APA - An Arbitrary Precision Arithmetic toolbox

for Octave and Matlab

オールフス　カイトーベン
OHLHUS, Kai Torben

Graduate School of Science
Tokyo Woman's Christian University

"Development of Verified Numerical Computations for Mathematical Modeling"
JST/CREST Result Briefing http://verifiedby.me/nvr2021/
(online)

November 27, 2021

GNU Multiple Precision (MP) Libraries

- **GMP - GNU MP Arithmetic Library**[1] (C/C++, version 6.2.1; since 1991)
  - supports: signed integers, rational numbers, and floating-point numbers
  - arithmetic and logic functions for supported data types
  - focus: **speed, cryptography**
- **MPFR - GNU MP Floating-Point Reliable Library**[2] (C, version 4.1.0; since 2000)
  - supports: floating-point numbers (arithmetic and logic functions)
  - focus: **correct rounding**, precise semantics (IEEE 754)
- **MPC - GNU MP Complex Library**[3] (C, version 1.2.1; since 2003)
  - addition: **complex** floating-point numbers

- Many interfaces (C++, Python, ...) are available.

[1] https://gmplib.org/
[2] https://www.mpfr.org/
[3] http://www.multiprecision.org/

Existing Octave/Matlab toolboxes

- **Matlab VPA (Variable-Precision Arithmetic, Symbolic toolbox)**[4]
  - Slow for dimensions > 200.
- **Advanpix**[5] (version 4.8.5.14569; since 2010) Very fast and complete!
  - closed source (P-code), proprietary license
  - Matlab only
- **GEM - Gmp Eigen Matrix Library**[6] (version 2.0.2; since 2016)
  - free, open source (MPL-2.0, others)
  - Matlab and Octave
- **mptoolbox**[7] (version 1.5; since 2004)
  - free, open source (BSD-3-Clause)
  - Matlab and "Octave" (needs some adaptions)

---

[4] https://www.mathworks.com/help/symbolic/vpa.html

[5] https://www.advanpix.com/

[6] https://gem-library.github.io/

[7] https://sourceforge.net/projects/mptoolbox/ and https://www.mathworks.com/matlabcentral/fileexchange/6446-multiple-precision-toolbox-for-matlab

What makes APA different?

– Work with C Code[8] interactively in Octave/Matlab.

```c
#include <stdio.h>

#include <gmp.h>
#include <mpfr.h>

int main (void)
{
  unsigned int i;
  mpfr_t s, t, u;

  mpfr_init2 (t, 200);
  mpfr_set_d (t, 1.0, MPFR_RNDD);
  mpfr_init2 (s, 200);
  mpfr_set_d (s, 1.0, MPFR_RNDD);
  mpfr_init2 (u, 200);
  for (i = 1; i <= 100; i++)
    {
      mpfr_mul_ui (t, t, i, MPFR_RNDU);
      mpfr_set_d (u, 1.0, MPFR_RNDD);
      mpfr_div (u, u, t, MPFR_RNDD);
      mpfr_add (s, s, u, MPFR_RNDD);
    }
  printf ("Sum is ");
  mpfr_out_str (stdout, 10, 0, s, MPFR_RNDD);
  putchar ('\n');
  mpfr_clear (s);
  mpfr_clear (t);
  mpfr_clear (u);
  mpfr_free_cache ();
  return 0;
}
```

```matlab
t = mpfr_t (1.0, 200, MPFR_RNDD);
s = mpfr_t (1.0, 200, MPFR_RNDD);
u = mpfr_t (nan, 200);

for i = 1:100
  mpfr_mul_ui (t, t, i, MPFR_RNDU);
  mpfr_set_d (u, 1.0, MPFR_RNDD);
  mpfr_div (u, u, t, MPFR_RNDD);
  mpfr_add (s, s, u, MPFR_RNDD);
end

fprintf ('Sum is ');
disp (s, MPFR_RNDD);
```

APA **Low-Level** interface:

- Easy testing and debugging.
- Less interpreter overhead.

```matlab
setround = @(rnd) ...
mpfr_set_default_rounding_mode (rnd);
mpfr_set_default_prec (200);

setround (MPFR_RNDD)
t = mpfr_t (1.0);
s = mpfr_t (1.0);

for i = 1:100
  setround (MPFR_RNDU);
  t = t * i;
  setround (MPFR_RNDD);
  s = s + (1 / t);
end

fprintf ('Sum is ');
disp (s);
```

APA **High-Level** interface:

- Mathematical language.
- More interpreter overhead.

[8] Example from https://www.mpfr.org/sample.html (2021-11-19).

What makes APA different?

– Work with C Code interactively in Octave/Matlab.

mpfr_add.m*

```
function ret = mpfr_add    (       rop ,     op1 ,      op2 ,     rnd)
  ret = mex_apa_interface (1031 , rop.idx , op1.idx , op2.idx , rnd);
end
```

mex_apa_interface.mex*

mpfr_data = | ... | **mpfr_t** | **mpfr_t** | **mpfr_t** | **mpfr_t** | ... |

```
void mexFunction ( /* ... */ ) {
  switch (cmd_code) {
    case 1031:
     #pragma omp parallel for
     for (size_t i = 0; i < length (&rop); i++)
       ret[i] = mpfr_add (rop + i, op1 + i, op2 + i, rnd);
  }
}
```

*Code simplified for presentation.

Low-level **function calls** are fast:

- Almost no m-code (less interpreter overhead).
- Vectorized: no loops in m-code necessary.
- Almost no data transfer (two array indices for each matrix of **any size**).
- OpenMP **parallelization** where possible.
- Access to MPFR return value.

What makes APA different?

Fast and easy installation with 5 Octave/Matlab commands.

```
urlwrite ('https://github.com/gnu-octave/apa/releases/download/v1.0.0/apa-1.0.0.zip', 'apa-1.0.0.zip');
unzip ('apa-1.0.0.zip');
cd (fullfile ('apa-1.0.0', 'inst'))
install_apa
test_apa
```

- Internally used GMP and MPFR libraries provided as pre-compiled static libraries.
- Tested and works with Octave and Matlab on **MS Windows, macOS, and Linux**.

APA - Familiar syntax/semantic to other Octave/Matlab toolboxes

MPFR Matrix datatype and operations

```
>> A = mpfr_t (eye (3)) + 2
A =
    3    2    2
    2    3    2
    2    2    3
```

Output customization

```
>> apa ('format.base', 16)
>> a = 1 - mpfr_t (eps)
a = 0.fffffffffffff

>> apa ('format.base', 2)
>> apa ('format.fmt', 'scientific')
>> e = mpfr_t (eps)
e = 1 * 2^(-52)
```

Precision as binary digits (MPFR compatiblity)

```
>> prec10 = 50;                        % decimal digits
>> prec2  = ceil(prec10*log2(10));     % binary  digits
>> PI = mpfr_t ('pi', prec2)
PI =
   3.1415 92653 58979 32384 62643 38327 95028 84197 16939 93751 01
```

- Estimated necessary binary precision slightly larger, 50 decimal digits guaranteed correct.

What else makes APA different?

– Analysis of accuracy loss in Algorithms[9].

```
>> A = mpfr_t ([2 1 1; 1 2 1; 1 1 2])
A =

    2    1    1
    1    2    1
    1    1    2

>> [L,U,~,ret] = lu (A)

L =

      1                      0    0
    0.5                      1    0
    0.5    0.3333333333333331    1

U =

    2      1                     1
    0    1.5                   0.5
    0      0    1.333333333333333

ret =

    0    0    0
    0    0    0
    0   -1    1
```

Most MPFR functions provide a ternary return value:

- 0: returned value is exact.
- ± 1: returned value is greater/lower than the exact result.

- Many MPFR-based libraries ignore this value.
- Possibility to detect loss of accuracy in algorithms (faster than interval arithmetic).
- Support design decisions about least necessary precision for given input.

[9] Experimental feature and not contained in APA 1.0.0.

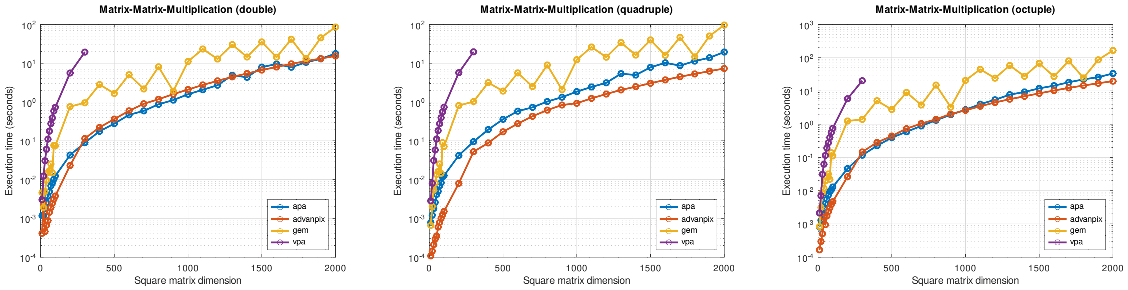
Benchmark (1/4)

- Test system:
  - Intel(R) Xeon(R) Platinum 8276 CPU @ 2.20GHz, 28 cores (56 threads), 1 TB RAM
  - Matlab R2021a (Update 5) on CentOS 7
  - APA (1.0.0), Advanpix (4.8.5.14569), GEM (2.0.2), VPA (R2021a)
- Tests with A = rand(N,N) and b = A ∗ ones(N,1) mean value of 10 repetitions:
  - Matrix-Matrix-Multiplication A ∗ A
  - LU-factorization [L,U] = lu (A)
  - Solve $Ax = b$ x = A \ b
  - Precisions [1, Chapter 3.1]:

Name	Binary	Decimal[10]
"double"	53	15
"quadruple"	113	34
"octuple"	237	71

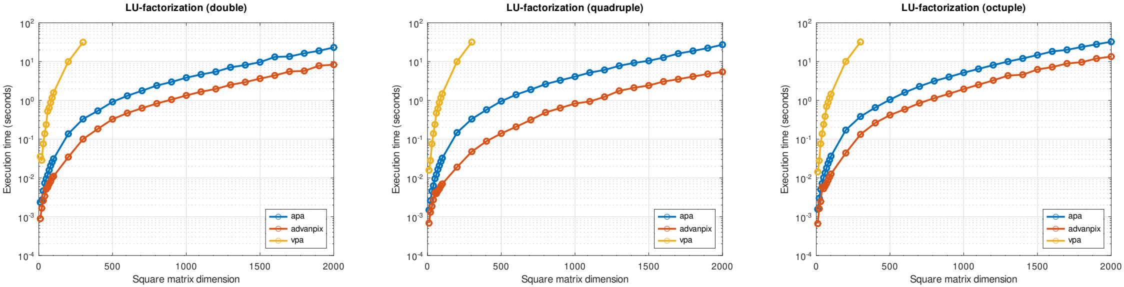
[10] decimal precision := $\lfloor$binary precision $\times \log_{10}(2)\rfloor$, see "bits2digits" in http://www.holoborodko.com/pavel/mpfr/.

Benchmark (2/4) Matrix-Matrix-Multiplication



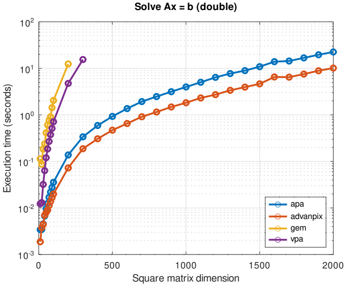
- VPA is slow for dimensions > 300.
- APA is comparable to Advanpix in double precision, partially by factor $\backsim 2$ slower in higher precisions, but **slightly faster than Advanpix** in dimensions $500 - 900$.
- GEM shows a zig-zagging performance with a factor $\backsim 4 - 5$ slower than APA.
- APA multiplies two matrices of dimension $N = 2000$ in 34 seconds in octuple precision.

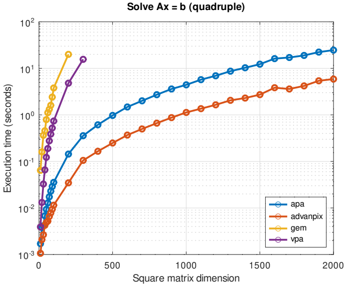
Benchmark (3/4) LU-factorization

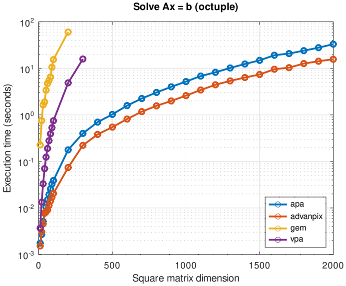


- VPA is slow for dimensions > 300.
- APA is by factor $\backsim 3$ (quadruple $\backsim 4$) slower than Advanpix.
- APA factorizes a matrix of dimension $N = 2000$ in 27 seconds in octuple precision.

Benchmark (4/4) Solve $Ax = b$







- GEM and VPA are slow for dimensions > 200.
- APA is by factor $\backsim 2$ (quadruple $\backsim 4$) slower than Advanpix.
- APA solves a system of dimension $N = 2000$ in 33 seconds in octuple precision.

Summary and outlook

APA is an arbitrary precision arithmetic toolbox for Octave/Matlab:

- **Free, extensible, and fast** compared to other toolboxes.
- **Very easy installation** from inside Octave/Matlab command window.
- Microsoft Windows, macOS, and Linux support.

What comes next?

- Extend implementation:
  - Cholesky factorization, SVD-, Eigenvalue-decomposition
  - MPC interface (complex data)
  - Sparse and N-dimensional data type and algorithms
  - Research and test published MPFR-algorithms (dot-product [2], matrix multiplication [3], ...)
  - Usage of other MPFR-based libraries (MPLAPACK [4], ...)?
- Use MPFR ternary return value to detect accuracy loss in algorithms.

Thank you for your attention!

Questions?

Slides and sources available at: https://github.com/siko1056/slides_nvr2021

APA-Toolbox: https://github.com/gnu-octave/pkg-apa

References I

[1] Jean-Michel Muller et al. *Handbook of Floating-Point Arithmetic*. Springer, 2018. DOI: 10.1007/978-3-319-76526-6.

[2] Konstantin Isupov, Vladimir Knyazkov, and Alexander Kuvaev. "Design and Implementation of Multiple-Precision BLAS Level 1 Functions for Graphics Processing Units". In: *Journal of Parallel and Distributed Computing* 140 (2020), pp. 25–36. ISSN: 0743-7315. DOI: 10.1016/j.jpdc.2020.02.006.

[3] Tomonori Kouya. "Performance Evaluation of Multiple Precision Matrix Multiplications Using Parallelized Strassen and Winograd Algorithms". In: *JSIAM Letters* 8 (2016), pp. 21–24. DOI: 10.14495/jsiaml.8.21.

[4] Maho Nakata. "MPLAPACK Version 1.0.0 User Manual". In: *arXiv:2109.13406 [cs]* (2021). arXiv: 2109.13406 [cs].