Challenges in Reliable Computing (@ SE Level)

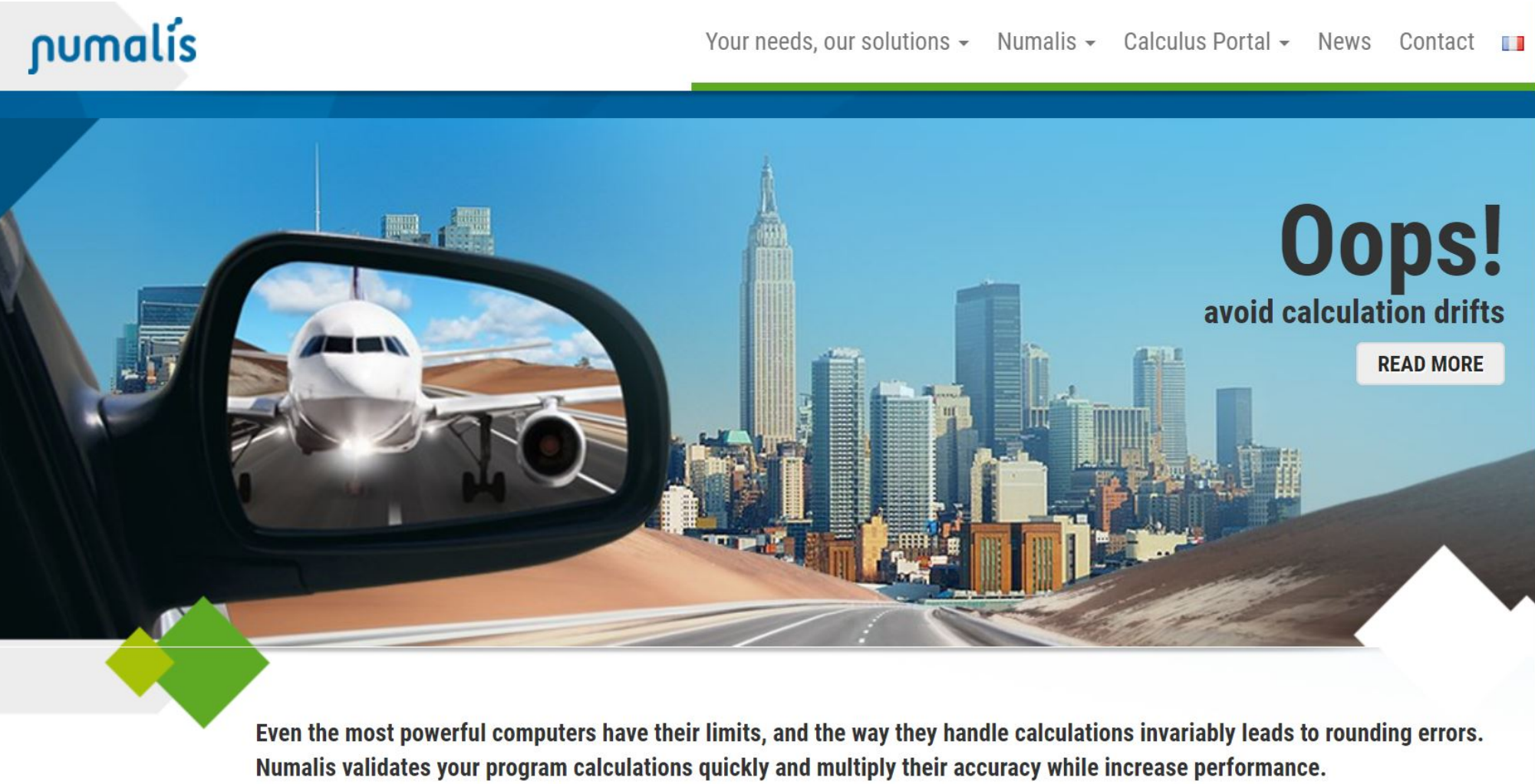
MATTHIEU MARTEL

University of Perpignan & Numalis
Laboratory of Mathematics and Physics (LAMPS)

matthieu.martel@univ-perp.fr

Numerical Computations are Everywhere...

...Need of Reliability !



Challenges in Reliable Computing (@ SE Level)

- Numerical accuracy determination
- Automatic accuracy optimization
- Precision tuning
- Standard benchmarks & metrics to compare tools
- Specific features in programming languages

[A. Ioualalen, M. Martel, N. Normand, *An Overview of Numalis Software Suite for Reliable Numerical Computation*, ISSRE'17, Best Industry Paper]

Image Transmission Protocol from Satellite to Earth

```
29  eq3d LinearReg_3d(vec3d* vect, unsigned int n)
30  {
31      float sumxi2 = 0; float sumxiyi = 0;
32      float sumxi = 0; float sumyi2 = 0;
33      float sumyi = 0; float sumxizi = 0;
34      float sumyizi = 0; float sumzi = 0;
35      vec3d pt; unsigned int i; eq3d res;
36
37      /* Multiple regression using least square method */
38      for (i = 0; i < n; ++i) {
39          pt = vect[i];
40          sumxi2 += pt.x * pt.x; sumxiyi += pt.x * pt.y;
41          sumxi += pt.x; sumyi2 += pt.y * pt.y;
42          sumyi += pt.y; sumxizi += pt.x * pt.z;
43          sumyizi += pt.y * pt.z; sumzi += pt.z;
44      }
45
46      /* Equation of plane : ax + by + c = z */
47      res.c = ((sumxi * sumxizi - sumxi2 * sumzi) * (-sumxiyi * sumyi + sumxi * sumyi2) -
48              ((-sumxi2 * sumyi + sumxi * sumxiyi) * (sumxi * sumyizi - sumxiyi * sumzi))) /
49          (-((-sumxi2 * sumyi + sumxi * sumxiyi) * (-sumxiyi * n + sumxi * sumyi)) +
50           (-sumxiyi * sumyi + sumxi * sumyi2) * (-sumxi2 * n + sumxi * sumxi));
51      res.b = ((sumxi * sumyizi - sumxiyi * sumzi) - (-sumxiyi * n + sumxi * sumyi) * res.c) /
52          (-sumxiyi * sumyi + sumxi * sumyi2);
53      res.a = (sumzi - sumyi * res.b - n * res.c) / sumxi;
54
55      return res;
56  }
```

European Cooperation for Space Standardization
http://ecss.nl/

Image Transmission Protocol from Satellite to Earth

```
29  eq3d LinearReg_3d(vec3d* vect, unsigned int n)
30  {
31      float sumxi2 = 0; float sumxiyi = 0;
32      float sumxi = 0; float sumyi2 = 0;
33      float sumyi = 0; float sumxizi = 0;
34      float sumyizi = 0; float sumzi = 0;
35      vec3d pt; unsigned int i; eq3d res;
36
37      /* Multiple regression using least square method */
38      for (i = 0; i < n; ++i) {
39          pt = vect[i];
40          sumxi2 += pt.x * pt.x; sumxiyi += pt.x * pt.y;
41          sumxi += pt.x; sumyi2 += pt.y * pt.y;
42          sumyi += pt.y; sumxizi += pt.x * pt.z;
43          sumyizi += pt.y * pt.z; sumzi += pt.z;
44      }
45
46      /* Equation of plane : ax + by + c = z */
47      res.c = ((sumxi * sumxizi - sumxi2 * sumzi) * (-sumxiyi * sumyi + sumxi * sumyi2) -
48              ((-sumxi2 * sumyi + sumxi * sumxiyi) * (sumxi * sumyizi - sumxiyi * sumzi))) /
49          (-((-sumxi2 * sumyi + sumxi * sumxiyi) * (-sumxiyi * n + sumxi * sumyi)) +
50           (-sumxiyi * sumyi + sumxi * sumyi2) * (-sumxi2 * n + sumxi * sumxi));
51      res.b = ((sumxi * sumyizi - sumxiyi * sumzi) - (-sumxiyi * n + sumxi * sumyi) * res.c) /
52          (-sumxiyi * sumyi + sumxi * sumyi2);
53      res.a = (sumzi - sumyi * res.b - n * res.c) / sumxi;
54
55      return res;
56  }
```

Best formats?

Best writing?

Accuracy?

Challenges in Reliable Computing (@ SE Level)

- **Numerical accuracy determination**
- Automatic accuracy optimization
- Precision tuning
- Standard benchmarks & metrics to compare tools
- Specific features in programming languages

Numerical Accuracy Determination (Static)

❑ **Fluctuat**

Affine forms, abstract interpretation, some loops

[E. Goubault, *Static Analysis by Abstract Interpretation of Numerical Programs and Systems, and FLUCTUAT*, SAS'13]

❑ **Rosa / Daisy**

Affine forms, SMT solvers, very special loops

[E. Darulova, A. Izycheva, F. Nasir, F. Ritter, H. Becker, R. Bastian, *Daisy - Framework for Analysis and Optimization of Numerical Programs*, TACAS'18]

❑ **FP-Taylor**

Taylor series, no loops

[A. Solovyev, C. Jacobsen, Z. Rakamaric, G. Gopalakrishnan : *Rigorous Estimation of Floating-Point Round-off Errors with Symbolic Taylor Expansions*, FM'15]

From [E. Darulova et al, Towards a Compiler for Reals]

Table I. Absolute errors computed by Rosa, Fluctuat and FPTaylor for double-precision floating-point arithmetic. (r) marks refactored benchmarks, (e) marks benchmarks with additional input errors

benchmark	Rosa	Fluctuat	Fluctuat (subdiv)	FPTaylor	Rosa	Fluctuat	Fluctuat (subdiv)	FPTaylor
with roundoff errors only								
doppler	4.15e-13	3.90e-13	1.54e-13	1.35e-13	8	1	2	7
doppler (r)	2.42e-13	3.90e-13	1.40e-13	1.35e-13	7	1	2	7
jet	5.33e-9	4.08e-8	2.10e-11	1.17e-11	95	1	2	12
jet (r)	4.91e-9	4.08e-8	1.88e-11	1.17e-11	77	1	2	12
rigidBody	3.65e-11	3.65e-11	3.65e-11	3.61e-11	1	1	2	6
rigidBody (r)	3.65e-11	3.65e-11	3.65e-11	3.61e-11	1	1	2	5
sine	5.74e-16	7.97e-16	7.41e-16	5.52e-16	2	1	1	6
sineOrder3	9.96e-16	1.15e-15	1.09e-15	8.90e-16	1	1	1	4
sqroot	2.87e-13	3.21e-13	3.21e-13	2.87e-13	1	1	1	7
turbine1	5.99e-14	9.20e-14	2.21e-14	2.11e-14	5	1	2	8
turbine1 (r)	5.15e-14	9.26e-14	2.21e-14	2.11e-14	2	1	2	8
turbine2	7.68e-14	1.29e-13	2.87e-14	2.62e-14	2	1	2	6
turbine2 (r)	6.30e-14	1.34e-13	2.87e-14	2.62e-14	1	1	2	7
turbine3	4.62e-14	6.99e-14	1.34e-14	1.55e-14	4	1	2	7
turbine3 (r)	4.02e-14	7.03e-14	1.32e-14	1.55e-14	2	1	2	7
				total	209	15	27	109
				total (-jet)	37	13	23	85

From [E. Darulova et al, Towards a Compiler for Reals]

with input errors								
doppler (re)	1.83e-11	5.45e-11	2.21e-11	1.82e-11	13	1	2	7
jet (re)	3.36e-7	4.67e-4	1.37e-7	3.85e-8	76	1	2	13
turbine1 (re)	4.61e-10	1.82e-9	6.02e-10	4.61e-10	2	1	2	8
turbine2 (re)	5.87e-10	2.82e-9	6.14e-10	5.86e-10	1	1	2	10
turbine3 (re)	3.33e-10	1.24e-9	2.53e-10	3.32e-10	2	1	2	8
rigidBody (re)	1.50e-7	1.50e-7	1.50e-7	1.50e-7	2	1	2	6
sine (e)	1.00e-11	2.09e-11	1.01e-11	1.00e-11	2	1	1	6
				total	98	7	13	58
				total (-jet)	22	6	11	45
with input constraint								
doppler (r)	1.76e-14	1.09e-13	4.84e-14	1.57e-14	7	1	2	5
doppler (re)	4.67e-13	1.37e-11	6.28e-12	4.77e-13	12	1	2	11
jet (r)	4.91e-9	4.08e-8	1.88e-11	1.48e-11	84	1	2	1731
jet (re)	3.36e-7	4.67e-4	1.37e-7	-	81	1	2	-
rigidBody (r)	1.66e-11	3.65e-11	3.34e-11	1.52e-11	13	1	2	61
rigidBody (re)	8.84e-8	1.50e-7	1.15e-7	6.78e-8	13	1	2	284
turbine1 (r)	4.26e-14	8.66e-14	2.21e-14	2.48e-14	3	1	2	6
turbine1 (re)	4.61e-10	1.94e-9	6.51e-10	4.59e-10	2	1	2	109
turbine2 (r)	5.26e-14	1.45e-13	2.44e-14	2.92e-14	2	1	2	6
turbine2 (re)	5.87e-10	3.02e-9	6.33e-10	4.84e-10	2	1	2	37
turbine3 (r)	3.55e-14	7.32e-14	9.50e-15	1.49e-14	5	1	2	11
turbine3 (re)	3.33e-10	1.33e-09	2.30e-10	2.76e-10	5	1	2	316

Numerical Accuracy Determination (Dynamic)

- **Cadna**

Stochastic arithmetic, overloading of operators

[J.-L. Lamotte, J.-M. Chesneaux, F. Jézéquel : *CADNA_C : A version of CADNA for use with C or C++ programs*, Computer Physics Communications, 2010]

- **Verificarlo**

Stochastic arithmetic, integrated in compiler (CLang)

[Ch. Denis, P. de Oliveira Castro, E. Petit : *Verificarlo : Checking Floating Point Accuracy through Monte Carlo Arithmetic*, ARITH'16]

Challenges in Reliable Computing (@ SE Level)

- Numerical accuracy determination
- **Automatic accuracy optimization**
- Precision tuning
- Standard benchmarks & metrics to compare tools
- Specific features in programming languages

Automatic Accuracy Optimization

- **Salsa**

Error bounds by static analysis, interprocedural code transformation

[N. Damouche, M. Martel, A. Chapoutot, *Improving the numerical accuracy of programs by automatic transformation*, STTT, 2017]

- **Herbie**

Error bounds by dynamic analysis, arithmetic expressions only

[P. Panchekha, A. Sanchez-Stern, J. R. Wilcox, Z. Tatlock, *Automatically improving accuracy for floating point expressions*, PLDI'15]

Example of Interprocedural Transformation

```
double main(){
e0 = 0.0; t = 0.0; m = 150.0;
i0 = 0.0; dt = 0.2; c = 0.0;
while (t < 10.0) {
  e1 = c - m ;
  p = prop(e1) ;
  i = integral(i0,m,c,dt) ;
  d = deriv(e1,e0) ;
  r = p + i + d ;
  m = m + r * 0.01 ;
  t = t + dt ;
  e0 = e1 ;
  }
 return m ;
}
double prop(double e1){
  kp = 9.4514 ;
  res = kp * e1 ;
  return res ;
}
double integral(double ii, double mm,double c
                double ddtt){
  ki = 0.69006 ;
  res = ii + (ki * ddtt * (cc - mm)) ;
  return res ;
}
double deriv(double e1, double e0){
  kd = 2.8454 ;
  invdt = 5.0 ;
  res = kd * (e1 - e0) * invdt ;
  return res ;
}
```

```
double main() {
e0 = 0.0; t = 0.0; m = 0.150e3;
while (t < 10.0) {
  e1 = -0.150e3 ;
  kp = 0.94514e1 ;
  res1 = (kp * -0.150e3) ;
  ii = 0.0 ;
  mm = 0.150e3 ;
  cc = 0.0 ;
  ddtt = 0.2 ;
  ki = 0.69006 ;
  res2 = (ii + ((ki * ddtt) * (cc - m))) ;
  kd = 0.28454e1 ;
  invdt = 0.5e1 ;
  res3 = ((kd * (-0.150e3 - e0)) * invdt) ;
  m = (((0.1e-1 * (-0.213405e4)) + ((0.1e-1
    * (-0.207018e2)) + (0.1e-1 * (-0.141771e
  t = (0.2 + t) ;
  e0 = -0.150e3
  }
return m ;
}
```

```
double main() {
e0 = 0.0;
t = 0.0;
m = 0.150e3 ;
while (t < 10.0) {
  e1 = (c - m) ;
  p = propTMP_2() ;
  i = integralTMP_7() ;
  d = derivTMP_10() ;
  m = (((+0.1e-1 * (-0.213405e4)) + ((+0.1e
    * (-0.207018e2)) + (+0.1e-1 * (-0.14177
  t = (0.2 + t) ;
  e0 = -0.150e3
 }
return m ;
}
double derivTMP_10() {
  TMP_8 = -0.150e3 ;
  res = (+0.28454e1 * (-0.150e3 * 0.5e1))
  return res ;
}
double integralTMP_7() {
  TMP_4 = 0.150e3 ;
  TMP_6 = 0.2 ;
  res = (TMP_6 * (TMP_4 * 0.69006))
  return res ;
}
double propTMP_2() {
  TMP_1 = -0.150e3 ;
  res = -0.141771e4
  return res ;
}
```

From [N. Damouche, M. Martel, CODIT' 18]

Code	Initial error	Error after transformation	% of improvement	Code size$_o$ (Bytes)	Code size$_t$ (Bytes)	$\frac{codesize_o}{codesize_t}$
Odometry	$0.5794e^{-14}$	$0.4788e^{-14}$	17.36	900	1439	0.62
Rocket	$2.8569e^{-14}$	$1.3977e^{-14}$	51.07	1911	1624	1.1
Runge-Kutta 4	$4.6666e^{-13}$	$2.9226e^{-13}$	37.37	678	1649	0.41
Jacobi	$2.9142e^{-16}$	$1.7258e^{-16}$	40.77	1032	1959	0.52
PID	$0.2059e^{-13}$	$0.2221e^{-13}$	7.86	623	1068	0.58

Code	Original Program			Transformed Program (bal. tree)		
	Initial error	Transfo. error	Transfo. time(s)	Initial error	Transfo. error	Transfo. time(s)
Odometry	$4.9801e^{-13}$	$1.9829e^{-13}$	0.098	$2.5637e^{-14}$	$1.9829e^{-15}$	0.090
Runge-Kutta 4	$4.6666e^{-13}$	$2.9226e^{-13}$	0.098	$4.6047e^{-14}$	$2.9253e^{-15}$	0.067
Jacobi	$2.9142e^{-16}$	$1.7258e^{-16}$	1.098	$2.5658e^{-16}$	$1.5517e^{-16}$	0.125
PID	$4.6680e^{-14}$	$2.1346e^{-16}$	0.125	$2.5637e^{-15}$	$1.9829e^{-16}$	0.098

Code	Original Program			Transformed Program (nb. op)		
	Initial error	Transfo. error	Transfo. time(s)	Initial error	Transfo. error	Transfo. time(s)
Odometry	$4.9801e^{-13}$	$1.9829e^{-13}$	0.098	$2.4011e^{-14}$	$1.8574e^{-15}$	0.091
Runge-Kutta 4	$4.6666e^{-13}$	$2.9226e^{-13}$	0.098	$4.6805e^{-14}$	$2.9312e^{-16}$	0.751
Jacobi	$2.9142e^{-16}$	$1.7258e^{-16}$	1.098	$1.7176e^{-16}$	$4.4408e^{-17}$	0.115
PID	$4.6680e^{-14}$	$2.1346e^{-16}$	0.125	$2.8574e^{-15}$	$1.5211e^{-15}$	0.100

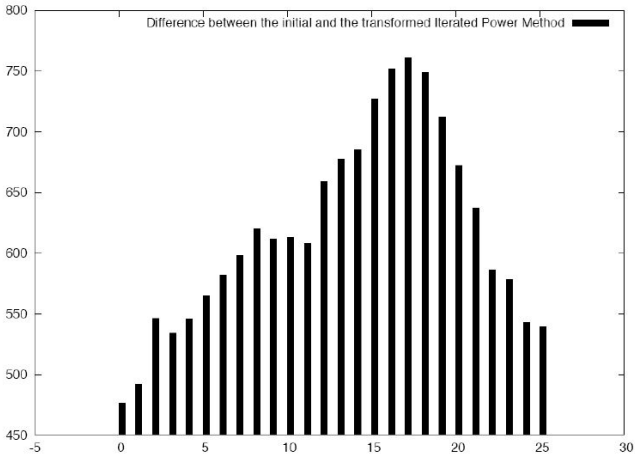
Code	Original Program			Transformed Program (bal. tree & nb. op.)		
	Initial error	Transfo. error	Transfo. time(s)	Initial error	Transfo. error	Transfo. time(s)
Odometry	$4.9801e^{-13}$	$1.9829e^{-13}$	0.098	$1.5847e^{-14}$	$1.9229e^{-15}$	0.080
Runge-Kutta 4	$4.6666e^{-13}$	$2.9226e^{-13}$	0.098	$3.6095e^{-14}$	$2.9253e^{-15}$	0.050
Jacobi	$2.9142e^{-16}$	$1.7258e^{-16}$	1.098	$1.5658e^{-16}$	$1.5017e^{-16}$	0.120
PID	$4.6680e^{-14}$	$2.1346e^{-16}$	0.125	$2.5622e^{-15}$	$1.9829e^{-16}$	0.070

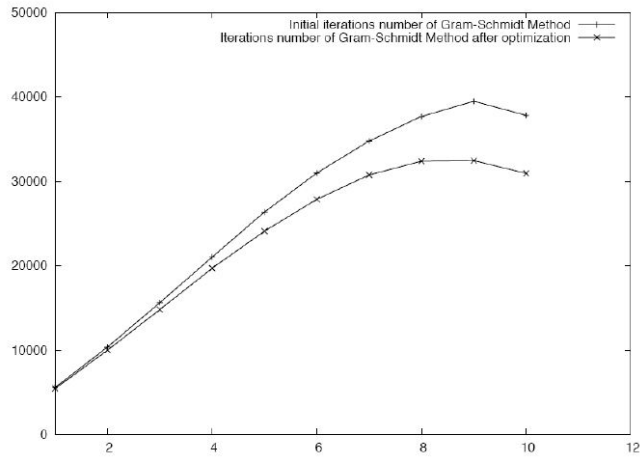
Topics Related to Program Transformation

❑ Convergence acceleration

Iterative methods converge faster when computations are accurate

[N. Damouche, M. Martel, A. Chapoutot, *Numerical program optimisation by automatic improvement of the accuracy of computations*, IJIEI, 2018]





❑ Formal proofs

Generate certificates for the correctness of Salsa transformations

[D. Delmas, M. Martel, A. Werey, *Certifying Expression Optimization for Numerical Accuracy in the Embedded Software Context*, Submitted'18]

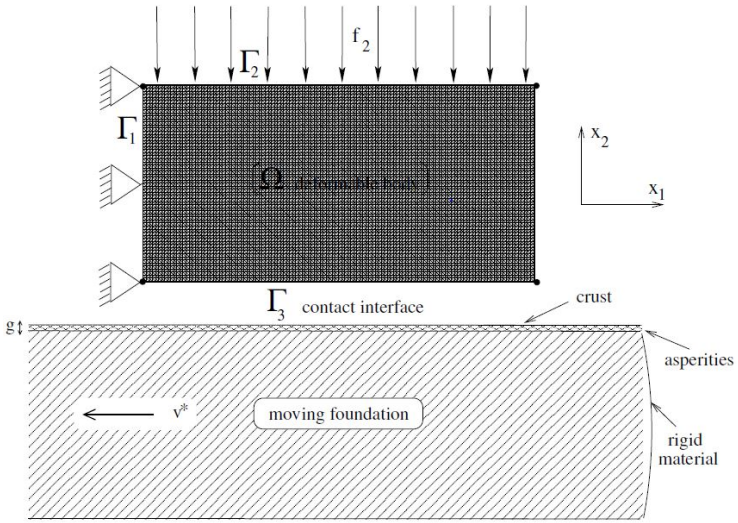
Topics Related to Program Transformation

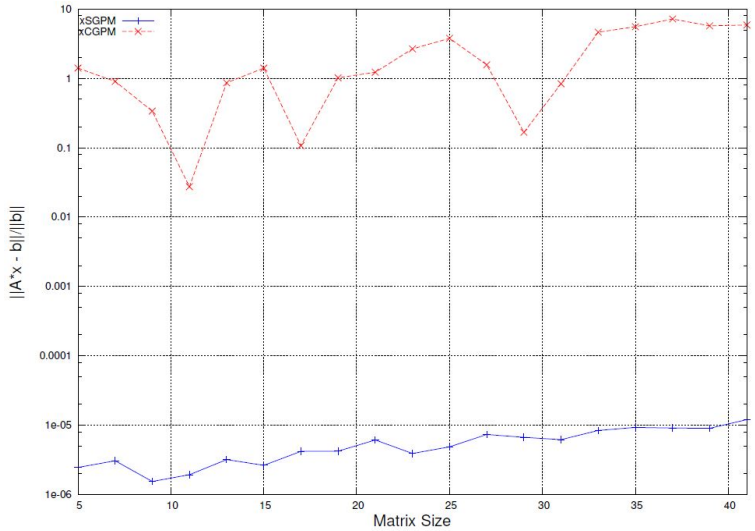
❑ Code synthesis

Linear algebra in fixed-point arithmetic, accurate codes for mechanics

[M. Martel, A. Najahi, G. Revy, *Trade-offs of certified fixed-point code synthesis for linear algebra basic blocks* JSA, 2017]

[M. Barboteu, N. Djehaf, [M. Martel, *Toward the Synthesis of Gauss Pivoting Code for Linear Systems Resolution : Application Mechanical Problems* submitted'18]



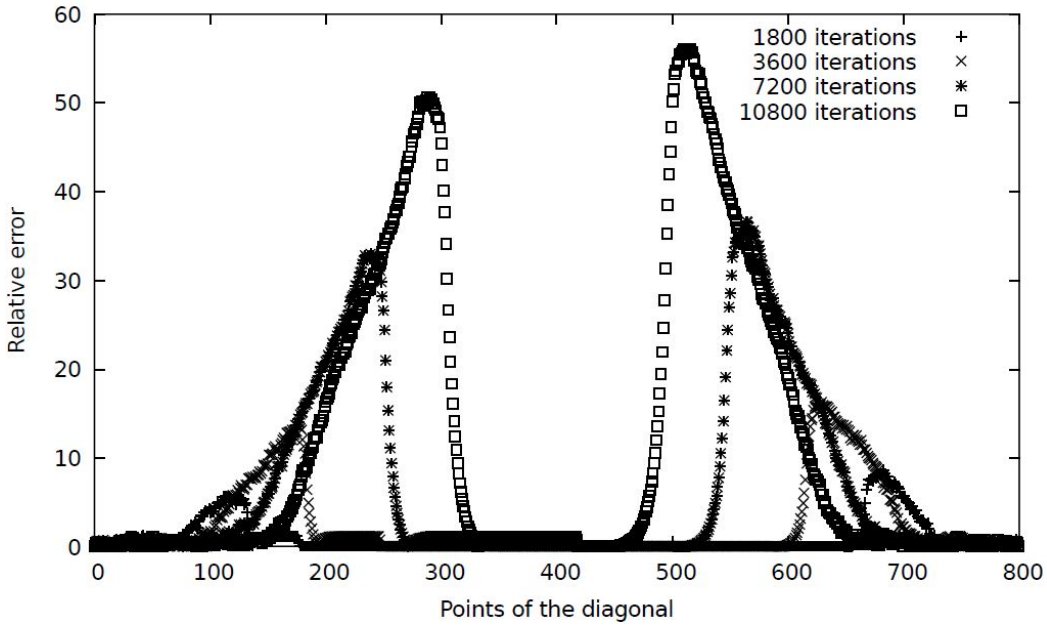


Topics Related to Program Transformation

- **Parallel codes**

Specialization by process, accurate reductions, mapping for accuracy

[F. Benmouhoub, N. Damouche, M. Martel, *Improving the Numerical Accuracy of High Performance Computing Programs by Process Specialization*, TNC'18]



Challenges in Reliable Computing (@ SE Level)

- Numerical accuracy determination
- Automatic accuracy optimization
- **Precision tuning**
- Standard benchmarks & metrics to compare tools
- Specific features in programming languages

Precision Tuning

Input : A program, accuracy requirements for outputs

Output : Least format needed for each variable/intermediary result

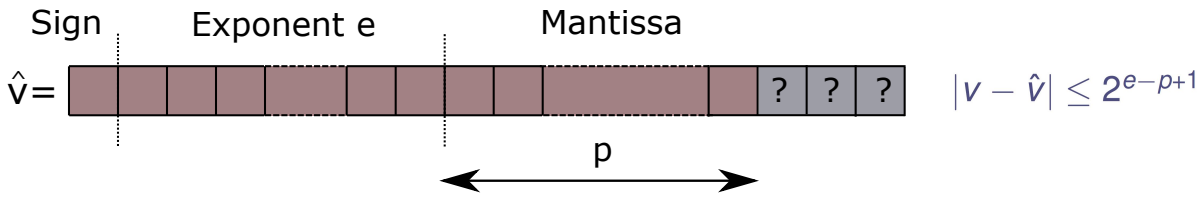
Formats : `half`, `float`, `double`, `quad`, `gmp`, `...`

Applications : Compilers, memory usage, bandwidth, etc...

Example

```
x_{t-1}:=[1.0,3.0]#16;
x_t:=[1.0,3.0]#16;
y_{t-1}:=0.0;
while(c) {
  u:=0.3 * y_{t-1};
  v:=0.7 * (x_t + x_{t-1});
  y_t:=u + v;
  y_{t-1}:=y_t;
};
require_accuracy(y_t,10);
```

Accuracy P:



$$x_{t-1}^{|16|} := [1.0,3.0]^{|16|};$$
$$x_t^{|16|} := [1.0,3.0]^{|16|};$$
$$y_{t-1}^{|52|} := 0.0^{|52|};$$
$$u^{|52|} := 0.3^{|52|} *^{|52|} y_{t-1}^{|52|};$$
$$v^{|15|} := 0.7^{|52|} *^{|15|} (x_t^{|16|} +^{|16|} x_{t-1}^{|16|});$$
$$y_t^{|15|} := u^{|52|} +^{|15|} v^{|15|};$$
$$y_{t-1}^{|15|} := y_t^{|15|};$$

$$x_{t-1}^{|9|} := [1.0,3.0]^{|9|};\ x_t^{|9|} := [1.0,3.0]^{|9|};$$
$$y_{t-1}^{|8|} := 0.0^{|8|};$$
$$u^{|10|} := 0.3^{|8|} *^{|10|} y_{t-1}^{|8|};$$
$$v^{|10|} := 0.7^{|11|} *^{|10|} (x_t^{|9|} +^{|10|} x_{t-1}^{|9|});$$
$$y_t^{|10|} := u^{|10|} +^{|10|} v^{|10|};$$
$$y_{t-1}^{|10|} := y_t^{|10|};$$
$$\texttt{require\_accuracy}(y_t,10);$$

Tools for Precision Tuning

❑ **Precimonious**

Delta-debugging, dynamic analysis, mixed precision

[C. Rubio-González, C. Nguyen, H. D. Nguyen, J. Demmel, W. Kahan, K. Sen, D. Bailey, C. Iancu, D. Hough, *Precimonious : tuning assistant for floating-point precision*, SC'13]

❑ **Twist**

SMT Solver, static analysis, mixed precision

[M. Martel, *Floating-Point Format Inference in Mixed-Precision*, NFM'17]

[D. Benkhalifa, M. Martel, *Precision Tuning by Static Analysis*, submitted'18]

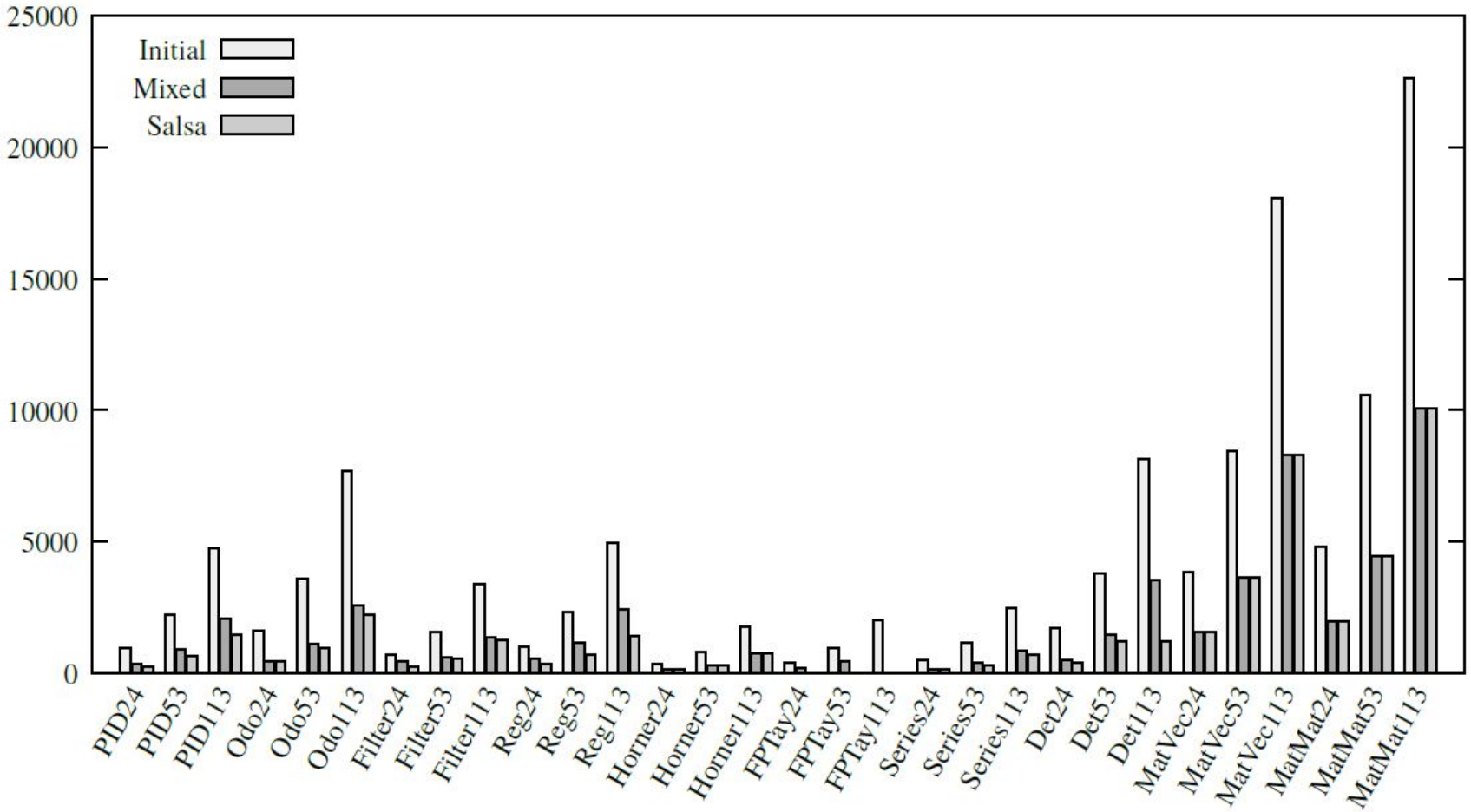
❑ **Daisy, FP-Taylor**

Static analysis, uniform precision

[E. Darulova, E. Horn, S. Sharma, *Sound mixed-precision optimization with rewriting*, ICCPS'18]

[W.-F. Chiang, M. Baranowski, I. Briggs, A. Solovyev, G. Gopalakrishnan, Z. Rakamaric, *Rigorous floating-point mixed-precision tuning*, POPL'17]

From [N. Damouche, M. Martel, Submitted'18]



Challenges in Reliable Computing (@ SE Level)

- Numerical accuracy determination
- Automatic accuracy optimization
- Precision tuning
- **Standard benchmarks & metrics to compare tools**
- Specific features in programming languages

Benchmarks & Metrics

- **How to compare tools ?**

- Input codes : expressions vs programs (which language) ?
- Input data : intervals, initial errors, etc. ?

- Measured error : worst error, mean error, absolute error, relative error, etc.
- Measurment : static vs dynamic analysis (if dynamic which inputs) ?

[N. Damouche, M. Martel, P. Panchekha, C.Qiu, A. Sanchez-Stern, Z. Tatlock, *Toward a Standard Benchmark Format and Suite for Floating-Point Analysis*, NSV'16]

FP-Bench

FPBench

Common standards for the floating-point research community

FPBench makes it easier to compare and combine tools from the floating-point research community.

Documentation
- Download
- Benchmarks
- Community
- Tools
- FPImp

Standards
- FPCore 1.0
- Metadata 1.0
- Measures 1.0

About
- Github
- Mailing list
- List archives

Current Status

FPBench was introduced at NSV'16. Since then, the benchmark suite has grown to 111 benchmarks, several implementations have appeared, and the standards have been deepened and improved.

FP-Bench

FPCore 1.0

A common format for floating-point computations

FPBench 1.0 standards

FPBench is a standard benchmark suite for the floating point community. The benchmark suite contains a common format for floating-point computation and metadata and a common set of accuracy measures:

1. The FPCore input format
2. Metadata for FPCore benchmarks
3. Standard measures of error

FPCore benchmark format

FPCore is the format used for FPBench benchmarks. It is a simple functional programming language with conditionals and simple loops. The syntax is an easy-to-parse S-expression syntax.

```
(FPCore (x)
 :name "NMSE example 3.1"
 :cite (hamming-1987)
 :pre (>= x 0)
 (- (sqrt (+ x 1)) (sqrt x)))
```

FP-Bench

Current Status

FPBench was introduced at NSV'16. Since then, the benchmark suite has grown to 111 benchmarks, several implementations have appeared, and the standards have been deepened and improved.

Benchmark sources	
Rosa	37
Herbie	28
Salsa	25
FPTaylor	21

Features used	
Arithmetic	111
Temporaries	57
Comparison	33
Loops	28
Exponents	16
Trigonometry	15
Conditionals	10

Domains	
Textbooks	28
Mathematics	24
Controls	10
Science	10

Challenges in Reliable Computing (@ SE Level)

- Numerical accuracy determination
- Automatic accuracy optimization
- Precision tuning
- Standard benchmarks & metrics to compare tools
- **Specific features in programming languages**

Numl

[M. Martel, Strongly typed Numerical Computations, submitted'18]

```
> 1.234 ;; (* precision of 53 bits by default *)
- : real{+,0,53} = 1.2340000000000000

> 1.234{4} ;; (* precision of 4 bits specified by user *)
- : real{+,0,4} = 1.2

> let f = fun x -> x + 1.0 ;;
val f : real{'a,'b,'c} -> real{<expr>,<expr>,<expr>} = <fun>

> verbose true ;;
- : unit = ()

> f ;;
- : real{'a,'b,'c} -> real{(SignPlus 'a 'b 1 0),((max 'b 0) +_ (sigma+
'a 1)), ((((max 'b 0) +_ (sigma+ 'a 1)) -_ (max ('b -_ 'c) -53))-_
(iota ('b -_ 'c) -53))} = <fun>
```

Numl

```
> f 1.234 ;;
- : real{+,1,53} = 2.2340000000000000

> f 1.234{4} ;;
- : real{+,1,5} = 2.2


> (1.0e15 + 1.0) - 1.0e15 ;;
- : real{+,50,54} = 1.0

> (1.0e16 + 1.0) - 1.0e16 ;;
Error: The computed value has no significant digit. Its ufp is 0 but
the ulp of the certified value is 1
```

Numl

```
> let rec g x = if x < 1.0 then x else g (x * 0.07) ;;
val g : real{+,0,53} -> real{+,0,53} = <fun>

> g 1.0 ;;
- : real{+,0,53} = 0.07000000000000

> g 2.0 ;;
Error: This expression has type real{+,1,53} but an expression was
expected of type real{+,0,53}

> let rec g x = if x <{10,15} 1. then x else g (x * 0.07) ;;
val g : real{*,10,15} -> real{*,10,15} = <fun>

> g 456.7 ;;
- : real{*,10,15} = 0.1

> g 4567.8 ;;
Error: This expression has type real{+,12,53} but an expression was
expected of type real{*,10,15}
```

Numl

```
> let rec h n = if (n=0) then 1.0 else 3.33 * (h (n -_ 1)) ;;
Error: This expression has type real{+,-1,-1} but an expression was
expected of type real{+,-3,-1}
```

```
> let rec taylor x{*,-1,25} xn i n = if (i > n) then 0.0{*,10,20}
                          else xn + (taylor x (x * xn) (i +_ 1) n) ;;
val taylor : real{*,-1,25} -> real{*,10,20} -> int -> int ->
real{*,10,20} = <fun>

> taylor 0.2 1.0 0 5;;
- : real{*,10,20} = 1.2499 +/- 0.0009765625
```

Numl

```
> let deriv f x h = ((f (x + h)) - (f x)) / h ;;
val deriv : (real{<expr>,<expr>,<expr>} -> real{'a,'b,'c}) ->
real{<expr>,<expr>,<expr>} -> real{'d,'e,'f} ->
real{<expr>,<expr>,<expr>} = <fun>

> let g x = (x*x) - (5.0*x) + 6.0 ;;
val g : real{'a,'b,'c} -> real{<expr>,<expr>,<expr>} = <fun>

> let gprime x = deriv g x 0.01 ;;
val gprime : real{<expr>,<expr>,<expr>} -> real{<expr>,<expr>,<expr>} =
<fun>

> let rec newton x xold f fprime = if ((x-xold)<0.01*,10,20) then x
                    else newton (x-((f x)/(fprime x))) x f fprime ;;
val newton : real{*,10,21} -> real{0,10,20} -> (real{*,10,21} ->
real{'a,'b,'c}) -> (real*,10,21 -> real{'d,'e,'f}) -> real{*,10,21} =
<fun>

> newton 9.0 0.0 g gprime ;;
- : real{*,10,21} = 5.771
```

Who Does What in our Group ?

Dorra Benkhalifa : Precision Tuning

Farah Benmouhoub : Accurate parallel codes (with P.-L. Garoche - ONERA)

Amine Benyelles : Accuracy determination (with Y. Seladji - U. Tlemcen)

Nasrine Damouche : Accuracy Optimization

Nacera Djehaf : Code synthesis (with M. Barboteu - U. Perpignan)

Matthieu Martel : Type systems

Alexis Werey : Formal proofs (with D. Delmas - Airbus)

Numalis : Accuracy determination (static & dynamic), accuracy optimization, precision tuning, C, Ada

Questions ?