An update on Mystika

Erik Wallace

Harbin Engineering University

September 10, 2019

Format change

For a negative number like -453, the former representation was:

```
10 0 1 9 9 9 9 9 9 5 4 7
```

the new representation is:

```
10 0 1 0 0 0 0 0 0 4 5 3
```

Note: the first axis is still reserved for the places of the bignum.

Supported primitive operators

red	/	rdf	$\not=$	scn	\	scf	$\not<$
dot	.	out	.$\circ$	pop	$\ddot{\star}$	rop	$\ddot{\circ}$

Supported primitive functions

ima	11○	rea	9○	cnj/add	+	sub	−
mul	×	cat	⍪	rav	,	trn	⍉
rot	⌽	rof	⊖	pic	⊃	sqd	⌷
eql	=	neq	≠	leq	≤	geq	≥
gth	>	lth	<	flo/min	⌊	cel/max	⌈
abs/mod	|	rho	⍴	eps	⍳	ind	⍳
rol	?	tke/mix	↑	drp/spl	↓	div	÷
exp	*	cis	¯12○	sin	1○	snh	5○
tan	3○	pie	○				

Other functions

- AES
- SHA
- Modular Exponentiation
- Multiplication of big polys
- Pseudo prime generation (Also Sophie Germain primes)
- Extending a smallnum array to a bignum array
- Extending the number of places
- Moving the radix point
- Base change
- Square root
- “Decode”

add/sub times

16 places in base 256 (1 pair= two 128 bit numbers)

pairs	100	200	400	800
µs/pair	4.45	3.01	2.70	2.58
MB/s	7.19	10.64	11.87	12.41

32 places in base 256 (1 pair= two 256 bit numbers)

pairs	100	200	400	800
µs/pair	5.55	5.08	4.77	4.77
MB/s	11.54	12.61	13.43	13.43

64 places in base 256 (1 pair= two 512 bit numbers)

pairs	100	200	400	800
µs/pair	10.15	9.38	9.06	9.06
MB/s	12.60	13.65	14.12	14.12

mul times

16 places in base 256 (1 pair= two 128 bit numbers)

pairs	100	200	400	800
µs/pair	15.625	19.0625	18.75	19.375
MB/s	2.05	1.68	1.71	1.65

32 places in base 256 (1 pair= two 256 bit numbers)

pairs	100	200	400	800
µs/pair	39.38	40.63	41.88	45.63
MB/s	1.635	1.57	1.53	1.40

64 places in base 256 (1 pair= two 512 bit numbers)

pairs	100	200	400	800
µs/pair	86.25	90	140	132.5
MB/s	1.48	1.42	0.91	0.96

in0 times

16 places in base 256 (1 pair= two 128 bit numbers)

pairs	100	200	400	800
ms/pair	0.52	0.45	0.39	0.37
KB/s	30.77	35.95	40.76	46.21

32 places in base 256 (1 pair= two 256 bit numbers)

pairs	100	200	400	800
ms/pair	1.01	0.84	0.78	0.70
KB/s	31.68	38.32	42.24	45.88

64 places in base 256 (1 pair= two 512 bit numbers)

pairs	100	200	400	800
ms/pair	1.86	1.72	1.62	1.53
KB/s	34.41	37.21	39.38	41.73

AES times

AES 128 (1 block = 128 bits)

blocks	8192	16384	32768	65536	131072	262144
µs/block	44.5	40.41	37.57	36.42	36.20	35.45
Mbps	2.87	3.17	3.41	3.51	3.54	3.61

AES 196 (1 block = 128 bits)

blocks	8192	16384	32768	65536	131072	262144
µs/block	51.88	47.30	44.74	43.49	42.82	42.53
Mbps	2.47	2.71	2.86	2.94	2.99	3.01

AES 256 (1 block = 128 bits)

blocks	8192	16384	32768	65536	131072	262144
µs/block	59.33	53.77	51.78	50.47	49.87	49.74
Mbps	2.16	2.38	2.47	2.54	2.57	2.57

SHA times

SHA 256 (1 message = 256 bits)

messages	512	1024	2048	4096	8192
µs/message	247.07	215.33	175.54	181.27	179.81
Mbps	1.04	1.19	1.46	1.41	1.43

SHA 384 (1 message = 512 bits)

messages	512	1024	2048	4096	8192
µs/message	683.59	546.88	570.8	543.7	533.57
Mbps	0.75	0.93	0.90	0.94	0.95

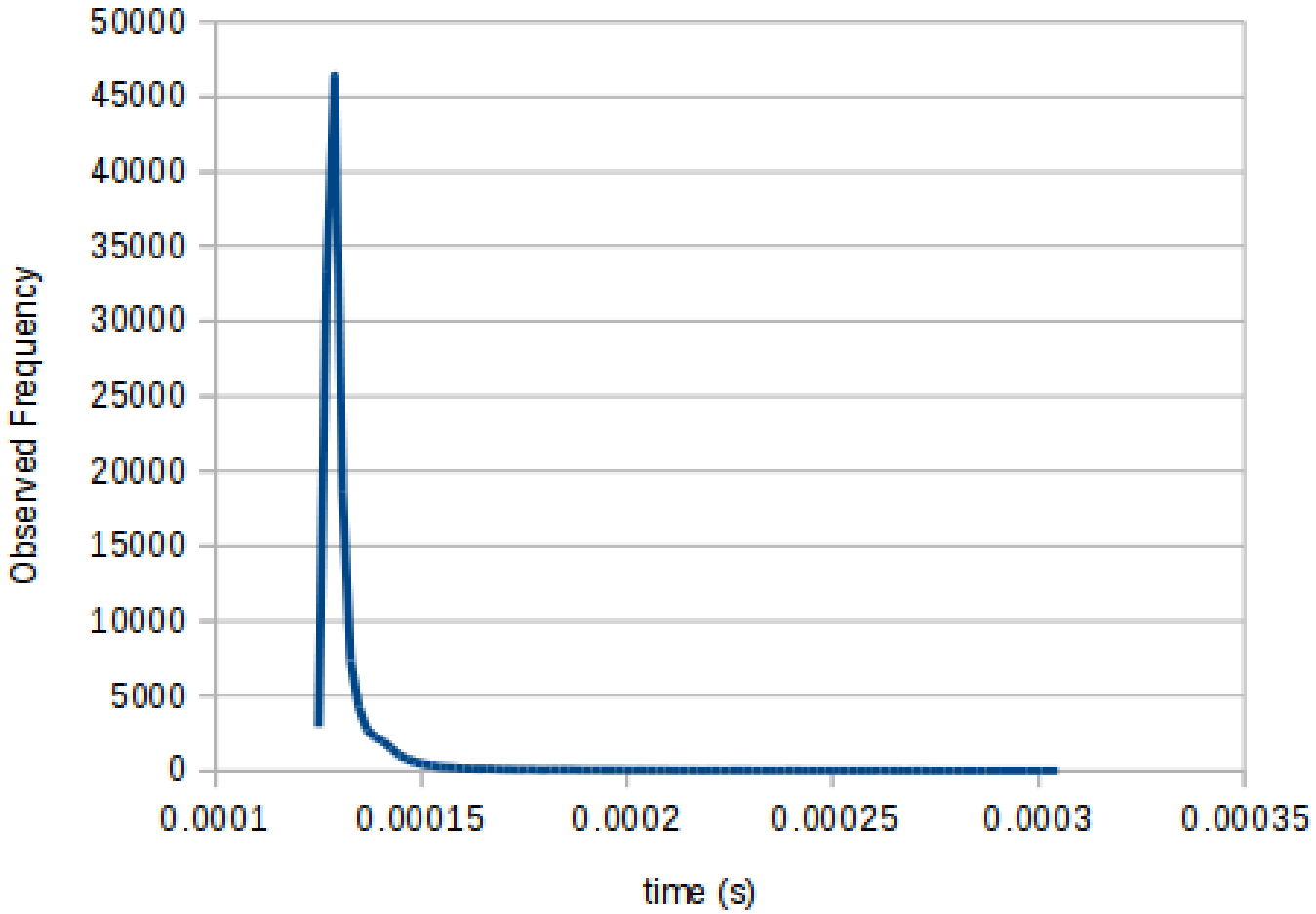
SHA 512 (1 message = 512 bits)

messages	512	1024	2048	4096	8192
µs/message	685.54	551.76	570.8	545.17	544.19
Mbps	0.75	0.93	0.90	0.94	0.94

My Hardware

CPU:	Intel Core i5-3320M 2.60 GHz
Memory:	8GB DDR3 1600 MHz

AES timing distribution



Some numerical analysis

Guideline

Let $f(x)$ be a function with non-zero slope at $x = a$, and let $|b - a| \sim \varepsilon$. Then $|f(b) - f(a)| \sim |f'(a)|\varepsilon$.

Some numerical analysis

Guideline

Let $f(x)$ be a function with non-zero slope at $x = a$, and let $|b - a| \sim \varepsilon$. Then $|f(b) - f(a)| \sim |f'(a)|\varepsilon$.

Examples:

1. $f(x) = x^2$, $a = 3$, and $b = 3.001$ so $\varepsilon = 0.001$ and $f'(x) = 2x$.

Some numerical analysis

Guideline

Let $f(x)$ be a function with non-zero slope at $x = a$, and let $|b - a| \sim \varepsilon$. Then $|f(b) - f(a)| \sim |f'(a)|\varepsilon$.

Examples:

1. $f(x) = x^2$, $a = 3$, and $b = 3.001$ so $\varepsilon = 0.001$ and $f'(x) = 2x$.
   $|3.001^2 - 3^2| = 0.006001$ and $2 \cdot 3 \cdot 0.001 = 0.006$.

Some numerical analysis

Guideline

Let $f(x)$ be a function with non-zero slope at $x = a$, and let $|b - a| \sim \varepsilon$. Then $|f(b) - f(a)| \sim |f'(a)|\varepsilon$.

Examples:

1. $f(x) = x^2$, $a = 3$, and $b = 3.001$ so $\varepsilon = 0.001$ and $f'(x) = 2x$.
   $|3.001^2 - 3^2| = 0.006001$ and $2 \cdot 3 \cdot 0.001 = 0.006$.
2. $f(x) = 1/x$, $a = \frac{1}{\pi}$, and $b = 0.318$ so $\varepsilon \sim 0.0003$ and $f'(x) = -\frac{1}{x^2}$.

Some numerical analysis

Guideline

Let $f(x)$ be a function with non-zero slope at $x = a$, and let $|b - a| \sim \varepsilon$. Then $|f(b) - f(a)| \sim |f'(a)|\varepsilon$.

Examples:

1. $f(x) = x^2$, $a = 3$, and $b = 3.001$ so $\varepsilon = 0.001$ and $f'(x) = 2x$.
   $|3.001^2 - 3^2| = 0.006001$ and $2 \cdot 3 \cdot 0.001 = 0.006$.
2. $f(x) = 1/x$, $a = \frac{1}{\pi}$, and $b = 0.318$ so $\varepsilon \sim 0.0003$ and $f'(x) = -\frac{1}{x^2}$.
   $\left|\frac{1}{\pi} - \frac{1}{0.318}\right| \approx 0.00306$ and $\pi^2 \cdot 0.0003 \approx 0.00296$.

Some numerical analysis

Guideline

Let $f(x)$ be a function with non-zero slope at $x = a$, and let $|b - a| \sim \varepsilon$. Then $|f(b) - f(a)| \sim |f'(a)|\varepsilon$.

Examples:

1. $f(x) = x^2$, $a = 3$, and $b = 3.001$ so $\varepsilon = 0.001$ and $f'(x) = 2x$.
   $|3.001^2 - 3^2| = 0.006001$ and $2 \cdot 3 \cdot 0.001 = 0.006$.
2. $f(x) = 1/x$, $a = \frac{1}{\pi}$, and $b = 0.318$ so $\varepsilon \sim 0.0003$ and $f'(x) = -\frac{1}{x^2}$.
   $\left|\frac{1}{\pi} - \frac{1}{0.318}\right| \approx 0.00306$ and $\pi^2 \cdot 0.0003 \approx 0.00296$.
3. $f(x) = x^{2^n}$, $a = 1$, and $b = 0.9999999999$ so $\varepsilon \sim 0.0000000001$.

n	16	24	32
b	0.9999934464	0.9983236848	0.6508365359

Sage vs. Mystika

Consider the task of calculating $e^{29.5}$ to 20 decimal places.

```
sage: N(e^29.5,digits=40)
---------------------------------------------------------------------------
TypeError                                 Traceback (most recent call last)
<ipython-input-32-5473c939fb73> in <module>()
----> 1 N(e**RealNumber('29.5'),digits=Integer(40))

/usr/lib/sagemath/local/lib/python2.7/site-packages/sage/misc/functional.py in numerical_approx(x, prec, digits, algorithm)
   1293         return numerical_approx_generic(x, prec)
   1294     else:
-> 1295         return n(prec, algorithm=algorithm)
   1296
   1297 n = numerical_approx

sage/structure/element.pyx in sage.structure.element.Element.numerical_approx (/usr/lib/sagemath//src/build/cythonized/sage/structure/element.c:7806)()

sage/arith/numerical_approx.pyx in sage.arith.numerical_approx.numerical_approx_generic (/usr/lib/sagemath//src/build/cythonized/sage/arith/numerical_approx.c:2662)()

TypeError: cannot approximate to a precision of 137 bits, use at most 53 bits
```

Sage vs. Mystika

Consider the task of calculating $e^{29.5}$ to 20 decimal places.

```
sage: N(e^29.5,digits=40)
---------------------------------------------------------------------------
TypeError                                 Traceback (most recent call last)
<ipython-input-32-5473c939fb73> in <module>()
----> 1 N(e**RealNumber('29.5'),digits=Integer(40))

/usr/lib/sagemath/local/lib/python2.7/site-packages/sage/misc/functional.py in numerical_approx(x, prec, digits, algorithm)
   1293         return numerical_approx_generic(x, prec)
   1294     else:
-> 1295         return n(prec, algorithm=algorithm)
   1296
   1297 n = numerical_approx

sage/structure/element.pyx in sage.structure.element.Element.numerical_approx (/usr/lib/sagemath//src/build/cythonized/sage/structure/element.c:7806)()

sage/arith/numerical_approx.pyx in sage.arith.numerical_approx.numerical_approx_generic (/usr/lib/sagemath//src/build/cythonized/sage/arith/numerical_approx.c:2662)()

TypeError: cannot approximate to a precision of 137 bits, use at most 53 bits
```

Translation into plain English:

"We didn't bother implementing e^x for bignums. Sage does have the capability of computing the Taylor series, so you could try plugging into that, otherwise you will have to use some other software such as Mystika.

Regards,

The Sage Developers"

The Taylor remainder term of e^x.

Let $|x| \leq R \leq \frac{N}{2}$. Then

$$\left| e^x - \sum_{k=0}^{N} \frac{1}{k!} x^k \right| = \left| \sum_{k=N+1}^{\infty} \frac{1}{k!} x^k \right| \leq \frac{2R^{N+1}}{(N+1)!}$$

Now consider the task of calculating $e^{29.5}$ to 40 digits.

The Taylor remainder term of e^x.

Let $|x| \leq R \leq \frac{N}{2}$. Then

$$\left| e^x - \sum_{k=0}^{N} \frac{1}{k!} x^k \right| = \left| \sum_{k=N+1}^{\infty} \frac{1}{k!} x^k \right| \leq \frac{2R^{N+1}}{(N+1)!}$$

Now consider the task of calculating $e^{29.5}$ to 40 digits.

The Naive approach (just plugging in):

Take $R = 30$ and $N \geq 60$.

We need 27 decimal places, so we must have $\frac{2 \cdot 30^{N+1}}{(N+1)!} < 10^{-27}$, hence $N \geq 129$. Ugh!

Implementing e^x

1. Reduce to $\frac{x}{2^s} \in [\frac{1}{b^2}, \frac{1}{b})$ using repeated division by 2.

Implementing e^x

1. Reduce to $\frac{x}{2^s} \in [\frac{1}{b^2}, \frac{1}{b})$ using repeated division by 2.
2. Plug into the taylor series with $\frac{x}{2^s}$.

Implementing e^x

1. Reduce to $\frac{x}{2^s} \in [\frac{1}{b^2}, \frac{1}{b})$ using repeated division by 2.
2. Plug into the taylor series with $\frac{x}{2^s}$.
3. Square the result s times. This works because

$$\left(e^{\frac{x}{2^s}}\right)^{2^s} = e^{\frac{x}{2^s} \cdot 2^s} = e^x$$

Implementing e^x

1. Reduce to $\frac{x}{2^s} \in [\frac{1}{b^2}, \frac{1}{b})$ using repeated division by 2.
2. Plug into the taylor series with $\frac{x}{2^s}$.
3. Square the result s times. This works because

$$\left(e^{\frac{x}{2^s}}\right)^{2^s} = e^{\frac{x}{2^s} \cdot 2^s} = e^x$$

In the previous example mystika computes $s = 10$ and $r = 22$. Would you rather compute 22 terms of a Taylor series or 129?

Implementing e^x

1. Reduce to $\frac{x}{2^s} \in [\frac{1}{b^2}, \frac{1}{b})$ using repeated division by 2.
2. Plug into the taylor series with $\frac{x}{2^s}$.
3. Square the result s times. This works because

$$\left(e^{\frac{x}{2^s}}\right)^{2^s} = e^{\frac{x}{2^s} \cdot 2^s} = e^x$$

In the previous example mystika computes $s = 10$ and $r = 22$. Would you rather compute 22 terms of a Taylor series or 129?

```
      exp 10 1 0,¯40↑2 9 5
10 27 0 6 4 8 1 6 7 4 4 7 7 9 3 4 3 2 0 2 1 7 9 2 1 4 4 2 2 1 8 5 1 6 3 0 9 9 7 9 6 8 9
```

No error message. Mystika wins!

Further directions

1. The bignum library and crypto library will remain under GPLv3.
2. Some AI support may be added to mystika (closed source).