Software for Verified Computing & Modeling with Uncertain Data

W. Krämer, Univ. Wuppertal

E. Popova, BAS, Sofia

Contents:

- Why verified computing.
- Interval arithmetic as a tool.
- Interval support – Historical remarks.

- Sun's compiler support.

- Intlab for Matlab.

- C-XSC, CToolbox for verified computing.

Why Verified Computing?

- **Computing speed produces unprecedented opportunities & risks.**
- **The existing floating-point paradigm can**

  neither exploit all the opportunities, nor avoid the risks.

  - Fl-p numbers are logically disconnected from the real world;
  - a single fl-p number contains no accuracy information.

Unvalidated computer results are used to make critical decisions.

The greater the dependence on comp.result;

The more important their use;

⇒ **the greater the risk!**

For disasters attributable to bad numerics see

http://www.ima.umn.edu/~arnold/disasters/sleipner.html

There must be a tight logical connection btw computing and reality!

How do we obtain verified results?

- **1958, T. Sunaga** – rigorous error estimation alternative to J.v.Neuman & H. Goldsteine's
- **1966, R. Moore** -- commence systematic investigations

Interval Analysis as a Tool

$$[x] = [\underline{x},\ \overline{x}] \ := \ \{\tilde{x} \in \mathbb{R} \mid \underline{x}, \overline{x} \in \mathbb{R};\ \underline{x} \leqq \tilde{x} \leqq \overline{x}\}$$

$[\underline{x},\ \overline{x}]$ brings continuum on the computer

$\mathrm{width}([x])$ represents accuracy of the approximation

Properties:

$$[a] \subseteq [b] \wedge [c] \subseteq [d] \implies [a] \circ [c] \subseteq [b] \circ [d]$$

$$a \in [a],\ b \in [b] \implies a \circ b \in [a] \circ [b]$$

Self-validating algorithms establish:

- **existence**
- **uniqueness**
- **inclusion**

of the problem solution within the computed bounds.

IA is an extension of fl-p arithmetic, not a replacement for it

The extended tool delivers a guaranteed answer faster than

the restricted tool of fl-p arithmetic delivers an "approximation“:

- **numerical integration (because of automatic step size control);**
- **global optimization (i-ls bring the continuum on the computer);**
- **one int evaluation of a function over an interval may suffice to prove that the function definitely has no zero in this interval, while 1000 fl-p evaluations of the function in the i-l could not provide a safe answer;**
- **systems of ODE and integral eqs deliver not just unproven numbers but close bounds and prove existence and uniqueness of the solution within the computed bounds.**

  The bounds include both discretization and rounding errors.

  This can save a lot of computing time by avoiding experimental reruns.

Validated computing uses controlled rounding of computer arithmetic

to guarantee that hypotheses of suitable mathematical theorems

are (or are not) satisfied.

Mathematical rigor in

- **the computer arithmetic,**
- **in algorithm design, and**
- **in program execution**

allow us to guarantee that the stated problem has (or does not have) a solution in an enclosing interval we compute.

If the enclosure is narrow, we are certain that we know the answer reliably and accurately.

If wide, we have a clear warning that our uncertainty is large, and a closer study is demanded.

Interval Computations on the Computers

$$\Diamond[x] := [\bigtriangledown \underline{x}, \bigtriangleup \overline{x}] \qquad \diamondsuit := \Diamond([x] \circ [y]), \qquad \circ \in \{+, -, \times, /\}$$

IEEE-Std. 754, 1984

$$\bigtriangledown \tilde{x} := \sup\{x_m \in \mathbb{F} \mid x_m \leqq \tilde{x}\} \qquad \bigtriangleup \tilde{x} := \inf\{x_m \in \mathbb{F} \mid x_m \geqq \tilde{x}\}$$

While floating-point arithmetic is provided by fast hardware,

interval arithmetic has to be simulated by software.

XSC – Languages

- 1967-69, Uni.Karsruhe, **ALGOL 60** extension;
- 1978, with Nixdorf, **PASCAL-SC**;
- 1983-89, with IBM, **ACRITH, ACRITH-XSC**;
- 1988, W. Walter, **FORTRAN-SC, FORTRAN-XSC**;
- 1990, U. Basel, **MODULA-SC**;
- 1985-91, with NAG, **ADA**;
- 1998, with ETHZ, **Oberon-XSC**;

- 1990, **PASCAL-XSC**, new runtime system;
- 1992, **C-XSC**;

language eXtensions for Scientific Computation

provide all features indispensable for modern numerical software development

- Operator concept (user-defined operators)
- Overloading concept
- Module concept
- Dynamic arrays
- Controlled rounding
- Predefined arithmetic data types real, (extended real), complex, interval, complex interval, and corresponding vector and matrix types
- Predefined arithmetic operators of highest accuracy for the arithmetic data types
- Predefined elementary functions of highest accuracy for the arithmetic data types
- Data type dotprecision for the exact representation of dot products

Library of mathematical problem-solving routines with automatic result verification and high accuracy.

IA libraries:

1978, S. Markov et all., HIFICOMP;

1992, B. Kearfott, INTLIB for FORTRAN 77;

1994, O.Knueppel, PROFIL (BIAS) in C++;

Hardware designs:

to overcome speed limitations of software tools

1990-- , Uni. Karlsruhe, Dotproduct coprocessor

1995--, Uni. Leicester, Variale Precision IA coprocessor

Sun Microsystems

Commersial compiler support of IA

2000, **Forte Fortran 95**

http://wwws.sun.com/software/sundev/previous/fortran/interval/

language extension supporting intrinsic INTERVAL data types

2001, **Forte C++**

http://wwws.sun.com/software/sundev/previous/cplusplus/interval/

C++ interface to the C++ IA library

Platforms ranging from desktops to clustered enterprise-class servers

Sun's goal is:

to stimulate development of commercial interval solver libraries & applications by providing program developers with

- **Quality interval code;**
- **Narrow-width interval results;**
- **Rapidly executing interval code;**
- **Easy-to-use software development environment.**

The Forte Fortran 95 compiler

contains the following interval features and extensions:

- Full support for extended interval data types

- Valid results for any possible operator-operand combination

interval arithmetic operations and intrinsic mathematical functions form a closed system.

- Interval versions of all Fortran 95 intrinsic operators and functions that accept real arguments

- A number of interval-specific intrinsic operators and functions, including:
  - interval-specific relational operators
  - set-theoretic intrinsic operators
  - mixed-mode expression evaluation
  - full input/output support

With the **Forte Fortran 95** compiler, it is a simple matter to write interval programs to compute rigorous bounds on the value of arithmetic expressions:

- Declare variables to be type INTERVAL.

- Write normal Fortran code using the intrinsic INTERVAL functions, operators, relational operators, and format edit descriptors.

- Compile code using the -xia command-line option.

Interval Arithmetic support for C++ provides

a C++ header file and

library that implements three interval classes,

one each for float, double, and long double.

The interval classes include:

- Interval arithmetic operations and mathematical functions that form a closed mathematical system.

  This means that valid results are produced for any possible operator-operand combination, including division by zero and other indeterminate forms involving zero and infinities.

- Three types of interval relational functions:

  Certainly, Possibly, Set

- Interval-specific functions, such as intersect and interval_hull

- Interval-specific functions, such as inf, sup, and wid

- Interval input/output, including single-number input/output

Containment Mode

Extended interval arithmetic (containment computations):
Exception free computations in a closed interval system with

- infinite intervals
- only partially defined functions
- functions with singularities

Containment set:

$$f^*(X) := \{ \quad f(x) \mid x \in X \cap D_f\}$$
$$\bigcup \quad \{ \lim_{D_f \ni x_k \to x^*} f(x_k) \mid x^* \in X\} \subseteq I\!R^*$$

$\sqrt{[-1,4]} = [0,2],\ \log[-1,e] = [-\infty,1],\ 1/[0,0] = \{-\infty,\infty\}$

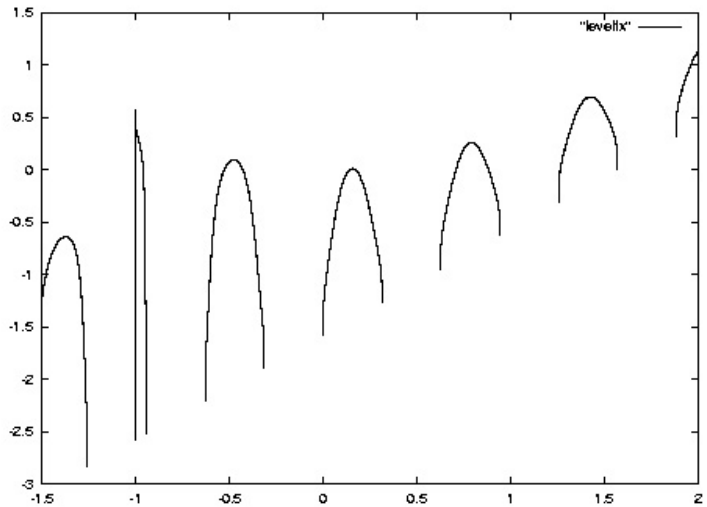
$$f(x) = x + \arctan(\log(\sin(10x)/(x+1)))$$

All zeros of f on the **complete** real line using bisection

Note:

- natural domain of definition D_f is not the complete real line
- $x = -1$ leads to division by zero

Due to containment computations **no** exceptions are thrown!



Numerical Results

```
All zeros of x + atan( log( sin(10*x)/(x+1) ) )
Search range: [ ENTIRE ]
eps: 1e-7
There may be zeros in the interval(s):
  1  [-1.000000059604645,  -0.9999999999999999]
  2  [-0.9594483375549315, -0.9594482779502868]
  3  [-0.5274564027786254, -0.5274562835693358]
  4  [-0.4268767237663268, -0.4268766045570373]
  5  [0.1427576541900634,    0.1427577733993531]
  6  [0.1746257543563842,   0.1746259331703186]
  7  [0.7093482613563536,   0.7093483209609984]
  8  [0.8833859562873839,   0.8833860158920287]
  9  [1.266079187393188,     1.266079246997833]
  10 [1.570796310901642,     1.570796370506286]  lim f(x) = 0
                                                 x->pi/2
```

Interactive Programming Environments

- **Maple**
  * intpack, 1993 (A. Connel, R. Corless)
  * intpackX, 1999 (I. Geulig, W. Krämer)
  * intpackX v1.0 for Maple 6, 2001 (M. Grimmer)
    integrates intpack with intpackX
    http://www.math.uni-wuppertal.de/wrswt/software

- ***Mathematica***
  * IA packages, 1990-1993 (J. Keiper)
  * since ver. 2.2 the object Interval smoothly integrated into the kernel
  * several additional packages developed in Sofia

- **MATLAB**
  * **INTLAB**, 1998 (S. Rump) -> best supported!

INTLAB for MATLAB

http://www.ti3.tu-harburg.de/rump/intlab/

toolbox for self-validating algorithms comprising:

IA for real and complex data including vectors and matrices

IA for real and complex SPARSE matrices

rigorous real interval & complex interval standard functions

rigorous input/output

multiple precision IA with error bounds

automatic differentiation (forward mode, vectorized computations)

automatic slopes

toolbox univariate and multivariate (interval) polynomials

problem solving routines for systems of linear & nonlinear equations
eigenvalue problems, and more.

INTLAB features:

- everything is written in Matlab code -> portability

  INTLAB tested under Windows with Matlab 5.3, 6.0 and 6.5.

- required IEEE 754 arithmetic and switching the rounding mode.

  rounding is already integral part of Matlab 5.3, Windows
  routines for switching the rounding mode are available for other platforms/OS

- infimum-supremum & midpoint-radius representations of intervals

- INTLAB extensively uses BLAS routines => very fast matrix operations

- Midpoint-radius implementation takes full advantage of the speed of vector & parallel architectures.

- INTLAB code is elegant, easy to read and to maintain;
  powerful interactive tool to implement prototypes of verification algorithms.

Download INTLAB 4.1.1 source code for Unix and for Windows
http://www.ti3.tu-harburg.de/rump/intlab/

J. More: A Collection of Nonlinear Model Problems.
In: Computational Solution of Nonlinear Systems of Equations (Eds.: E.L. Allgower, K. Georg), Lectures in Applied Mathematics, Vol. 26, AMS (1990).

Fletcher describes a "distillation column test problem",
and gives data for three processes.

For the third (the methanol-8 problem with 31 unknowns) he writes:
"I still do not know if there exists a solution to this problem."
The best he found was an approximation with 1e-2 residual norm.

All three problem where solved with verified bounds in
G. Alefeld, A. Gienger and F. Potra: Efficient Numerical Validation of Solutions of Nonlinear Systems, SIAM J. Num. Anal. 31, 252-260 (1994).

The call

xs = fletcher('init1') 'init2' or 'init3' generates the approximation

All three problem where solved with verified bounds.

X = verifynlss('fletcher', xs)

The number of the problem is choosen in fletcher.m according the dimension of xs.

problem	unknowns	rel. error median	rel. error maximum	time
1 hydrocarbon-6	29	9.2e-14	3.6e-13	2.5 sec
2 hydrocarbon-20	99	2.8e-12	8.4e-11	19 sec
3 methanol-8	31	3.8e-14	2.7e-13	3.0 sec

The computing time is measured on a 750 MHz Laptop.

C++ Class Library C-XSC

a tool for development of numerical algorithms delivering highly accurate & verified results

- 1990/91 Univ. Karlsruhe; 1992 Language Reference, Springer-Verlag

- 1994, CToolbox for Verified Computing in C-XSC

- June 1997, XSC General Public License

- 1999/2000 redesign to conform to ISO/IEC C++ standard 14882-1998

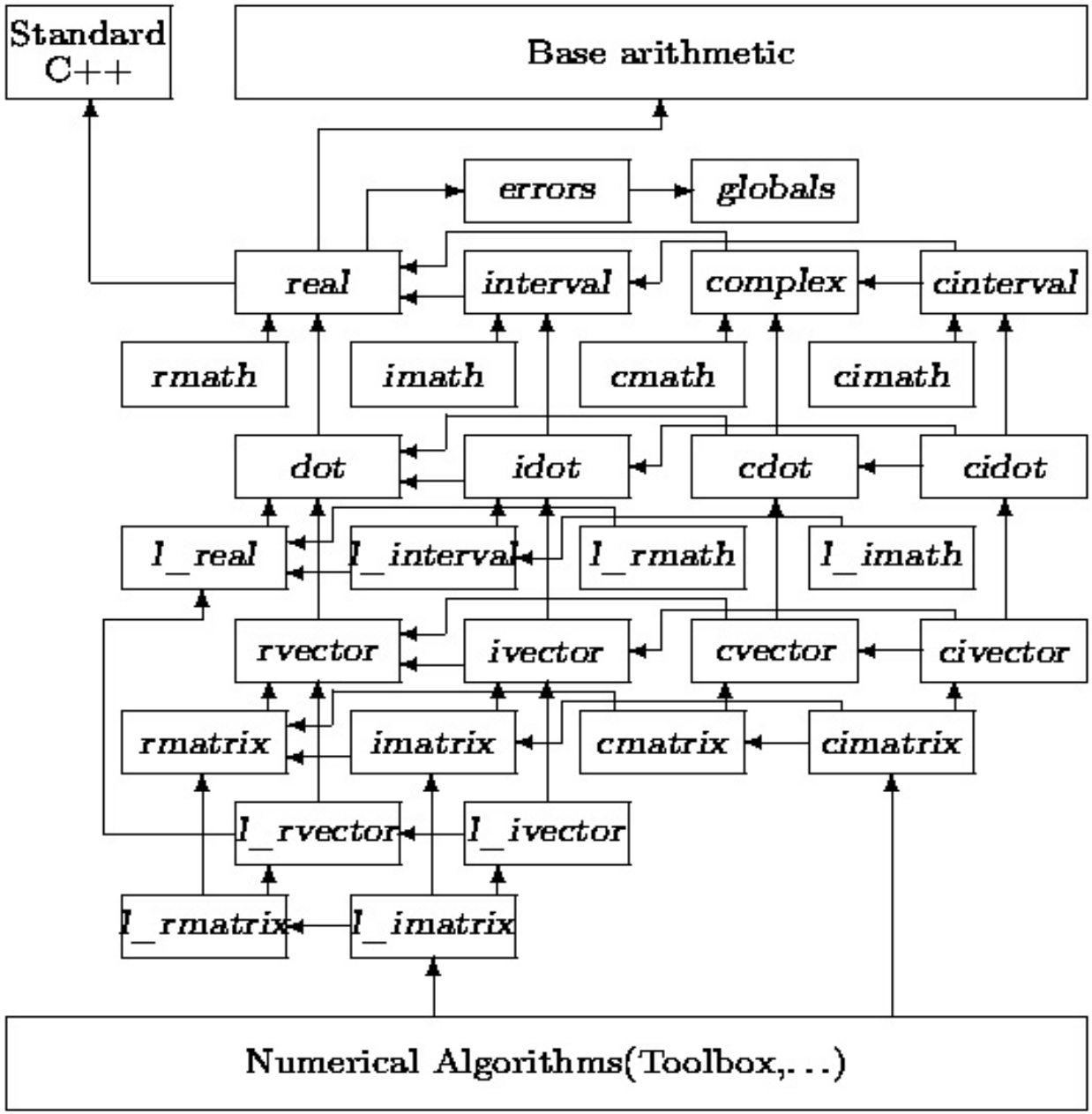
Tested on:
PC with LINUX and GNU C++ compiler gcc 2.95.2/gcc 2.95.3/gcc 3.0.1/gcc 3.2
SUN Solaris workstation with GNU C++ compiler gcc 2.95.2/gcc 2.95.3/gcc 3.2
DEC alpha with LINUX with GNU C++ compiler gcc 2.95.2

supported in cooperation Karlsruhe/Wuppertal, GPL

http://www.math.uni-wuppertal.de/~xsc

The most important features of C-XSC are:

- Real, complex, interval, and complex interval arithmetic
  with mathematically defined properties
- Dynamic vectors and matrices
- Subarrays of vectors and matrices
- Dotprecision data types
- Predefined arithmetic operators with highest accuracy
- Standard functions of high accuracy
- Dynamic multiple-precision arithmetic and standard functions
- Rounding control for I/O data
- Error handling
- Library of problem solving routines


Standard C++
Base arithmetic
errors
globals
real
interval
complex
cinterval
rmath
imath
cmath
cimath
dot
idot
cdot
cidot
l_real
l_interval
l_rmath
l_imath
rvector
ivector
cvector
civector
rmatrix
imatrix
cmatrix
cimatrix
l_rvector
l_ivector
l_rmatrix
l_imatrix
Numerical Algorithms(Toolbox,...)

left operand \ right operand	integer real complex	interval cinterval	rvector cvector	ivector civector	rmatrix cmatrix	imatrix cimatrix
monadic	–	–	–	–	–	–
integer real complex	$+, -, *, /$ |	$+, -, *, /$ |	*	*	*	*
interval cinterval	$+, -, *, /$ |	$+, -, *, /$ |, &	*	*	*	*
rvector cvector	$*, /$	$*, /$	$+, -, *^{\dagger}$ |	$+, -, *^{\dagger}$ |		
ivector civector	$*, /$	$*, /$	$+, -, *^{\dagger}$ |	$+, -, *^{\dagger}$ |, &		
rmatrix cmatrix	$*, /$	$*, /$	$*^{\dagger}$	$*^{\dagger}$	$+, -, *^{\dagger}$ |	$+, -, *^{\dagger}$ |
imatrix cimatrix	$*, /$	$*, /$	$*^{\dagger}$	$*^{\dagger}$	$+, -, *^{\dagger}$ |	$+, -, *^{\dagger}$ |, &

|: Interval Hull &: Intersection †: Dot Product with Maximum Accuracy

CToolbox for Verified Computing in C-XSC

collection of routines for standard problems of numerical analysis producing guaranteed results of high accuracy

One-Dimensional Problems:

- Evaluation of Polynomials (Module rpoly, Module rpeval)
- Automatic Differentiation (Module ddf_ari)
- Nonlinear Equations in One Variable (Module xi_ari, Module nlfzero)
- Global Optimization (Module lst1_ari, Module gop1)
- Evaluation of Arithmetic Expressions (Module expreval)
- Zeros of Complex Polynomials (Module cpoly, Module cipoly, Module cpzero)

CToolbox for Verified Computing in C-XSC

Multi-Dimensional Problems:

- Linear Systems of Equations (Module matinv, Module linsys)
- Linear Optimization
  (Module set_ari, Module lop_ari, Module rev_simp, Module lop)
- Automatic Differentiation for Gradients, Jacobians, and Hessians
  (Module hess_ari, Module grad_ari)
- Nonlinear Systems of Equations (Module nlinsys)
- Global Optimization (Module lst_ari, Module gop)

Further packages based on C-XSC

- Complex interval standard functions library (CoStLy)
- Automatic Computation of Estimates for Taylor Coefficients of Analytic Functions (ACETAF)
- One- and multidimensional (interval) slope arithmetic
- Classes for Verified Integration over Singularities (CLAVIS)
- Parametric interval linear systems
- Initial value problems in ordinary differential equations

and more ...

Verified Graphical Results

Level curves

Given: $f : \mathbb{R}^2 \longrightarrow \mathbb{R}$, $c \in \mathbb{R}$, Box $X \subset \mathbb{R}^2$

To do: plot all points $x \in X$ with $f(x) = c$
or with $f(x)$ in some given interval

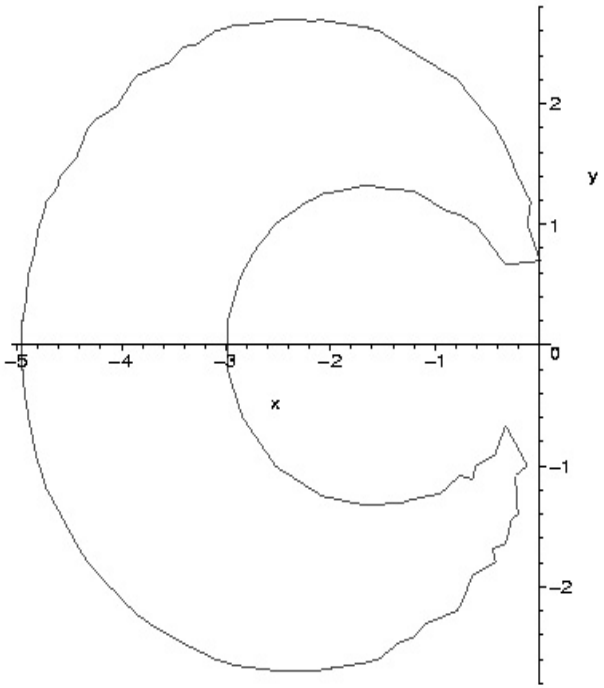
Example (Pascalsche Schnecke):

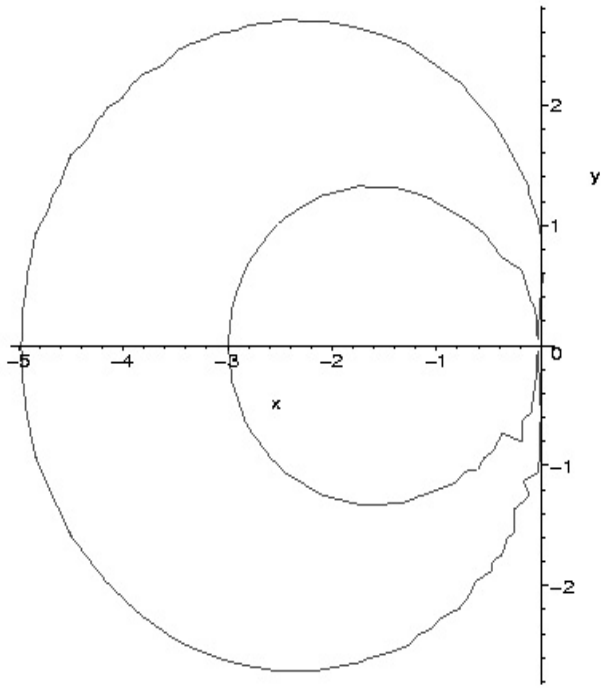
$$f(x, y) = (x^2 + y^2 + 4x)^2 - x^2 - y^2 \stackrel{!}{=} 0$$

using Maple and using C-XSC, the STL and gnuplot

What is the correct shape of $f(x,y) = (x^2+y^2+4x)^2 - x^2 - y^2 \stackrel{!}{=} 0$

Using Maple's `implicitplot`-command we get

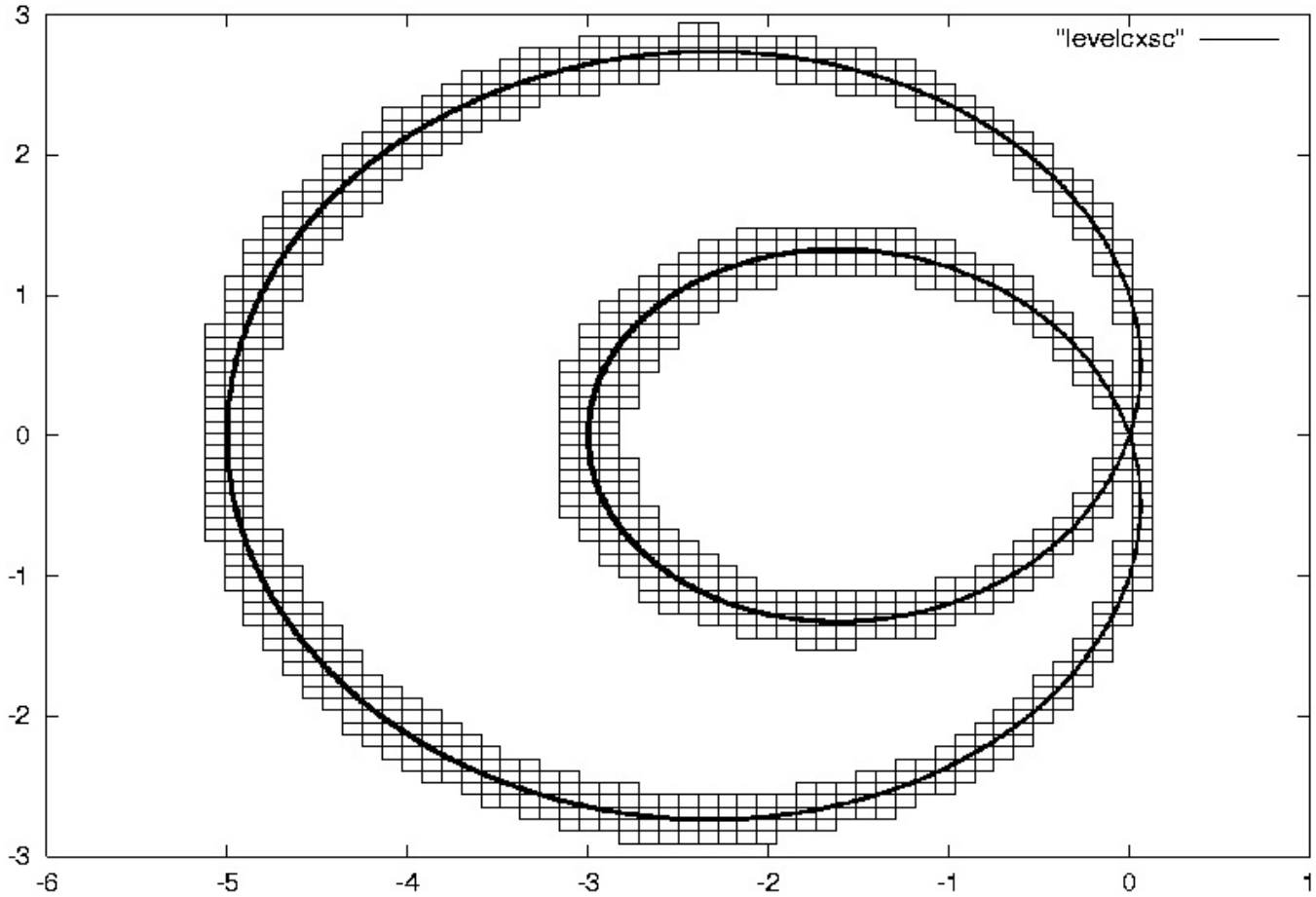




Symmetry with respect to y?

Coarse and fine coverings of the level curve (verified)

```
xRange: [ -6.000000,  1.000000]
yRange: [ -6.000000,  5.000000]
level : 0
Epsilon (fine): 0.01
```



Whenever you have to:

-**compute the solution of a numerical problem rigorously verified to be correct,**

- **model/design with uncertain data**

do not reinvent the wheel,
but have a look at software already available!

http://www.cs.utep.edu/int-comp/

ask help writing to

reliable_computing@interval.louisiana.edu