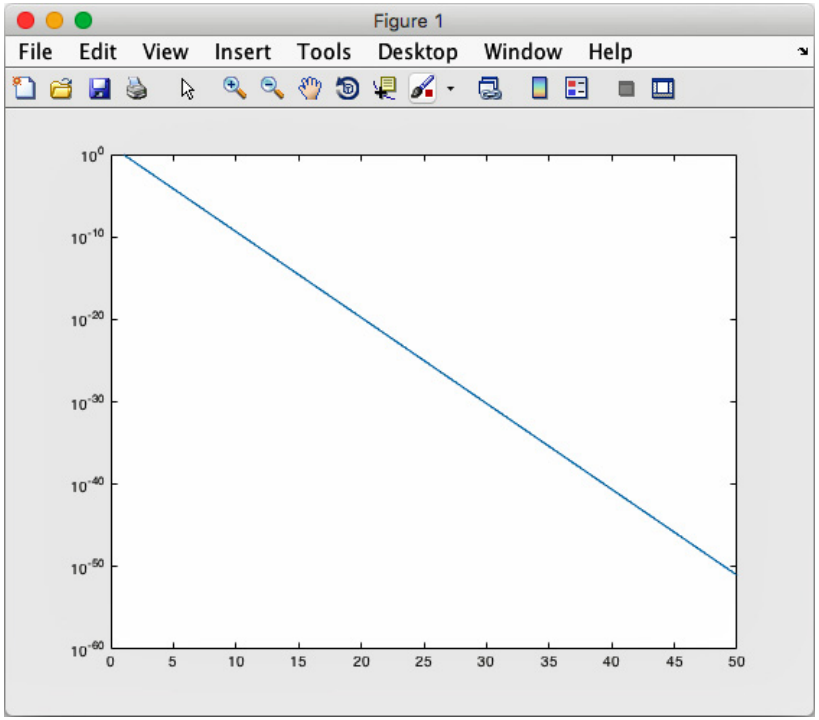


What happens in double precision arithmetic?

Answer (semi-logarithmic plot)



Double Precision



100 decimal digits

```
figure
semilogy( a )
```

Implemented methods

Arithmetic

- `+` (`plus`), `-` (`minus`), `*` (`mtimes`), `/` (`mrdivide`)

Array, matrix operations

- `\` (`mldivide`), `'` (`transpose`), `sum`, `dot`, `prod`

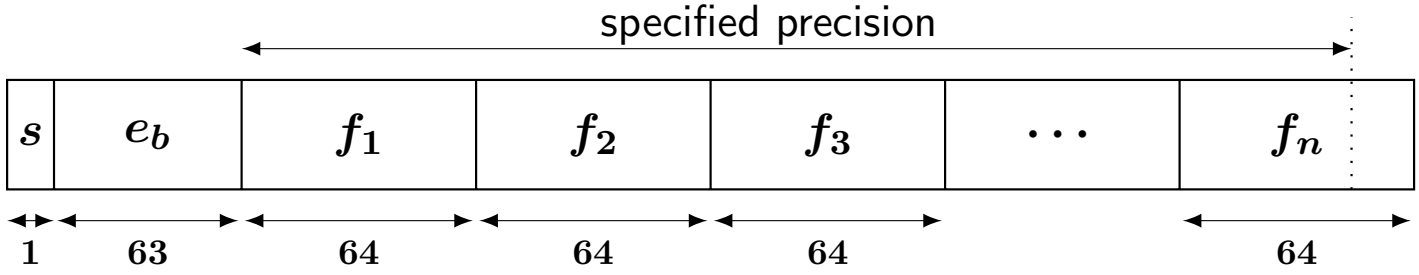
Built-in functions

- `abs`, `sqrt`
- `sin`, `cos`, `tan`, `asin`, `acos`, `atan`, `atan2`, `sinh`, `cosh`, `tanh`
- `exp`, `.^` (`power`), `log`, `log10`

Comparisons, entry-wise operations (`.*` (`times`), `.\` (`ldivide`), `./` (`rdivide`)) are also implemented.

exflib on MATLAB — Multiple-Precision Number Sizes

```
% specified precision (decimal digits)
exfloat.get_req_precision()

% internal precision (decimal digits) ≈ 19.26n
exfloat.get_precision()

% memory size (bytes) = 8(n + 1)
exfloat.get_exfloat_byte()

% size of fraction parts = n
exfloat.get_exfloat_precision64()
```



64 bits $\approx$ **19.26** digits.

§ Technical Issues

Output — as double precision type

`double( )` : Converting `exfloat` to a double precision

- fast
- graphics

```
x = exfloat( '#PI' );
double ( x )

t(i) = double( ... );          % t(1) ... t(N)
u(i) = exfloat( ... );         % u(1) ... u(N)
plot( t, double ( u ) );       % array conversion
```

Computational speed : solving linear equation

```
tic;  x = A \ b; toc;
```

unit : sec.

digits	size	MATLAB VPA	MATLAB proposed	MATLAB ratio	C++ exflib
100	**100**	1.54	0.520	3.0	0.00960
	200	10.0	2.19	4.6	0.0903
	400	78.3	10.2	7.7	2.97
	800	636	51.0	13	23.8
	1000	1220	89.9	14	40.2
500	**100**	2.04	0.755	2.7	0.0257
	200	13.4	4.16	3.2	1.97
	400	104	25.9	4.0	15.5
	800	787	179	4.4	124
	1000	1510	337	4.5	284

MATLAB R2015a (Version 8.5) / Linux on Xeon E5-2670 (2.6GHz)

▸ On-Going Work

Overhead of MEX

- `your_source.m`
- `@exfloat/` MP class and methods definition
- `exflib/` Interface to exflib by MEX (MATLAB Executable)
  (`*.mexa64` for Linux, `*.mexmaci64` for MacOSX)

'`x + y`' in user programs

↓

`@exfloat/plus.m`

↓

`exflib/exflib_float_add.mexa64` (MEX)

↓

'add' (assemble code) in exflib

Overhead of MEX

Example: inner product

```
% 2N times MEX calling
for i=1:N
  s = s + a(i) * b(i);
end
```

```
% Only one time MEX calling
s = dot( a, b );
```

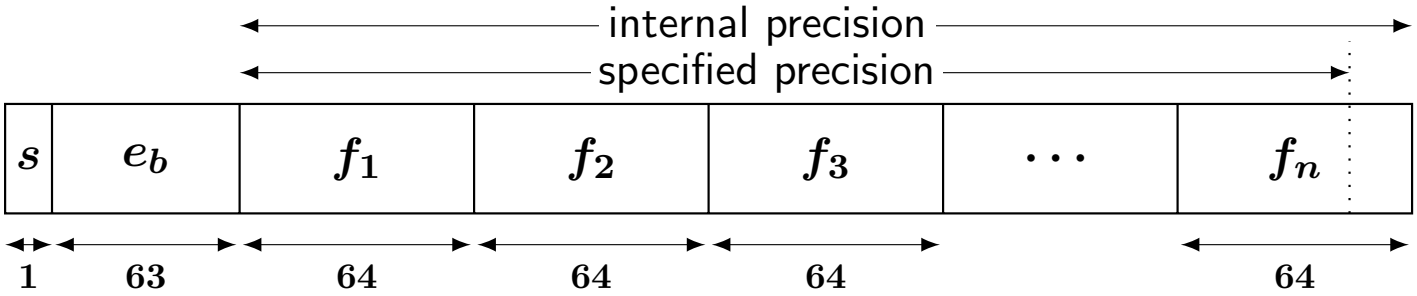
100 decimal digits, unit : sec.

N	entry-wise	dot(a,b)	ratio	sum(a.*b)	a*b'
10	0.0241	0.00576	4	0.0174	0.00702
100	0.107	0.00934	11	0.0389	0.0107
1000	1.04	0.0853	12	0.377	0.102
10000	10.4	0.856	12	3.79	1.00

MATLAB R2015a (Version 8.5) / Linux on Xeon E5-2670 (2.6GHz)

Data Structure of Exfloat Number

`uint64(1:1+n)` : unsigned 64-bit integer array



$$(-1)^s \times 2^e \times \left(1 + \sum_{k=1}^{n} \frac{f_k}{2^{64k}}\right)$$

64 bits $\approx$ **19.26** digits.

specified digits	64-bit array length n			internal digits
100	**5.17**	$\rightarrow$	**6**	**115.90**
200	**10.36**	$\rightarrow$	**11**	**212.23**
500	**25.93**	$\rightarrow$	**26**	**501.21**
1000	**51.89**	$\rightarrow$	**52**	**1002.13**

Example — Variable Precision Arithmetic (VPA)

```
clear classes;
digits(1000);

n = 50;
mat = vpa( hilb(n) );
rhs = vpa( ones(size,1) );

sol = vpa( mat \ rhs )
```