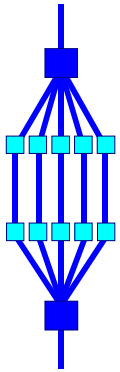
Intervals @ Sun
Connect Computing to the World



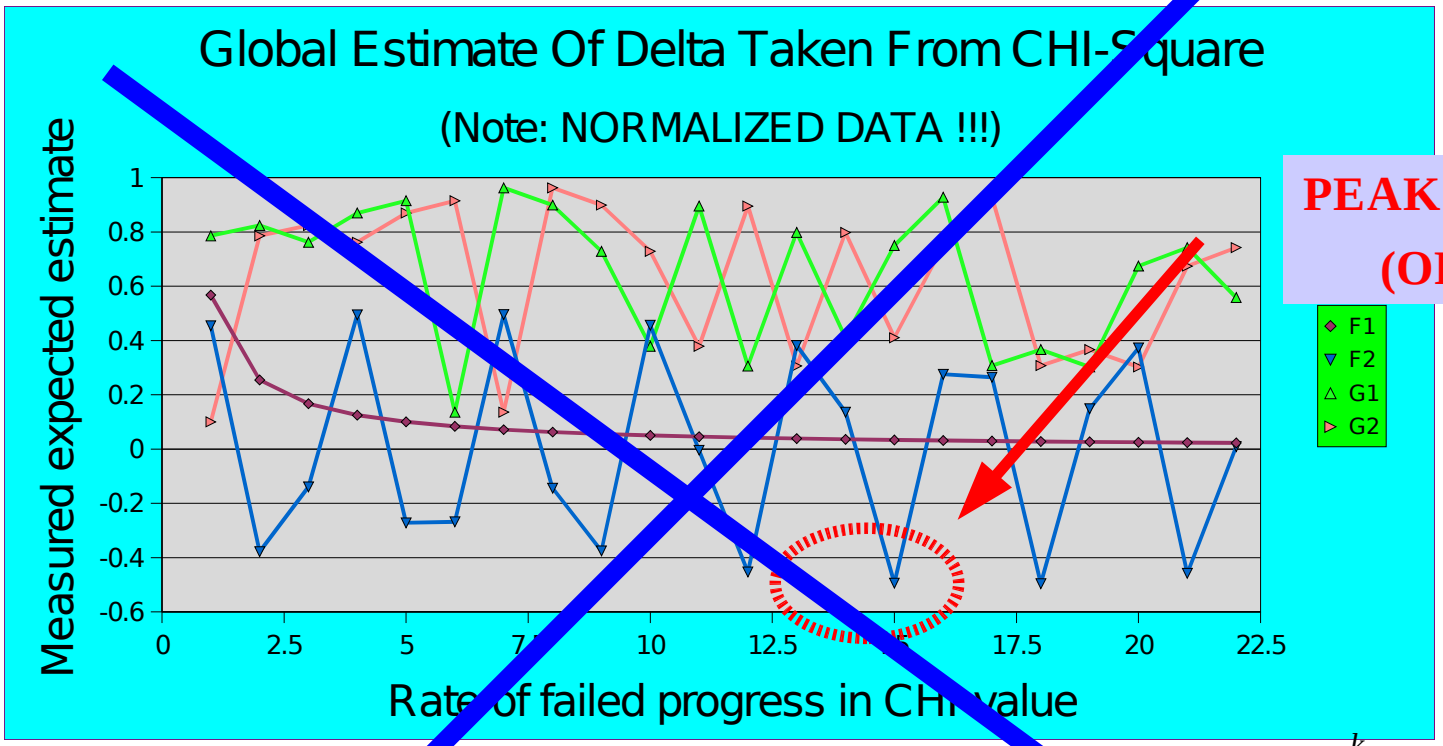
Interval Arithmetic Support In The Fortran Compiler

P.Sambath Narayanan

Sun Microsystems

Computing School
January 11, 2005
Chennai

This Is Not A Scientific Talk



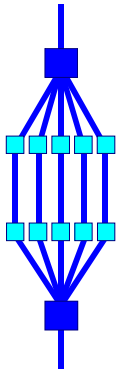
PEAK EXPECTED ?!
(OF COURSE)

?

$$G2(X,Y)=\oint\!\!\iint Z(h,j)=\sum_{i=0}^{k} D(i,h)*\prod F(j)$$

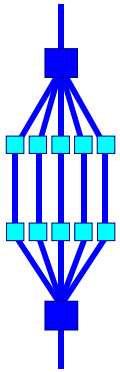
```
nextTLP[0]->previous = all_list_t_bottom[0];
     all_list_t_bottom[0] =  nextTLP[0];

        tree_top[0]->left = tree_top[0]->right = NULL;      /*initialize search  tree*/
        for (l=0;l<NO_IDS; l++) tree_top[0]->conf[l] = 0;
NO_FREE[0] = BLOCK_LENGTH-1;
where_to_append[0] = tree_top[0] ;
where_to_append[0] ++;
no_to_do[0] = 1;
no_done[0] = 0;
new_gstate_found[0] = 0;

        for (i=1; i<TWO_TO_THE_H; i++)
{
```

Outline



- *Introduction*
- *Interval-Specific Operators and Intrinsic Functions*
- *Quality Of Implementation Opportunities*
- *Conclusions*

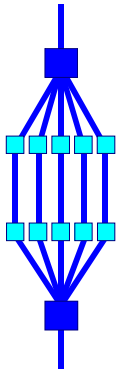
Good News For Intervals !



- *Sun has been awarded the DARPA HPCS contract*
  - *HPCS = High Productivity Computing Systems*
  - *HPCS Goal: Build a Peta-scale system*
  - *This is phase 2 of a 3-tier project*
  - *Three vendors selected (IBM, Cray and Sun)*
- *One key element in Sun's proposal is the usage of*

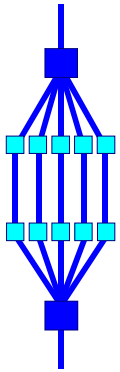
[Inter , vals]

Introduction



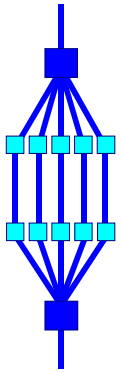
- *This presentation is based on discussions with Bill Walster (Sun Microsystems)*
  - *Without his relentless enthusiasm there would not have been an interval compiler from Sun*
  - *An extensive interview with Bill can be found at* ***http://www.sun.com/presents/minds/2004-0527***
- *The Sun Fortran and C++ compilers support intervals since 2000 (available on SPARC based systems)*
  - *Fortran: native data type*
  - *C++: class library*
- *In this talk we would like to give an overview of the interval features supported in the Sun Fortran compiler*

Pointers To More Information/1



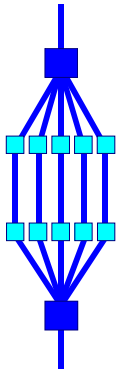
- *Download the compilers (Sun Studio 8)*
  - *URL is* ***http://developers.sun.com/prodtech/cc***
  - *Can use a free "try and buy" license and/or*
  - *Interval support (Fortran and C++) included*
    - ✔ ***Use -xia option to activate***

Pointers To More Information/2



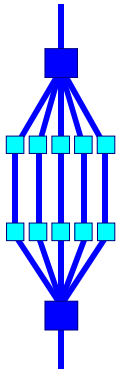
- *Documentation*
  - *Fortran Interval Arithmetic Programming Reference*
    - ✔ *http://docs.sun.com/db/doc/817-5076*
  - *C++ Interval Arithmetic Programming Reference*
    - ✔ *http://docs.sun.com/db/doc/817-5077*
- *More information plus code examples can be downloaded from* **http://developers.sun.com**
  - *Go to the "Compiler Collection" portal*
- *Another useful web site (on numerical computations):*
  - *http://developers.sun.com/prodtech/cc/numerics_index.html*

Intervals @ Sun
Connect Computing to the World



Basic Interval Arithmetic

Basic Arithmetic Operations



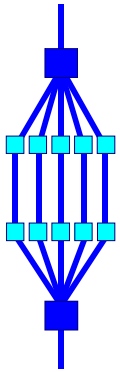
Assume that [a,b] and [c,d] are intervals

For a basic operator "op" in {+,-,*,/} we can then define:

$$[a,b] \text{ "op" } [c,d] \supseteq \{x \text{ "op" } y \mid x \in [a,b] \text{ and } y \in [c,d]\}$$

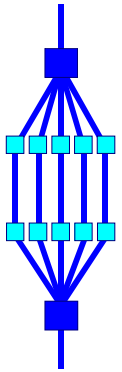
Formulas for basic operations:

```
[a,b] + [c,d] = [a+c,b+d]
[a,b] - [c,d] = [a-d,b-c]
[a,b] * [c,d] = [min(a*c,a*d,b*c,b*d,
                     max(a*c,a*d,b*c,b*d)]
[a,b] / [c,d] = [min(a/c,a/d,b/c,b/d),
                     max(a/c,a/d,b/c,b/d)]
                          (if 0 is not included in [c,d])
```



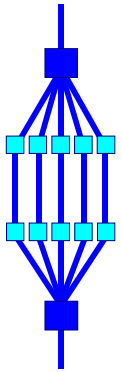
Interval-Arithmetic Applications

Types of problems



- *Unconstrained Global nonlinear minimization*
- *Constrained Global nonlinear minimization*
- *Guaranteed equation solving*
- *Imaging of a set by nonlinear function*

Types of problems



- *Probably the most current source for applications is the Proceedings of the NSF Workshop on Reliable Engineering Computing, held in November, 2004 a the Georgia Institute of Technology. See:*

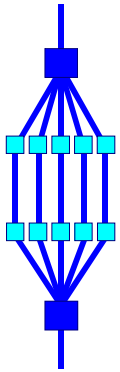
<http://www.gtsav.gatech.edu/rec/recworkshop/>

- *Papers on mechanical and chemical engineering.There has been a lot of excellent work on integrating ODEs.*

<http://www.bt.pa.msu.edu/cgi bin/displaytest.pl?name=VIRC03>

- *The use of Taylor multinomials and arithmetic on them has been used to solve particle-beam accelerator design problems. See:*

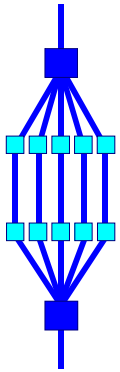
<http://www.bt.pa.msu.edu/index_files/cosy.htm>



F95 Interval Support Goal

F95 interval support Goal

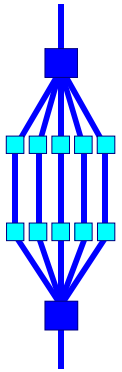
Intervals @ Sun
Connect Computing to the World



- *Quality interval code*
- *Narrow-width interval results*
- *Rapidly executing interval code*
- *An easy to use interval software development environment that includes intervalspecific language support and compiler features*

This will stimulate interval solver libraries and applications

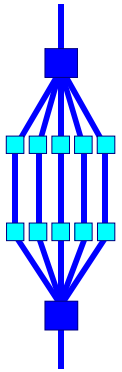
Narrow-Width Interval Results



- *Minimize the width of computed intervals while always satisfying the containment constraint.*
- *If an interval's width is as narrow as possible, it is said to be sharp.*
- *Width of intervals produced by the f95 compiler:*
  - *Individual intervals are sharp approximations of constants.*
  - *Individual interval arithmetic operators produce sharp results.*
  - *Intrinsic mathematical functions usually produce sharp results.*

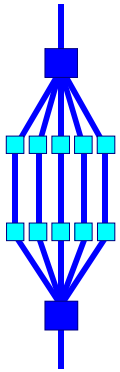
Easy to Use Development Environment

Intervals @ Sun
Connect Computing to the World



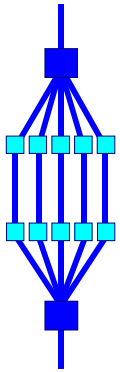
- *INTERVAL data types*
- *INTERVAL arithmetic operations and intrinsic mathematical functions form a closed mathematical system.*
- *Intrinsic INTERVAL-specific operators, such as .IX. (intersection) and .IH. (interval hull)*
- *INTERVAL-specific functions, such as INF, SUP, and WID*
- *Three classes of interval relational operators:*
  - *Certainly*
  - *Possibly*
  - *Set*

Command Line Options



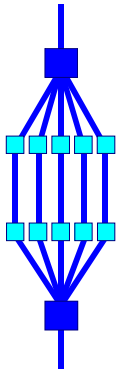
- *Compiler support for widest-need interval expression processing is invoked by including :*
  - *-xia or -xia=widestneed*
- *Compiler support for strict interval expression processing is invoked by including :*
  - *-xia=strict*

Intervals @ Sun
Connect Computing to the World



Interval-Specific Operators and Intrinsic Functions

Basic Arithmetic Operations



Assume that [a,b] and [c,d] are intervals

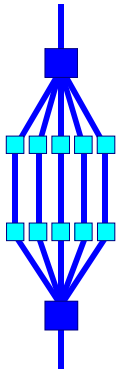
For a basic operator "op" in {+,-,*,/} we can then define:

$$[a,b] \text{ "op" } [c,d] \supseteq \{x \text{ "op" } y \mid x \in [a,b] \text{ and } y \in [c,d]\}$$

Formulas for basic operations:

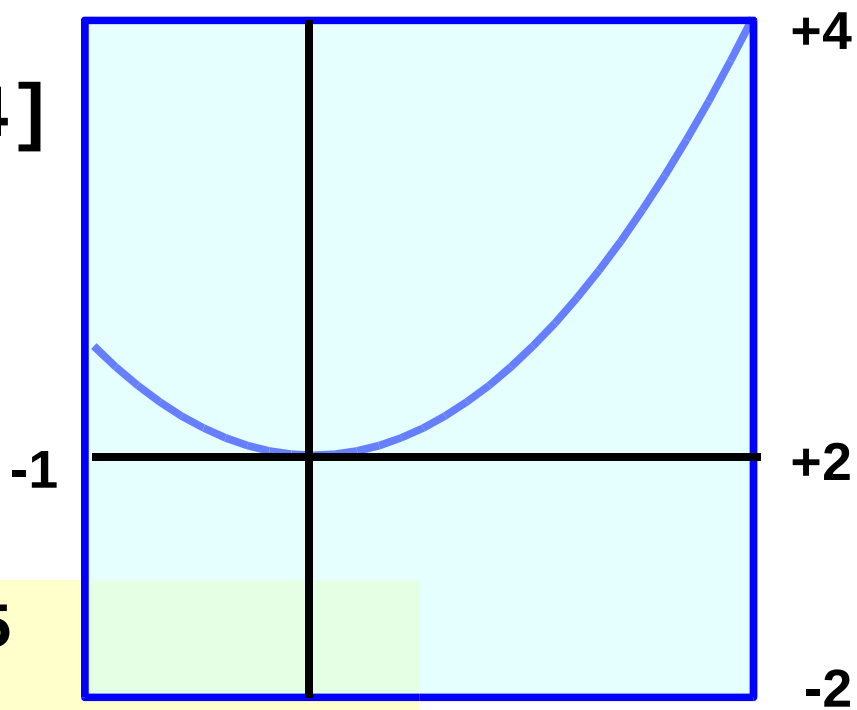
```
[a,b] + [c,d] = [a+c,b+d]
[a,b] - [c,d] = [a-d,b-c]
[a,b] * [c,d] = [min(a*c,a*d,b*c,b*d,
                     max(a*c,a*d,b*c,b*d)]
[a,b] / [c,d] = [min(a/c,a/d,b/c,b/d),
                     max(a/c,a/d,b/c,b/d)]
                          (if 0 is not included in [c,d])
```

Integer Powers



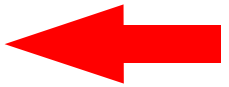
The Dependence Problem:

```
[-1,2]*[-1,2] = [-2,4]
```

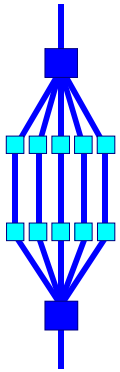


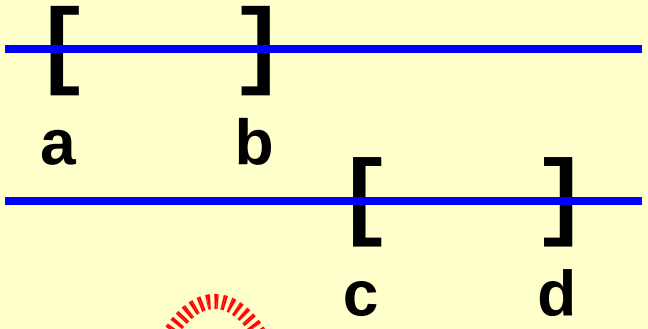
The Sun Compiler will do the right thing:

```
% f95 -o pow -xia pow.f95
% ./pow
X    = [ -1.00000000,  2.00000000]
X*X  = [ -2.00000000,  4.00000000]
X**2 = [  0.00000000,  4.00000000]
```

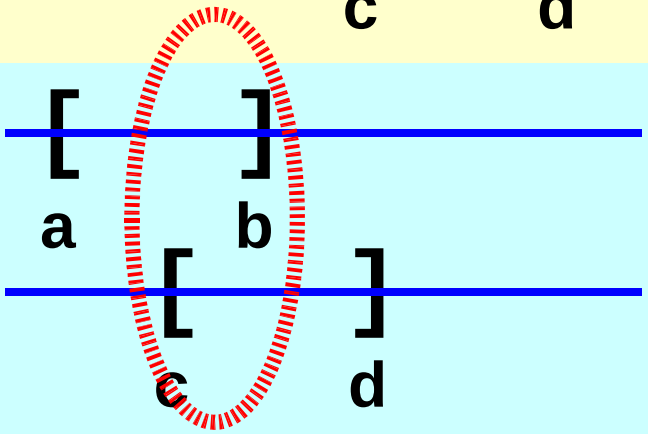


Order Relations - What To Do ?





[a,b] certainly less than [c,d]



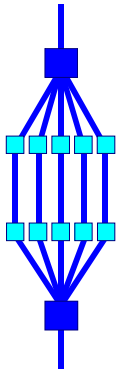
[a,b] possibly less than [c,d]

Implementation in the Sun compiler:

One of {C, P, S}, followed by LT/LE/EQ/NE/GE/GT

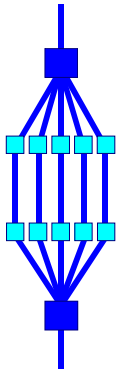
Example: A .CLT. B

Set-Theoretic Interval Operators



Name	Math. Notation	Fortran	Result Type
Interval hull	$X \cup Y$	X .IH. Y	Interval
Intersection	X ?Y	X .IX. Y	Interval
Disjoint	X ?Y = ∅	X .DJ. Y	Logical
Element	$r \in Y$	R .IN. Y	Logical
Interior	$\underline{X} < \underline{Y}$ and $\overline{X} < \overline{Y}$	X .INT. Y	Logical
Proper subset	$X \subset Y$	X .PSB. Y	Logical
Proper superset	$X \supset Y$	X .PSP. Y	Logical
Subset	$X \subseteq Y$	X .SB. Y	Logical
Superset	$X \supseteq Y$	X. SP. Y	Logical

Support For Intrinsic Functions

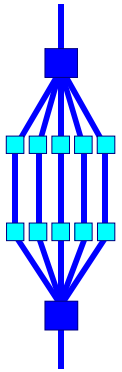


All Fortran intrinsic functions have an interval counterpart if they either return a REAL, or accept a REAL type argument

```
% cat -n cos.f95
  1    program demo
  2
  3 print *,'cos (-0.5)      = ',cos(-0.5D0)
  4    print *,'cos (+0.5)      = ',cos(+0.5D0)
  5 print *,'cos [-0.5,+0.5] = ',cos([-0.5,+0.5])
  6
  7 stop
  8 end
```

```
% f95 -o cos -xia cos.f95
% ./cos
 cos (-0.5)      =  0.8775825618903728
 cos (+0.5)      =  0.8775825618903728
 cos [-0.5,+0.5] =  [0.8775825618903726,1.0]
```

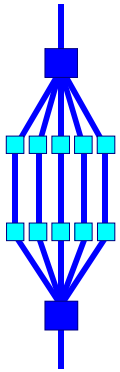
Interval Specific Intrinsics



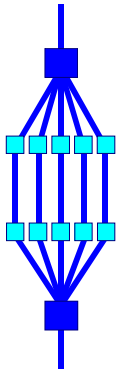
Name	Definition	Name	Result Type
Infimum	inf([a,b]) = a	INF	REAL
Supremum	sup([a,b]) = b	SUP	REAL
Width	w([a,b]) = b-a	WID	REAL
Midpoint	(a+b) / 2	MID	REAL
Magnitude	max(|a|,|b|)	MAG	REAL
Mignitude	min(|a|,|b|)*	MIG	REAL
Empty Test	TRUE if empty	ISEMPTY	LOGICAL
Number Of Digits	Max. digits	NDIGITS	INTEGER

) Returns 0 if 0 ∈ [a,b]

Input / Output

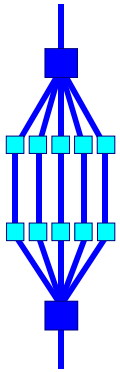


- *Square brackets (“[“and “]”) are used to delimit intervals*
  - *Example: X = [-0.1, 0.2]*
- *All edit descriptors that accept REAL data items also accept INTERVAL data*
- *Specific INTERVAL edit descriptors are supported as well*



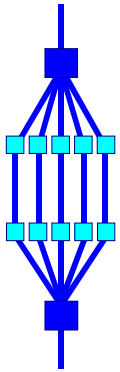
Quality Of Implementation Opportunities

Supported Features



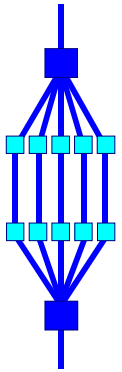
- *A closed interval system in which all expressions (including singularities and indeterminate forms) are defined*
  - *Examples: 1/0, x^y with x=y=0, operations involving $+\infty$ and/or $-\infty$*
- *Domain constraints on intrinsic functions are gracefully handled*
  - *Example: SQRT([-1 , +1]) = [0 , 1]*
- *Input / Output can be handled in different ways*
- *Context dependent literal interval constants*
- *Mixed mode expressions*

Example Code



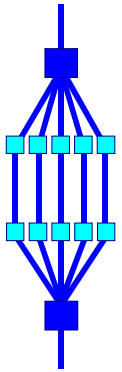
```
      Program Demo
      logical :: not_done = .true.
      interval(kind=8)    :: ai, bi
      write(*,*) 'Please give values for A and B'
      do while ( not_done )
         read(*,*,end=9000) ai, bi

         write(*,9010) '+',ai,'+',bi,ai+bi
         write(*,9010) '-',ai,'-',bi,ai-bi
         write(*,9010) '*',ai,'*',bi,ai*bi
         write(*,9010) '/',ai,'/',bi,ai/bi
         write(*,*)
      end do

9000  continue
      stop
9010  format(1X,'A',1X,(A),1X,'B =',VF17.4,1X,(A), &
      1X,VF17.4,' = ',VF17.4)
      end
```

Example Closed Interval System



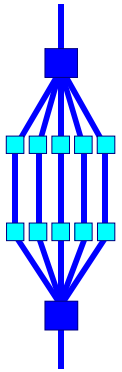
```
% f95 -xia math.f95
Please give values for A and B
A + B =[-1.0000, 3.0000] + [ 1.0000, 2.0000] = [ 0.0000, 5.0000]
A - B =[-1.0000, 3.0000] - [ 1.0000, 2.0000] = [-3.0000, 2.0000]
A * B =[-1.0000, 3.0000] * [ 1.0000, 2.0000] = [-2.0000, 6.0000]
A / B =[ 1.0000, 3.0000] / [ 1.0000, 2.0000] = [ 0.5000, 3.0000]

A + B =[ 1.0000, 2.0000] + [-1.0000, 3.0000] = [ 0.0000, 5.0000]
A - B =[ 1.0000, 2.0000] - [-1.0000, 3.0000] = [-2.0000, 3.0000]
A * B =[ 1.0000, 2.0000] * [-1.0000, 3.0000] = [-2.0000, 6.0000]
A / B =[ 1.0000, 2.0000] / [-1.0000, 3.0000] = [   -Inf,    Inf]
```

Summary



- *The Sun Fortran and C++ compilers support Interval Arithmetic*
- *The regular Basic Arithmetic Operations, intrinsic functions and logical operations have been extended to intervals*
- *In addition to this, several quality of implementation features are supported:*
  - *Closed interval system, domain constraints on intrinsic functions, input/output, ontext dependent literal interval constants, etc.*
- *We believe that this provides for a production quality interval compiler*

We Need Your Help

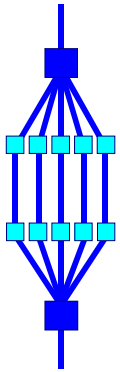


For Sun to continue interval support, we need you to use our compilers and give feedback

Feel free to send me an email:

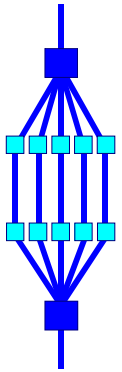
sambath.narayanan@sun.com

Intervals @ Sun
Connect Computing to the World



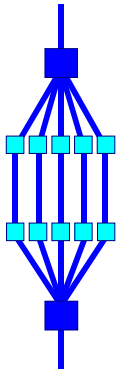
Thank You !

Why Is Interval Arithmetic Important?



- ***Interval arithmetic can be used to perform machine computations with guaranteed bounds on errors from all sources, including input data errors, machine rounding errors, and their interactions.***
- ***Interval algorithms can be used to solve nonlinear problems, such as the solution to nonlinear systems of equations and nonlinear programming (the nonlinear extension to linear programming).***
- ***As intervals become more widely used, libraries of interval solvers will be used routinely to compute sharp (narrow width) interval solutions to linear and nonlinear problems, while taking into account all sources of error. With these libraries, scientists, engineers, and developers of commercial applications will be able to write programs to solve problems that are currently beyond reach.***

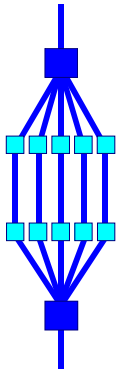
About Sun WorkShop 6 update 2 Interval Arithmetic



- ***Support for intrinsic `INTERVAL` data types is a feature in the Sun WorkShop 6 Fortran 95 compiler. Two new compiler flags, `-xia` and `-xinterval`, tell the compiler to recognize interval-specific language extensions and to generate executable interval code.***

- ***The Sun WorkShop 6 C++ compiler provides a C++ interface to the C++ interval arithmetic library. To use the C++ interval arithmetic features, add the `#include <suninterval.h>` header file to the code, and then compile the code using the `-xia` command-line option.***

Beam Dynamics



- *beam dynamics simulation and analysis code. It allows the study of accelerator lattices, spectrographs, beamlines, electron microscopes, and many other devices. It can determine high-order maps of combinations of particle optical elements of arbitrary field configurations. The elements can either be based on a large library of existing elements with realistic field configurations including fringe fields, or described in detail by measured data.*

- *Analysis options include computation of high-order nonlinearities; analysis of properties of repetitive motion via chromaticities, normal form analysis, and symplectic tracking; analysis of single-pass systems resolutions, reconstructive aberration correction, and consideration of detector errors; and analysis of spin dynamics via computation of spin maps, spin normal form and spin tracking.*