Multiprecision – solving and causing problems

Danielle Amethyst Brake

28 March, 2019

CUNY
Numeric-symbolic computation seminar
Outline

We need multiprecision

Numbers in a computer
  Words
  Types
  Resources
  Concerns

Numerical nonlinear algebra uses multiprecision
  Plain old straight-line tracking
  Endgames
  Certification

Concerns
  Tracking
  Convention
Crypto needs multiprecision

Combinatorial numbers easily run over the bounds of hardware types. Consider factorial...

- 13! is enough to overflow an uint32_t, as does $3^{21}$
- 21! overflows an uint64_t
- 19! causes a double to start rounding

64 bits is clearly not enough.

So what are we to do about needing very long numbers for encryption?
Linear Algebra needs multiprecision

- When solving linear system, we may lose up to $\log_{10} \text{cond}(M)$ digits of accuracy

Take a look at the Hilbert matrix of size $n$

$$H_{i,j} = \frac{1}{i + j - 1}$$

For example,

$$\text{hilb}(3) = \begin{bmatrix} 1 & \frac{1}{2} & \frac{1}{3} \\ \frac{1}{2} & \frac{1}{3} & \frac{1}{4} \\ \frac{1}{3} & \frac{1}{4} & \frac{1}{5} \end{bmatrix}$$
by $N = 11$ or so, $H$ is “unusable”
- condition number $\approx 5.22e + 14$
Magic square

Magic square – each row and column sums to the same number

\[
\text{magic}(4) = \begin{bmatrix}
16 & 2 & 3 & 13 \\
5 & 11 & 10 & 8 \\
9 & 7 & 6 & 12 \\
4 & 14 & 15 & 1
\end{bmatrix}
\]
How are we to tell whether it’s singular or ill-conditioned or what?
Compiling needs multiprecision

Have you ever needed compile GCC from source?
🤔 Did you notice that it needs GMP?
Why?

- Constant folding and compile time arithmetic – Fortran esp.
- Cross-compiling – from 64 to 32 bits, etc
- Polytope model – for loop unrolling
Science needs multiprecision

Some recent papers mentioning adaptive precision:

- “Finding Intersections of Algebraic Curves in a Convex Region using Encasement”
- “The Performance of an adaptive aggregate smoothing training algorithm for max-min fuzzy neural networks”
- “Adaptive precision in block-Jacobi preconditioning for iterative sparse linear system solvers”
- “Adaptive Precision Cellular Nonlinear Network”

Danielle sez: I like sentence case for titles. How Do I Know What’s A Proper Noun In Title Case?
Outline

We need multiprecision

Numbers in a computer
  Words
  Types
  Resources
  Concerns

Numerical nonlinear algebra uses multiprecision
  Plain old straight-line tracking
  Endgames
  Certification

Concerns
  Tracking
  Convention
Rounding incurs error

Definition

Rounding error: the distance from an approximation to the true result, due to rounding

We incur rounding error all the time in floating point arithmetic!

- Before 1985 and IEEE 754, rounding was all over the place, with no consistent behaviour specified.
- ∃ Lots of rounding modes, aimed at providing error control and consistency.
Cancellation incurs error

**Definition**

Cancellation error: when an operation makes the relative error increase faster than the absolute error

We incur cancellation error all the time in floating point arithmetic!

- The quadratic formula has catastrophic loss of significance when the constant term is small
- Subtracting two nearby numbers results in cancellation
Hardware numeric types

- **int**
  - Often 32 bits (int32_t)
  - Ranges up to perhaps 2 billion (±?)

- **float**
  - Often 32 bits
  - 24 bits of precision, other in exponent etc

- **double**
  - Often 64 bits
  - 53 bits of precision

- **higher – double double, quadruple**
  - 106-113 bits
  - Variously supported
Software and non-built-in types

- **Bigint**
  - arbitrary length integers

- **Bigrat**
  - About the only solution for rational numbers

---

- **fixed precision bigfloats**
  - 128, 256, 512 bits are common
  - Complex number support is ?

- **variable precision bigfloats**
  - Complex number support is ?

Offered by various libraries

do not use std::complex<T> with any higher precision types!!!
Resources – multiprecision

- “What Every Computer Scientist Should Know About Floating-Point Arithmetic” – David Goldberg, 1991
C & C++ Libraries

- Foundation - gmp
  "Arithmetic without limitations"

- Extended - mpfr
  "C library for multiple-precision floating-point computations with correct rounding"

- C - mpc
  "C library for the arithmetic of complex numbers with arbitrarily high precision and correct rounding of the result"

- C++ - mp_real
  Header-only, no updates since 2015 😞, 🔴

- C++ - Boost.Multiprecision
  Header-only, actively maintained 😊, expression templates, ✔️
Why not just use C?

Observe this basic Bertini code:

```c
comp_mp unit_width; init_mp2(unit_width, 1024);
set_one_mp(unit_width);
comp_mp num_intervals; init_mp2(num_intervals, 1024);
set_zero_mp(num_intervals);

mpf_set_d(num_intervals->r, target_num_samples - 1);
div_mp(unit_width, unit_width, num_intervals);

clear_mp(interval_width);
clear_mp(num_intervals);
```
Same code in C++

and now the same code written in Bertini2:

```cpp
mpc unit_width = mpc(1)/num_intervals;
```

Yes, there’s a cost 🧑‍🔧. ⇒ Temporaries are easy to generate and hard to control.

- Solution: use expression templates, a technique for eliminating temporaries

📖 Discovering Modern C++: An Intensive Course for Scientists, Engineers, and Programmers, Gottschling 2015
Ambient type choice

What should be the choice for ambient numeric type?

- Should everything be stored at the precision used to compute it?
- Should everything be at precision just sufficient for the desired level of accuracy?
- Should we use doubles for low-precision numbers and multiples for high, regardless of accuracy?

Conclusion: you need to keep metadata
Concerns – Type switching, part 1

How to deal with when to switch numeric types in an algorithm is not trivial.

- Policy – do you use a state machine? something ad hoc? How to write prescriptions for how to transition?
- Storage – need several numeric types, some of which are (probably) heap allocated. How to minimize the number of numbers you need, and allocations?
Concerns – Type switching, part 2

How to deal with when to switch numeric types in an algorithm is not trivial.

- Policy – when increasing, should state be sharpened?
- Implementation – how to represent the state of precision compactly?
Concerns – Parallelization

- CPU parallelization – ok, whatever, every type is good (thread safety concerns?)
- GPU parallelization – multiprecision not implemented, but some extended precision types are.

“Accelerating Polynomial Homotopy Continuation on a Graphics Processing Unit with Double Double and Quad Double Arithmetic” – Verschelde, Xiangcheng 2015
Outline

We need multiprecision

Numbers in a computer
  Words
  Types
  Resources
  Concerns

Numerical nonlinear algebra uses multiprecision
  Plain old straight-line tracking
  Endgames
  Certification

Concerns
  Tracking
  Convention
Solve the Davidenko Diffeq

From the homotopy’s equation $H(z, t) = 0$ we immediately get a diffeq to solve:

$$\dot{H} = H_{z_i} \dot{z}_i = 0$$

and from there, a concrete way to advance through time:

$$\dot{z} = -JH^{-1} \dot{H}$$

Easy, just invert JH, and track! Right!?!?! 😊
Homotopy continuation

\[ f(z) \quad \text{and} \quad g(z) \]

\[ H(z(t), t) = 0 \]
\[ H = (1 - t)f(z) + tg(z) \]

\( t = 0 \) \quad endgame boundary

\( t = 1 \) \quad smooth start points

© Danielle Amethyst Brake
A single path – higher precision needed

 риск — Condition number of $\approx 10^{12}$. Uh-oh! Double is no good!

From AMP2 – “Stepsize control for path tracking” [BHSW 2009]
Fundamental pattern

After several successful steps, take long steps

Predict
Correct
Predict
Predict
Correct
Correct
Large residual

==> reduce step size

Predict
Correct
Linear algebra "fails"

==> increase precision

After several successful steps, take long steps

$\Delta t_{i+1}$  $\Delta t_i$  $\Delta t_{i-1}$
As a flowchart
focus on the yellow boxes
How you know you need more precision

First, know some things about your system:

- Bound the coefficients – $c$
- Bound the degrees – $d$
- Bound the growth in error during linear solves – $\epsilon$.
  $\Rightarrow$ Precision dependent!!!!
- Bound the error in function evaluation – $\psi$.
  $\Rightarrow$ A good estimate is $d \cdot c$
- Bound the error in jacobian evaluation – $\phi$.
  $\Rightarrow$ A good estimate is $d(d - 1)c$

Also, make some choices:

- $s_1$, Safety digits during A,B – higher is more cautious
- $s_2$, Safety digits during C
- $M$, Maximum allowable precision
criterion A

Theorem (BHSW09)

If
\[
\text{current digits} < s_1 + \log_{10}(\|J^{-1}\|\epsilon(\|J\| + \phi))
\]

then you need more precision or to shrink stepsize

used in:

- Predicting – Explicit ODE method
- Newton’s method
criterion B

Theorem (BHSW09)

\[ \text{if} \]
\[ \text{current digits} < s_1 + D + ( - \log_{10}(\text{tol}) + \log_{10}(||z_{\text{newtres}}||))/\eta \]

then you need more precision or to shrink stepsize

used in:

- Newton’s method, correct loop

\[ D = \log_{10}(||J^{-1}||((2 + \epsilon)||J|| + \epsilon\phi) + 1) \]

- \( \eta \) is the number of allowable Newton steps left
criterion C

Theorem (BHSW09)

if

\[
\text{current digits} < s_2 - \log_{10}(\text{tol}) + \log_{10}(\|J^{-1}\|\psi + \|z\|)
\]

then you need more precision or to shrink stepsize

used in:

- Predicting – Explicit ODE method
- Correcting – Newton’s method
AMP tracking resources

 대하여 AMP1 – “Adaptive multiprecision path tracking”, [BHSW 2008]
   ▶ good foundation, poor for implementation

 대하여 AMP2 – “Stepsize control for path tracking”, [BHSW 2009]
   ▶ ok for implementation

   ▶ ok for implementation
Tracking toward a singularity

the points are regularly spaced in the projection. terrible.
adding more points doesn’t really help; need more at the ends.
re-scaling projection spacing by the cycle number
Endgames

Definition
An endgame is a path-tracking tool used to compute singular roots of a homotopy.
Cauchy – procedure

- Fact: the path is a Puiseaux series in $t$.
- Method: Extrapolate to $t = t_0$ via a Cauchy integral (just compute the mean!)
- Tool: Walk circles around $t_0$ to gather numerical data to do Trapezoid Rule quadrature.

$$f(s) = \frac{1}{2\pi i} \oint \frac{f(s)}{s - t_0} \, dz$$

1. Guess the cycle number
   1.1 Approximate the first terms of the power series
   1.2 Extrapolate forward to $t'$
   1.3 Track to next time
   1.4 Choose $c$ which minimizes error at $t'$

2. Extrapolate to $t_0$
Cauchy – a picture

integrate around the triangles (polygons) to extrapolate to target time
A demonstration

and now we go live to Matlab
The 78 paths tracked to decompose a sphere
max condition number: $1.079 \times 10^9$
condnum vs time -- sphere
condnum vs logtime -- sphere
150 of the $\approx 1800$ paths tracked to decompose the whitney umbrella
max condition number: $2.9933 \times 10^{29}$
My point about endgames

Fact

You **have to** use multiprecision to use endgames, and you **totally** want to use endgames

- Double is not sufficient for a sphere
- Quadruple is not sufficient even for the Whitney Umbrella
Endgame resources

Some papers on endgames

- “Computing singular solutions to nonlinear analytic systems” [MSW91]
- “Computing Singular Solutions to Polynomial Systems” [MSW92]
- “A power series method for computing singular solutions to nonlinear analytic systems” [MSW92]
- “A method for tracking singular paths with application to the numerical irreducible decomposition” [SVW02]
- “A parallel endgame” [BHS2011]
Certification – general

Let

\[ \beta(f, x) = \|x - \text{Newtonstep}(f, x)\| \]

\[ \gamma(f, x) = \sup_{k \leq 2} \left\| \frac{Jf(x)^{-1} J^k f(x)}{k!} \right\|^{\frac{1}{k-1}} \]

\[ \alpha(f, x) = \beta(f, x) \gamma(f, x) \]

if

\[ \alpha(f, x) < \frac{13 - 3\sqrt{17}}{4} \approx 0.157671 \]

then Newton’s method will converge to some \( \xi \) with

\[ f(\xi) = 0 \] exactly
Certification – real

To check that $x = \bar{x}$ – that $x$ is an approximate solution to a real $\xi$:

1. Compute $\beta$, $\gamma$, $\alpha$
2. If $||x - \text{real}(x)|| > 2\beta$, not real
3. If $\alpha < 0.03$ && $||x - \text{real}(x)|| < \frac{1}{20\gamma}$, real
4. take a newton step, goto 1
Certification resources

- “Algorithm 921: alphaCertified: certifying solutions to polynomial systems” [HS11]
- alphaCertified – the software
- “Newton’s method estimates from data at one point” [Smale86]
- “Complexity and real computation” [BCSS98]
Problems with certification

Problem

*if your roots are tightly clustered, or your system is large at all, you simply have to use multiprecision*
Outline

We need multiprecision

Numbers in a computer
  Words
  Types
  Resources
  Concerns

Numerical nonlinear algebra uses multiprecision
  Plain old straight-line tracking
  Endgames
  Certification

Concerns
  Tracking
  Convention
Concerns with AMP tracking

- **∃ > 1 Condition number**
  Paul Breiding, MPI Leipzig, is working on condition numbers, using better things than classic ‘ratio of singular values’.
  See his work in `homotopycontinuation.jl`

- **Slow arithmetic is slow**
  Some libraries can be configured to use stack memory instead of heap, but you risk OOM (out of memory)
Concerns with convention

I had a long thread on GitHub with John Maddock (Boost.Multiprecision issue #75, concluded 20180916)

- titled “writing a complex class using mpfr_float that is precision-semantically equivalent to the new mpc type”
- Eventually discovered that I was caught in an x-y problem, and changed title to “precision-semantics policy”
- Revealed shortcomings in operator overloads and inconsistencies in et_on and et_off
A problematic test

BOOST_AUTO_TEST_CASE (complex_precision) {
    DefaultPrecision(30);
    bertini::mpfr_complex a(1,2);

    DefaultPrecision(50);
    bertini::mpfr_complex b(3,4);

    DefaultPrecision(70);
    bertini::mpfr_complex c(5,6);

    a = b+c;
    BOOST_CHECK_EQUAL ( Precision (a ) ,30);
}

**APPoS vs APPoT**

**Definition**

APPoS:
Assignment Preserves Precision of Source 🟢

```cpp
mp_float a(1,30);
mp_float b(2,40);
mp_float c(nan,80);

c = a*b;
// c.precision() == 40
```

**Definition**

APPoT:
Assignment Preserves Precision of Target 🟣

```cpp
mp_float a(1,30);
mp_float b(2,40);
mp_float c(nan,80);

c = a*b;
// c.precision() == 80
```
Be aware of your precision policy

APPoS
- "change precision of operands early"
- "change precision of targets late"
- one higher-precision operand will cause everything below it to be calculated at high precision

APPoT
- "just maintain precision of target and it’ll be fine"
- masks non-uniform precision. harder to debug? 😞
Conclusion

*Boost::Multiprecision uses APPoS. (as of 7f5594c)*

*that is,*

```cpp
mp_float a(1,30);
mp_float b(2,40);
mp_float c(nan,80);

c = a*b;
// c.precision() == 40
```

Thankyou, John Maddock, for working through that with me 🌱
Outline

We need multiprecision

Numbers in a computer
  Words
  Types
  Resources
  Concerns

Numerical nonlinear algebra uses multiprecision
  Plain old straight-line tracking
  Endgames
  Certification

Concerns
  Tracking
  Convention
Thank you!