Hessians in INTLAB, the Matlab toolbox for verified computations

Siegfried M. Rump, Hamburg

new book:

- F. Bornemann, D. Laurie, S. Wagon, J. Waldvogel:
  The SIAM 100-Digit Challenge,
  A Study in High-Accuracy Numerical Computing,
  SIAM, Philadelphia 2004.

In 5 out of 10 chapters solutions are presented using INTLAB (our interval Matlab toolbox).

... Various ingredients needed ...

[Matlab operator concept, Interval library, self-validating methods, gradients/Hessians, validated standard functions, input/output ...]

Matlab V5+ Operator concept


```
polynom p'

[list1 list2]

q\stack

R * intval(A)

y = f(hessianinit(x));
    x = x - y.hx \ y.dx'
```

directory `@hessian`

operators `plus, mtimes, transpose, display, ...`

Objectives for Interval library

- portability - from PC to supercomputer
  and parallel computer
- only 1 code - no special versions
- suitable for compilation and interpretation
- no special hardware
- fast

Computation with sets - Intervals (Sunaga '58)

$$A = [\underline{a}, \overline{a}] := \{x \in \mathbb{R} : \underline{a} \le x \le \overline{a}\} \in \mathbb{IR}$$

$$A \circ B := \bigcap \{C \in \mathbb{IR} : \ \forall a \in A \, \forall b \in B : \ a \circ b \in C\}$$

$$\text{for } \circ \in \{+, -, \cdot, /\}$$

well known rules

$$[\underline{a}, \overline{a}] + [\underline{b}, \overline{b}] \ = \ [\underline{a} + \underline{b}, \overline{a} + \overline{b}]$$

$$\ldots \qquad \ldots \qquad \ldots$$

Inclusion isotonicity

$$\forall a \in A \quad \forall b \in B : \quad a \circ b \in A \circ B$$

$$\begin{pmatrix} [\underline{x_1}, \overline{x_1}] \\ \vdots \\ [\underline{x_n}, \overline{x_n}] \end{pmatrix} \in (\mathbb{IR})^n \cong \mathbb{IR}^n, \text{ similarly } \mathbb{IR}^{n\times n}$$

$$(A \cdot B)_{ij} := \sum_k A_{ik} \cdot B_{kj} \text{ with interval } +, \cdot$$

Arithmetical issues (IEEE 754, 1984)

$\mathbb{F}$ Floating point numbers including $\pm\infty$

$\square$ rounding to nearest

∇ rounding downwards

$\triangle$ rounding upwards

$$\forall\, a, b \in \mathbb{F} \quad \forall \circ \in \{+, -, \cdot, /\}$$

$$\begin{aligned} \nabla(a \circ b) &= \max\{x \in \mathbb{F} : \; x \le a \circ b\} \\ \triangle(a \circ b) &= \min\{x \in \mathbb{F} : \; a \circ b \le x\} \end{aligned}$$

Back
Close

Intervals using floating point numbers

$$A = [\underline{a}, \overline{a}] := \{x \in \mathbb{R} : \underline{a} \le x \le \overline{a}\} \in \mathbb{IF} \quad \text{for } \underline{a}, \overline{a} \in \mathbb{F}$$

$$A \circ B := \bigcap \{C \in \mathbb{IF} : \ \forall\, a \in A \quad \forall\, b \in B : \ a \circ b \in C\}$$
$$\text{for } \circ \in \{+, -, \cdot, /\}$$

well known rules

$$[\underline{a}, \overline{a}] + [\underline{b}, \overline{b}] \ = \ [\bigtriangledown(\underline{a} + \underline{b}), \bigtriangleup(\overline{a} + \overline{b})]$$
$$\dots \qquad \dots \qquad \dots$$

Inclusion isotonicity

$$\forall\, a \in A \quad \forall\, b \in B : \qquad a \circ b \in A \circ B$$

Back
Close

$$\begin{pmatrix} [\underline{x_1}, \overline{x_1}] \\ \vdots \\ [\underline{x_n}, \overline{x_n}] \end{pmatrix} \in (\mathbb{IF})^n \cong \mathbb{IF}^n, \text{ similarly } \mathbb{IF}^{n\times n}$$

$$(A \cdot B)_{ij} := \sum_k A_{ik} \cdot B_{kj} \text{ with interval } +, \cdot$$

Valid bounds for the range of n-dimensional functions over a real box using only floating point operations

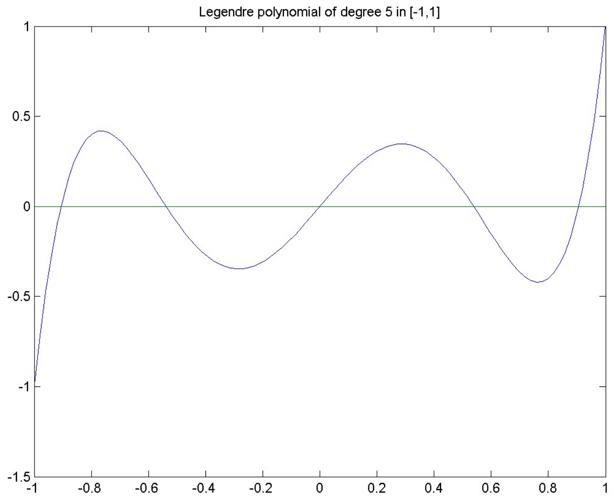
Bounds for the range of a function

Legendre polynomial

```
n = 5, P = legendre(n)
```

over $-1 \leq x \leq 1$

```
polynom P[x] =
   7.8750  x^5
  -8.7500  x^3
   1.8750  x
```




```
>> X = infsup(-1,1); P{X}
intval ans =
[  -18.5001,   18.5001]
```

Bounds for the range of a function

Griewank function

```
G = inline('(x^2+y^2)/4000+cos(x)*cos(y)/sqrt(2)+1')
```

over $-60 \le x, y \le 60$

2000 meshpoints in each coordinate:
upper bound for minimum min(G(x,y)) $\le$ 0.2957

```
>> X = infsup(-60,60); Y = X; G(X,Y)
intval ans =
[    0.2928,    3.5072]
```

A simple self-validating method

$f : \mathbb{R}^n \to R^n \in C^1$, $\tilde{x} \in \mathbb{R}^n$ with $f(\tilde{x}) \approx 0$

Compute bounds for $\hat{x} \approx \tilde{x}$ with $f(\hat{x}) = 0$.

$f(x) = 0 \quad \Leftrightarrow \quad g(x) = x$,
where $g(x) := x - R \cdot f(x)$, $det R \neq 0$.

$$X \in \mathbb{IR}^n,\ g(X) \subseteq X \ \overset{\text{Brouwer}}{\Rightarrow} \ \exists\, \hat{x} \in X : g(\hat{x}) = \hat{x}$$
$$\Rightarrow \quad f(\hat{x}) = 0$$

Check $g(X) \subseteq X$ and prove $\det R \neq 0$.

Naive approach:

$$g(X) \subseteq X - R \cdot f(X) \not\subseteq X$$

Back
Close

Bounds for the solution of nonlinear systems

Prove $g(X) = \{x - R \cdot f(x) : \; x \in X\} \subseteq X$

$$
\begin{aligned}
g(x) &= g(\tilde{x}) + M \cdot (x - \tilde{x}) \\
&\qquad \text{where } M_i = \frac{\partial g}{\partial x}(\zeta_i), \quad \zeta_i \in x \,\underline{\cup}\, \tilde{x} \\
&= \tilde{x} - R \cdot f(\tilde{x}) \;+\; \{I - R \cdot M\}(x - \tilde{x}) \\
&\subseteq \tilde{x} - R \cdot f(\tilde{x}) \;+\; \{I - R \cdot M\}(X - \tilde{x}) \\
&=: K_f(\tilde{x}, X) \qquad \text{Krawczyk operator}
\end{aligned}
$$

Theorem. $\tilde{x} \in X$ *and* $K_f(\tilde{x}, X) \subseteq$ *int*(X)
$\Rightarrow \quad \exists! \, \hat{x} \in X : \; f(\hat{x}) = 0.$

Computation of M by automatic differentiation

Fast matrix operations

Gradients

- Standard forward mode
- Sparse storage scheme

```
>> G = inline('(x^2+y^2)/4000+cos(x)*cos(y)/sqrt(2)+1')

>> X = gradientinit([2;3]); G(X(1),X(2))
gradient value ans.x =
   1.29456543963867
gradient derivative(s) ans.dx =
   0.63753584839404   0.04302600886466

>> X = gradientinit(midrad([2;3],1e-10)); G(X(1),X(2))
intval gradient value ans.x =
   1.294565440_____
intval gradient derivative(s) ans.dx =
   0.6375358484____   0.0430260089____
```

Hessian

- Standard forward mode
- Sparse storage scheme

Model Problem 61 (Conn, Gould, Lescrenier, Toint), Princeton:

```
N = length(x);   % model problem: N = 1000, initial x=ones(N,1);
I = 1:N-4;
y = sum( (-4*x(I)+3.0).^2 ) + sum( ( x(I).^2 + 2*x(I+1).^2 + ...
            3*x(I+2).^2 + 4*x(I+3).^2 + 5*x(N).^2 ).^2 );
```

size `(x.x)`	N elements
size `(x.dx)`	N^2 elements
size `(x.hx)`	N^3 elements

Hessian storage scheme

one possibility: symmetric

but not supported by Matlab

INTLAB: `a.hx` stored ⇒ `a.hx+a.hx`^{T} is Hessian

Example:

```
(a * b).hx = a.hx * b.x + a.dx * b.dx + a.x * b.hx
```

Matrix multiplication C=R*[A]

top-down approach (inner loop: single operations)

```
for i=1:n
    for j=1:n
       for k=1:n
          if R(i,k)>=0
             SetRoundDown
             Cinf(i,j)=Cinf(i,j)+R(i,k)*Ainf(i,k);
             SetRoundUp
             Csup(i,j)=Csup(i,j)+R(i,k)*Asup(i,k);
          else
             SetRoundDown
             Cinf(i,j)=Cinf(i,j)+R(i,k)*Asup(i,k);
             SetRoundUp
             Csup(i,j)=Csup(i,j)+R(i,k)*Ainf(i,k);
          end
       end
    end
end
```

Problem: inner loop not at all optimizable

Midpoint-radius arithmetic
Timing - Convex SPP 200

Compiled C-code [sec]

R*[A]	n=200	n=500	n=1000
top-down	2.48	71.4	570.4
new	0.07	1.7	12.3

[A]*[B]	n=200	n=500	n=1000
top-down	2.40	72.3	613.6
new	0.17	3.3	20.2

Matrix multiplication `C=R*[A]`

Matlab Timing (interpreted code): n=50, 120 Mhz Pentium

top-down	126	sec
new	0.06	sec

Details in S.M. Rump: Fast and parallel interval arithmetic. BIT, 39(3):539–560, 1999

Verified solution of linear systems: INTLAB timing

Solution of `Ax=b`, full matrix, randomly generated

		INTLAB	
n	built-in Matlab	point data	interval data
1000	2.8 s	20.2 s	22.3 s

Coconut example for n=1000, one Newton step, xs approximate minimum

```
>> tic,y.hx\y.dx';toc                    approximate solution
Elapsed time is 0.451000 seconds.

>> tic,verifylss(y.hx,y.dx');toc         verified inclusion
Elapsed time is 0.471000 seconds.
```

Rigorous standard functions: properties

- portability (pure Matlab code)
- $\lesssim$ 3 ulp accuracy over entire floating point range
- absolutely rigorous

Details in S.M. Rump: Rigorous and portable standard functions. BIT, 41(3):540–562, 2001

```
>> intvalinit('displayinfsup')
f=inline('x^2-2'), x=infsup(1,2) for i=1:4
  y=f(gradientinit(x)); x=intersect(x,x.mid-f(x.mid)/y.dx)
end

===> Default display of intervals by infimum/supremum
(e.g. [ 3.14 , 3.15 ])
f =
     Inline function:
     f(x) = x^2-2
intval x = [   1.00000000000000,   2.00000000000000]
intval x = [   1.37499999999999,   1.43750000000001]
intval x = [   1.41406249999999,   1.41441761363637]
intval x = [   1.41421355929452,   1.41421356594718]
intval x = [   1.41421356237309,   1.41421356237310]
```

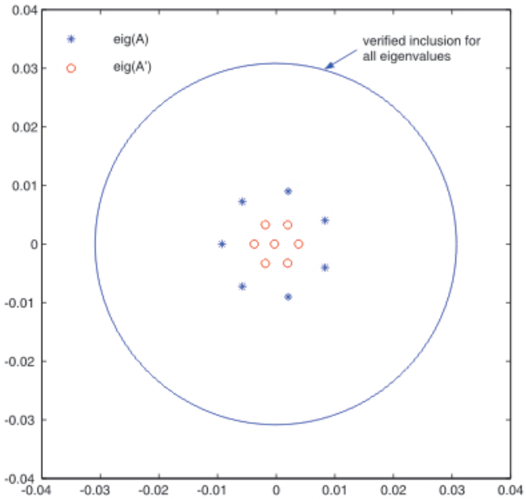
```
>> intvalinit('display_')
f=inline('x^2-2'), x=infsup(1,2) for i=1:4
  y=f(gradientinit(x)); x=intersect(x,x.mid-f(x.mid)/y.dx)
end

===> Default display of intervals with uncertainty (e.g. 3.14_)
f =
     Inline function:
     f(x) = x^2-2
intval x =
   2.______________
intval x =
   1.4_____________
intval x =
   1.414___________
intval x =
   1.41421356______
intval x =
   1.41421356237309
>>
```

Why self-validating methods:

Correct answer *or* error message

What are the eigenvalues of the following 7×7 matrix?

```
A =
 3.2704  4.2268 -2.9925  2.5902 -0.1306  1.3495  0.4708
 2.7041  0.9369 -0.7639  2.9434  1.1338 -0.5135 -1.2424
 2.6288  2.7926 -1.1863  2.4664 -0.4605  0.9760 -1.1777
-5.3920 -3.9846  3.7885 -5.2513 -2.7607  0.2610  0.0214
 3.3226  0.8233 -1.1560  3.3884  2.0575 -0.3869 -1.1942
-0.4403 -0.9826  0.4109 -0.3525  0.2232 -1.0126 -0.2296
-1.3145  1.8579 -0.4431 -1.6898 -1.9447  1.6597  1.1853
```


eig(A)
eig(A')
verified inclusion for
all eigenvalues
0.04
0.03
0.02
0.01
0
-0.01
-0.02
-0.03
-0.04
-0.04
-0.03
-0.02
-0.01
0
0.01
0.02
0.03
0.04

no warning!

Programming robustness

- Construct problem with integer entries
  and known solution, e.g. π
- A programming error may produce

$$[3.141592653589792, 3.141592653589793]$$

Floating point approximations of superb quality.

The bounds coincide to 16 decimal places
but are WRONG!

INTLAB Highlights

- all basic real and complex interval operations
- fast and rigorous real and complex standard functions
- gradient toolbox
- hessian toolbox
- slope toolbox
- polynomial toolbox
- multiple precision toolbox (slow)

25 KLOC (pure Matlab), $>$ 600 routines

some 3500 users in more than 40 countries

free from `http://www.ti3.tu-harburg.de`