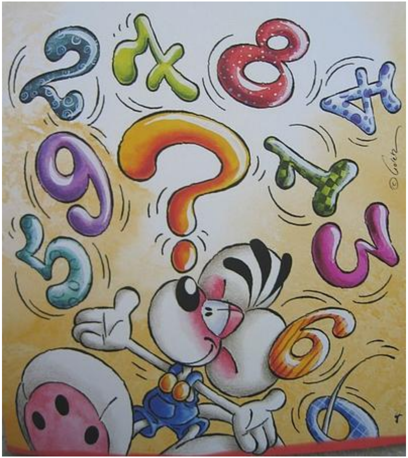


Diddl - Goletz

(cf. http://ymonka.free.fr/maths-et-tiques/index.php/histoire-des-maths/nombres/histoire-des-nombres)

Calcul numérique sur des ensembles, vérification de calculs numériques : un outil de choix, l'arithmétique par intervalles

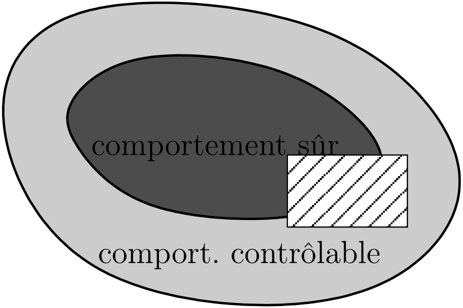
Nathalie Revol

19 avril 2011

Calcul ensembliste : exemples d'applications

On peut

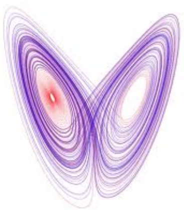
- déterminer si, avec l'ensemble des valeurs possibles pour les paramètre d'entrée, le système reste dans une zone sûre, contrôlable, incontrôlable ;



Calcul ensembliste : exemples d'applications

On peut

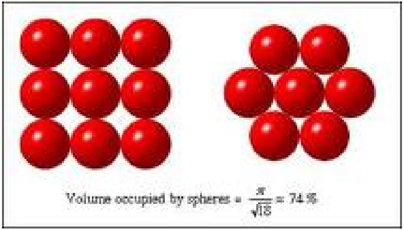
- déterminer si un robot peut aller d'un point A à un point B d'un bâtiment ;
- déterminer si un piano peut effectuer le trajet depuis la porte de votre immeuble jusquà votre salon
- suivre la trajectoire d'une particule ;
- prouver que le système de Lorenz admet un attracteur ;

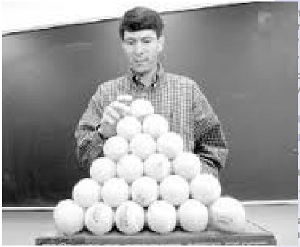


Calcul ensembliste : exemples d'applications

On peut

- prouver la conjecture de Kepler : 





- . . .

Calcul ensembliste : exemples de problèmes

Sur tout un ensemble $\boldsymbol{X}$ (un pavé) on peut

- garantir le signe d'une fonction ;
- garantir qu'un polynôme n'a pas de racines ;
- prouver l'existence, l'existence et l'unicité d'une racine ;
- déterminer l'optimum global d'une fonction « assez » continue.

Calcul ensembliste : exemples de problèmes

Sur tout un ensemble $\boldsymbol{X}$ (un pavé) on peut

- garantir le signe d'une fonction ;
- garantir qu'un polynôme n'a pas de racines ;
- prouver l'existence, l'existence et l'unicité d'une racine ;
- déterminer l'optimum global d'une fonction « assez » continue.

Théorème de Brouwer :

- soit f une fonction continue sur un compact K, si $f(K) \subset K$ alors f admet un point fixe dans K :
$$\exists x^* \in K,\ f(x^*) = x^*;$$
- si $f(K) \subset \mathrm{int}(K)$ alors le point fixe x^* est unique :
$$\exists ! x^* \in K,\ f(x^*) = x^*.$$

Calcul ensembliste : limites

On ne sait pas (encore)

- manipuler autre chose que des pavés ;
- intégrer sur des temps très longs ;
- intégrer des EDP ;
- résoudre des problèmes avec beaucoup de variables
- ...

Calcul ensembliste : arithmétique par intervalles

(Moore 1966, Kulisch 1983, Neumaier 1990, Rump 1994, Alefeld and Mayer 2000, Rump 2010. . .)

Les ensembles les plus simples à manipuler sont les intervalles.

Principe

Nombres remplacés par des intervalles.
π remplacé par $[3.14159, 3.14160]$

Contenu de mon porte-monnaie : entre 10 Euros et 20 Euros, $\in [10, 20]$ Euros.

Arithmétique par intervalles : calculer

$$[10, 20] + [5, 10] = [15, 30]$$
$$[-2, 3] + [5, 7] = [3, 10]$$

$[-3, 2] * [-3, 2] = [-6, 9]$ est différent de $[-3, 2]^2 = [0, 9]$

$$[-3, 2]/[0.5, 1] = [-6, 4]$$

$$X \diamond Y = \{x \diamond y \,/\, x \in X,\ y \in Y\}$$

$$\exp[-2, 3] = [\exp(-2), \exp(3)]$$

car exp est une fonction croissante.

$$\sin[\pi/3, \pi] = [0, 1]$$

attention, sin n'est pas monotone.

Arithmétique par intervalles : avantages

Prise en compte des incertitudes
données expérimentales, résultats de mesures physiques, valeurs non exactement représentables.

Calcul garanti
le résultat cherché appartient à l'intervalle calculé.

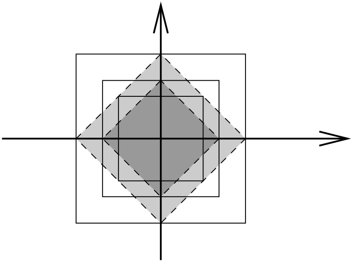
Information globale
on sait encadrer l'image d'une fonction sur tout un intervalle.

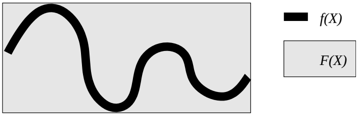
Arithmétique par intervalles : problèmes

Décorrélation des variables ou *variable dependency*

$$I - I = \{x - y \, ; \, x \in I, \, y \in I\} \supset \{0\} = \{x - x \, ; \, x \in I\}$$

Effet enveloppant ou *wrapping effect*





Arithmétique par intervalles : historique

Historique

- **1962 :** Ramon Moore dans sa thèse puis définition de faccon très complète et publication en 1966

Arithmétique par intervalles : historique

Historique

- **1962 :** Ramon Moore dans sa thèse puis définition de fa*c*con très complète et publication en 1966
- **1958 :** Tsunaga, dans son mémoire de master en japonais

Arithmétique par intervalles : historique

Historique

- **1962 :** Ramon Moore dans sa thèse puis définition de fa*c*con très complète et publication en 1966
- **1958 :** Tsunaga, dans son mémoire de master en japonais
- **1956 :** Warmus

Arithmétique par intervalles : historique

Historique

- **1962 :** Ramon Moore dans sa thèse puis définition de fa*c*con très complète et publication en 1966
- **1958 :** Tsunaga, dans son mémoire de master en japonais
- **1956 :** Warmus
- **1951 :** Dwyer dans le cas particulier des intervalles fermés

Arithmétique par intervalles : historique

Historique

- **1962 :** Ramon Moore dans sa thèse puis définition de fa*c*con très complète et publication en 1966
- **1958 :** Tsunaga, dans son mémoire de master en japonais
- **1956 :** Warmus
- **1951 :** Dwyer dans le cas particulier des intervalles fermés
- **1931 :** Rosalind Cecil Young, avec quelques formules dans sa thèse de doctorat à l'Université de Cambridge

Arithmétique par intervalles : historique

Historique

- **1962 :** Ramon Moore dans sa thèse puis définition de faccon très complète et publication en 1966
- **1958 :** Tsunaga, dans son mémoire de master en japonais
- **1956 :** Warmus
- **1951 :** Dwyer dans le cas particulier des intervalles fermés
- **1931 :** Rosalind Cecil Young, avec quelques formules dans sa thèse de doctorat à l'Université de Cambridge
- **1927 :** Bradis, pour des quantités positives, en russe

Arithmétique par intervalles : historique

Historique

- **1962 :** Ramon Moore dans sa thèse puis définition de faccon très complète et publication en 1966
- **1958 :** Tsunaga, dans son mémoire de master en japonais
- **1956 :** Warmus
- **1951 :** Dwyer dans le cas particulier des intervalles fermés
- **1931 :** Rosalind Cecil Young, avec quelques formules dans sa thèse de doctorat à l'Université de Cambridge
- **1927 :** Bradis, pour des quantités positives, en russe
- **1908 :** Young, pour certaines fonctions bornées, en italien

Arithmétique par intervalles : historique

Historique

- **1962 :** Ramon Moore dans sa thèse puis définition de faccon très complète et publication en 1966
- **1958 :** Tsunaga, dans son mémoire de master en japonais
- **1956 :** Warmus
- **1951 :** Dwyer dans le cas particulier des intervalles fermés
- **1931 :** Rosalind Cecil Young, avec quelques formules dans sa thèse de doctorat à l'Université de Cambridge
- **1927 :** Bradis, pour des quantités positives, en russe
- **1908 :** Young, pour certaines fonctions bornées, en italien
- **3e siècle avant J.-C. :** Archimède pour calculer un encadrement de π !

Exemples d'algorithmes

Deux algorithmes pour illustrer l'utilisation du calcul ensembliste et de l'arithmétique par intervalles :

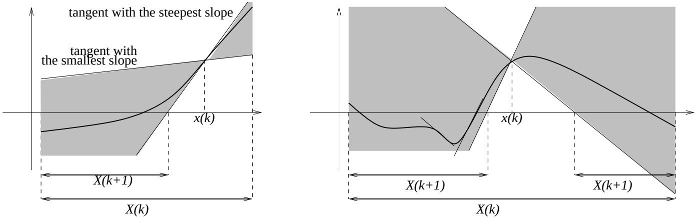
- **méthode de Newton** pour rechercher les zéros d'une fonction $\mathcal{C}^1$;
- **algorithme de Hansen** pour rechercher les optima globaux d'une fonction $\mathcal{C}^2$

sur un intervalle de départ donné.

Les exemples seront en 1 variable. . . pour pouvoir dessiner.

Algorithme de Newton par intervalles

(Hansen & Greenberg 1983, Mayer 1995, van Hentenryck et al. 1997. . .)



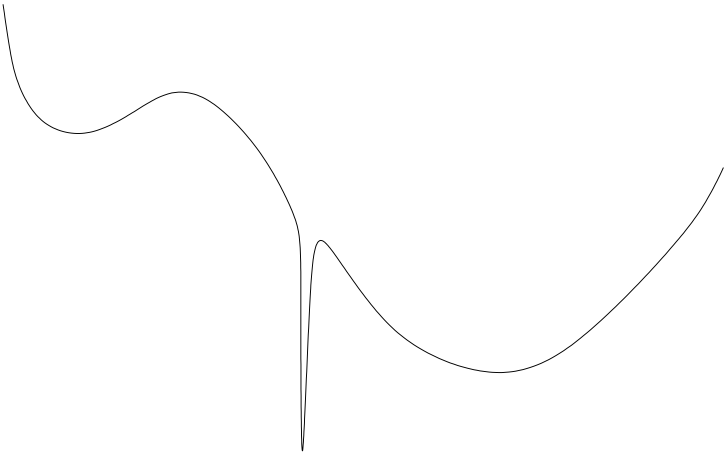
The result will be a list of intervals.

Théorème de Brouwer :

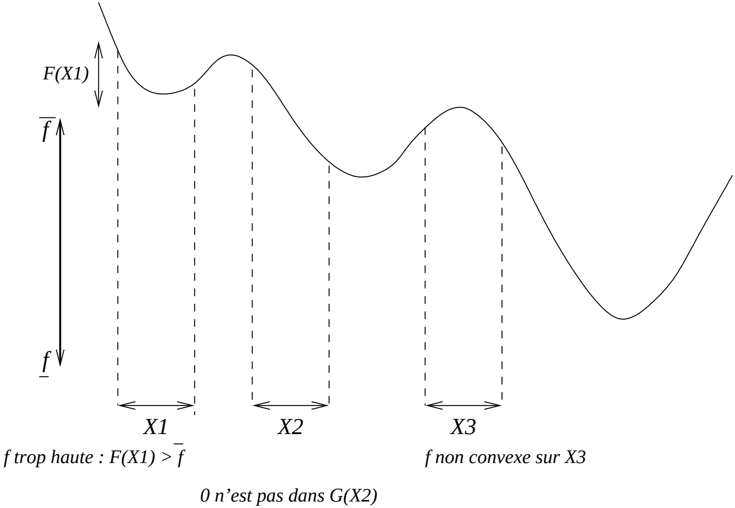
si $f(I) \subset I$ alors f admet un point fixe dans I.
permet de prouver l'existence (et l'unicité) de la solution.

Algorithme de Hansen par intervalles

(Hansen 1992)



Algorithme de Hansen par intervalles



Algorithme de Hansen

$\mathcal{L}$ = liste des pavés en attente $:= \{X_0\}$

tant que $\mathcal{L} \neq \emptyset$ faire

  sortir X de $\mathcal{L}$

  rejeter X **?**

    oui si $F(X) > \bar{f}$

    oui si $\mathrm{Grad}F(X) \not\ni 0$

    oui si $HF(X)$ à diagonale non > 0

  réduire X

    Newton sur le gradient

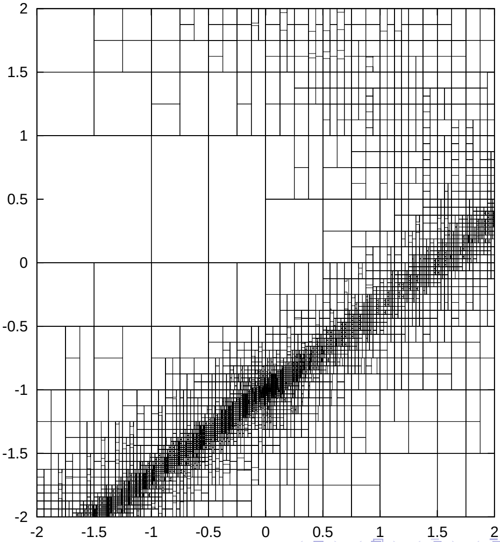
    résoudre $Y \subset X$ tel que $F(Y) \leq \bar{f}$

  couper Y **en 2** : Y_1 et Y_2

    ranger Y_1 et Y_2 dans $\mathcal{L}$

Algorithme de Hansen par intervalles

(Hansen 1992)



Algorithmique par intervalles et complexité :

presque tous les problèmes sont NP-durs

. . . même à ε près

Gaganov 1982, Rohn 1994 ff, Kreinovich. . .

- évaluer une fonction sur un pavé (un produit cartésien d'intervalles) ;
- évaluer une fonction sur un pavé à ε près '
- résoudre un système linéaire ;
- résoudre un système linéaire à $1/4n^4$ près ($n =$ dim. du système) ;
- déterminer si la solution d'un système linéaire est bornée ;
- calculer la norme matricielle $\|\boldsymbol{A}\|_{\infty,1}$;
- déterminer si un intervalle matriciel (ou une matrice à coefficients intervalles) est régulière, c-à-d si toute matrice qu'elle contient est régulière ;
- . . .

Digression : les nombres flottants

Exemple :
$\pi \simeq 3,14 \times 10^0 = 0,314 \times 10^1 = 0,031 \times 10^2 = 314 \times 10^{-2} \ldots$

Convention : on met un chiffre non nul avant la virgule :
$\pi \simeq 3,14 \times 10^0$.

Représentation :
nombre de chiffres utilisés dans la représentation : constant, dans notre exemple, 3 chiffres décimaux.
Population de Lyon : 445 452 habitants en 1999, représenté par $4,45 \times 10^5$.

Dénomination : notation scientifique dans la vie courante, nombre à virgule flottante ou par ellipse **nombre flottant**.

Les nombres flottants : définition

Exemple : $4.67 \times 10^3 + 3,14 \times 10^0$?

$$
\begin{array}{rl}
 & 4,67 \quad\quad\;\; \text{x } 10^3 \\
+ & 0,00314 \;\; \text{x } 10^3 \\
\hline
 & 4,67314 \;\; \text{x } 10^3 \\
\simeq & 4,67 \quad\quad\;\; \text{x } 10^3
\end{array}
$$

Les nombres flottants

Nombre de chiffres fixé $\rightarrow$ arrondi obligatoire.

Comment arrondir ?

- vers le haut
- vers le bas
- vers 0
- en s'éloignant de 0
- au plus près : dans ce cas, comment arrondir 6,785 ?

Besoin d'une norme

- sur certaines machines Cray, on avait

$$\text{parfois, } 1 \times x \Rightarrow \text{ overflow}$$
$$x + y \neq y + x$$
$$0.5 \times x \neq x/2.0$$

- IBM 370, en Fortran, on avait

$$\sqrt{-4} = 2$$
$$\texttt{I = 14.0/7.0} \rightarrow I = 1$$

Les nombres flottants : la norme IEEE-754

Jusqu'en 1985 : anarchie totale dans le monde des processeurs.

En 1985 : adoption de la norme IEEE-754

- formats fixés :
  simple précision et double précision
- arrondis possibles : vers $+\infty$, vers $-\infty$, vers 0 et au plus près (pair)
- opérations arithmétiques : $+$, $-$, $\times$, $/$ et $\sqrt{}$ doivent rendre l'arrondi du résultat exact (toujours possible avec 3 bits supplémentaires pendant les calculs).

Norme IEEE-754 : avantages et inconvénients

Avantages :

- représentation en ordre de grandeur
- opérations en temps constant (puisque nombre de chiffres constant)
- norme IEEE-754 : opérations les plus précises possible
- norme IEEE-754 : reproductibilité des calculs d'un ordinateur à l'autre
- norme IEEE-754 : calculs bien spécifiés, possibilité de prouver qu'un programme est correct.

Inconvénients :

- résultats toujours arrondis
- précision limitée

Norme IEEE-754 : valeurs spéciales

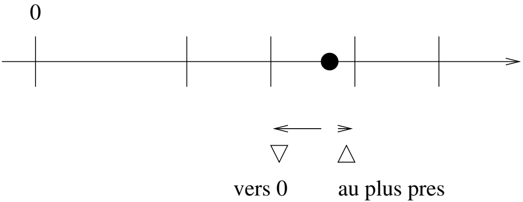
les infinis : $-\infty$ et $+\infty$

Not a Number : NaN

- permet de représenter le résultat d'une opération invalide telle que $\sqrt{-2}$, $0/0$ ou $\infty - \infty$
- se propage dans les calculs, en contaminant les résultats.

Norme IEEE-754 : modes d'arrondi

- **calculs**
  π : non représentable par un flottant
  produit de 2 nombres flottants : ne rentre pas dans un flottant
- **arrondis**



Norme IEEE-754 : modes d'arrondis

Quatre modes d'arrondi :

- **arrondi vers $+\infty$** ou par excès, noté $\triangle(x)$: retourne le plus petit nombre machine supérieur ou égal au résultat exact x ;
- **arrondi vers $-\infty$** ou par défaut noté $\bigtriangledown(x)$: retourne le plus grand nombre machine inférieur ou égal au résultat exact x ;
- **arrondi vers 0** : retourne $\triangle(x)$ pour les nombres négatifs et $\bigtriangledown(x)$ pour les nombres positifs ;
- **arrondi au plus près** : retourne le nombre machine le plus proche du résultat exact x, celui dont la mantisse se termine par 0 pour le milieu de deux nombres machine consécutifs (on parle d'arrondi pair).

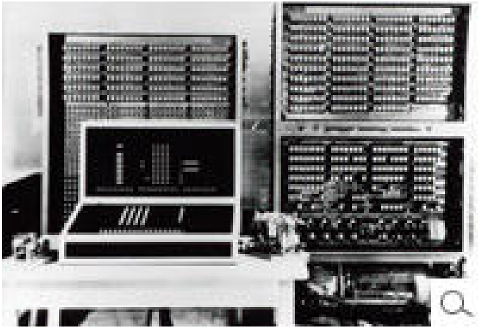
Norme IEEE-754 : arrondi correct

Arrondi correct : soient a et b deux nombres machine et $\odot$ une opération parmi $\{+, -, \times, /\}$ avec o l'un des 4 arrondis prévus par la norme,
le résultat $a \odot b$ doit être $o(a\dot{b})$ l'arrondi du résultat exact.

On a la même exigence pour la racine carrée.

Premier ordinateur flottant ?

Le premier ordinateur avec une arithmétique flottante date de 1941 et a été créé par Konrad Zuse.



Cf. http://www.computerhistory.org/timeline/?year=1941

Retour à l'arithmétique par intervalles

Grâce aux arrondis dirigés (vers $\pm\infty$), on peut implanter sur ordinateur l'arithmétique par intervalles
en conservant la propriété d'inclusion « Thou shalt not lie » : le résultat cherché appartient à l'intervalle retourné.

Retour à l'arithmétique par intervalles

Grâce aux arrondis dirigés (vers $\pm\infty$), on peut implanter sur ordinateur l'arithmétique par intervalles
en conservant la propriété d'inclusion « Thou shalt not lie » : le résultat cherché appartient à l'intervalle retourné.

On arrondit :

- vers $-\infty$ la borne gauche,
- vers $+\infty$ la borne droite.

Arithmétique par intervalles : implantation OK, efficacité ?

Surcoût théorique :

- pour une addition (ou une soustraction) :

$$[\underline{a}, \bar{a}] + [\underline{b}, \bar{b}] = [\triangledown(\underline{a} + \underline{b}), \triangle(\bar{a} + \bar{b})]$$

  soit 2 fois plus d'opérations ;
- pour une multiplication :

$$[\underline{a}, \bar{a}]\times[\underline{b}, \bar{b}] = [\triangledown(\max(\underline{a}\times\underline{b}, \underline{a}\times\bar{b}, \bar{a}\times\underline{b}, \bar{a}\times\bar{b})), \triangle(\max(\underline{a}\times\underline{b}, \underline{a}\times\bar{b}, \bar{a}\times\underline{b}, \bar{a}\times\bar{b}))]$$

  soit 8 fois plus d'opérations (et non 4)
- $\sin[\underline{a}, \bar{a}]$: beaucoup plus d'opérations et 2 calculs de sinus.

Arithmétique par intervalles : implantation OK, efficacité ?

Surcoût pratique (observé) :

- pour une addition (ou une soustraction) :

$$[\underline{a}, \bar{a}] + [\underline{b}, \bar{b}] = [\triangledown(\underline{a} + \underline{b}), \triangle(\bar{a} + \bar{b})]$$

  2 fois plus d'opérations
  et un changement de mode d'arrondi : sur la plupart des processeurs, un facteur 10 à 20 ;
- surcoût observé en pratique : au moins un facteur 20, voire pire.

Arithmétique par intervalles : vérification de calculs numériques

Idée (naïve) : vérifier les calculs numériques

c-à-d calculer un intervalle contenant le résultat et les effets de tous les arrondis.

Comment : en rempla*c*cant le type `double` par `interval`.

Arithmétique par intervalles : vérification de calculs numériques

Idée (naïve) : vérifier les calculs numériques

c-à-d calculer un intervalle contenant le résultat et les effets de tous les arrondis.

Comment : en rempla*c*cant le type `double` par `interval`.

C'est lent : cf. remarque précédente sur l'efficacité.

Arithmétique par intervalles : vérification de calculs numériques

Idée (naïve) : vérifier les calculs numériques
c-à-d calculer un intervalle contenant le résultat et les effets de tous les arrondis.
Comment : en remplaçcant le type `double` par `interval`.

C'est lent : cf. remarque précédente sur l'efficacité.

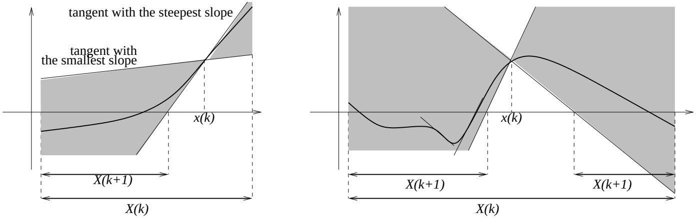
Cela ne marche pas : méthode de Newton.

Formule ponctuelle : $x_{k+1} = x_k - \frac{f(x_k)}{f'(x_k)}$

Équivalent intervalle : $\boldsymbol{x}_{k+1} = \boldsymbol{x}_k - \frac{f(\boldsymbol{x}_k)}{f'(\boldsymbol{x}_k)}$

$$w(\boldsymbol{x}_{k+1}) = w(\boldsymbol{x}_k) + w\left(\frac{f(\boldsymbol{x}_k)}{f'(\boldsymbol{x}_k)}\right) > w(\boldsymbol{x}_k)$$

divergence !



The result will be a list of intervals.

Problem : verified solution of a linear system

Goals : For a linear system $Ax = b$ with $A \in \mathbb{F}^{n \times n}$ non-singular and $b \in \mathbb{F}^n$, we want to

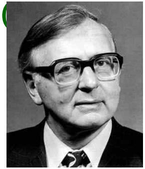
1. compute an approximation $\tilde{x} \in \mathbb{F}^n$ of the exact solution x^*,
2. simultaneously bound the error upon $\tilde{x}$, or enclose it in an interval

$$\boldsymbol{e} \ni x^* - \tilde{x}.$$

Remark : denote by e the error $x^* - \tilde{x}$.
Then e is the solution of the residual system $Ae = b - A\tilde{x}$.
Indeed, $Ae = A(x^* - \tilde{x}) = Ax^* - A\tilde{x} - b - A\tilde{x}$.

Method : contractant iteration
Classical iterative refinement

Wilkinson (1963), Higham (2000), Demmel *et al.* (



Algorithm (Classical iterative refinement)

Input : $A \in \mathbb{F}^{n \times n}, b \in \mathbb{F}^{n}$

$\tilde{x} = A \setminus b$ *% MatLab-like syntax*

while(not converged)

$\quad \tilde{r} = b - A\,\tilde{x}$

$\quad \tilde{e} = A \setminus \tilde{r}$

$\quad \tilde{x} = \tilde{x} + \tilde{e}$

end

Output : $\tilde{x}$

Method : contractant iteration
Interval iterative refinement

Neumaier (1990), Rump (1999)

Algorithm (certifylss)

Input : $A \in \mathbb{F}^{n \times n}, b \in \mathbb{F}^n$

$\tilde{x} = A \setminus b$ % *MatLab-like syn*

while(not converged)

$\quad \tilde{r} = b - A\,\tilde{x}$

$\quad \tilde{e} = A \setminus \tilde{r}$

$\quad \tilde{x} = \tilde{x} + \tilde{e}$

end

Output : $\tilde{x}$

Method : contractant iteration
Interval iterative refinement

Neumaier (1990), Rump (1999)

Algorithm (certifylss)

Input : $A \in \mathbb{F}^{n \times n}, b \in \mathbb{F}^n$

$\tilde{x} = A \setminus b$ % *MatLab-like syn*

while(not converged)

$\quad r = [b - A\,\tilde{x}]$ % $A(x^* - \tilde{x}) \in r$

$\quad \tilde{e} = A \setminus \tilde{r}$

$\quad \tilde{x} = \tilde{x} + \tilde{e}$

end

Output : $\tilde{x}$

Method : contractant iteration
Interval iterative refinement

Neumaier (1990), Rump (1999)

Algorithm (certifylss)

Input : $A \in \mathbb{F}^{n\times n}, b \in \mathbb{F}^n$

$\tilde{x} = A \setminus b$ % *MatLab-like syn*

while(not converged)

$\quad r = [b - A\,\tilde{x}]$ % $A(x^* - \tilde{x}) \in r$

$\quad e = A \setminus r$ % $x^* - \tilde{x} \in e$

$\quad \tilde{x} = \tilde{x} + \tilde{e}$

end

Output : $\tilde{x}$

Method : contractant iteration
Interval iterative refinement

Neumaier (1990), Rump (1999)

Algorithm (certifylss)

Input : $A \in \mathbb{F}^{n\times n}, b \in \mathbb{F}^n$

$\tilde{x} = A \setminus b$ *% MatLab-like sy*

while(not converged)

     $\boldsymbol{r} = [b - A\,\tilde{x}]$ % $A(x^* - \tilde{x}) \in \boldsymbol{r}$

     $\boldsymbol{e} = A \setminus \boldsymbol{r}$ % $x^* - \tilde{x} \in \boldsymbol{e}$

     $\tilde{x} = \tilde{x} + \mathrm{mid}(\boldsymbol{e}), \quad \boldsymbol{e} = \boldsymbol{e} - \mathrm{mid}(\boldsymbol{e})$

end

Output : $\tilde{x}$

Method : contractant iteration
Interval iterative refinement

Neumaier (1990), Rump (1999)

Algorithm (certifylss)

Input : $A \in \mathbb{F}^{n\times n}, b \in \mathbb{F}^n$

$\tilde{x} = A \setminus b$ *% MatLab-like syn*

while(not converged)

     $\boldsymbol{r} = [b - A\,\tilde{x}]$ % $A(x^* - \tilde{x}) \in \boldsymbol{r}$

     $\boldsymbol{e} = A \setminus \boldsymbol{r}$ % $x^* - \tilde{x} \in \boldsymbol{e}$

     $\tilde{x} = \tilde{x} + \mathrm{mid}(\boldsymbol{e}), \quad \boldsymbol{e} = \boldsymbol{e} - \mathrm{mid}(\boldsymbol{e})$

end

Output : $\boldsymbol{x} = \tilde{x} + \boldsymbol{e}$

Method : contractant iteration Interval iterative refinement

Neumaier (1990), Rump (1999)

Algorithm (certifylss)

Input : $A \in \mathbb{F}^{n\times n}, b \in \mathbb{F}^n$

$\tilde{x} = A \setminus b$ % *MatLab-like sy*

while(not converged)

$\quad \boldsymbol{r} = [b - A\,\tilde{x}]$ % $A(x^* - \tilde{x}) \in \boldsymbol{r}$

$\quad \boldsymbol{e} = A \setminus \boldsymbol{r}$ % $x^* - \tilde{x} \in \boldsymbol{e}$

$\quad \tilde{x} = \tilde{x} + \mathrm{mid}(\boldsymbol{e}), \quad \boldsymbol{e} = \boldsymbol{e} - \mathrm{mid}(\boldsymbol{e})$

end

Output : $\boldsymbol{x} = \tilde{x} + \boldsymbol{e}$

Solving interval residual system ?

Method : contractant iteration
Interval iterative refinement

Neumaier (1990), Rump (1999)

Algorithm (certifylss)

Input : $A \in \mathbb{F}^{n \times n}, b \in \mathbb{F}^n$

$\tilde{x} = A \setminus b, \qquad R = inv(A), \qquad \boldsymbol{K} = [RA]$

while(not converged)

$\quad \boldsymbol{r} = [b - A\,\tilde{x}] \qquad \% \quad A(x^* - \tilde{x}) \in \boldsymbol{r}$

$\quad \boldsymbol{e} = A \setminus \boldsymbol{r} \qquad \% \quad x^* - \tilde{x} \in \boldsymbol{e}$

$\quad \tilde{x} = \tilde{x} + \mathrm{mid}(\boldsymbol{e}), \qquad \boldsymbol{e} = \boldsymbol{e} - \mathrm{mid}(\boldsymbol{e})$

end

Output : $\boldsymbol{x} = \tilde{x} + \boldsymbol{e}$

$\boldsymbol{K}$ is close to Identity $\Rightarrow$ there are algorithms to solve this system.

Method : contractant iteration
Interval iterative refinement

Neumaier (1990), Rump (1999)

Algorithm (certifylss)

Input : $A \in \mathbb{F}^{n\times n}, b \in \mathbb{F}^n$

$\tilde{x} = A \setminus b, \qquad R = inv(A), \qquad K = [RA]$

while(not converged)

$\quad r = [Rb - K\,\tilde{x}] \qquad \%\quad RA(x^* - \tilde{x})$

$\quad e = A \setminus r \qquad \%\quad x^* - \tilde{x} \in e$

$\quad \tilde{x} = \tilde{x} + \mathrm{mid}(e), \quad e = e - \mathrm{mid}(e)$

end

Output : $x = \tilde{x} + e$

K is close to Identity $\Rightarrow$ there are algorithms to solve this system.

Method : contractant iteration Interval iterative refinement

Neumaier (1990), Rump (1999)

Algorithm (certifylss)

Input : $A \in \mathbb{F}^{n \times n}, b \in \mathbb{F}^n$

$\tilde{x} = A \setminus b, \quad R = inv(A), \quad K = [RA]$

while(not converged)

$\quad r = [Rb - K\,\tilde{x}] \quad \% \quad RA(x^* - \tilde{x})$

$\quad e = K \setminus r \quad \% \quad x^* - \tilde{x} \in e$

$\quad \tilde{x} = \tilde{x} + \mathrm{mid}(e), \quad e = e - \mathrm{mid}(e)$

end

Output : $x = \tilde{x} + e$

K is close to Identity $\Rightarrow$ there are algorithms to solve this system.

Method : contractant iteration Interval iterative refinement

Neumaier (1990), Rump (1999)

Algorithm (certifylss)

Input : $A \in \mathbb{F}^{n \times n}, b \in \mathbb{F}^n$

$\tilde{x} = A \setminus b, \qquad R = inv(A), \qquad K = [RA]$

while(not converged)

$\quad r = [Rb - K\,\tilde{x}] \qquad \% \quad RA(x^* - \tilde{x})$

$\quad e = K \setminus r \qquad \% \quad x^* - \tilde{x} \in e$

$\quad \tilde{x} = \tilde{x} + \mathrm{mid}(e), \quad e = e - \mathrm{mid}(e)$

end

Output : $x = \tilde{x} + e$

K is close to Identity $\Rightarrow$ there are algorithms to solve this system.

Method : contractant iteration
Interval iterative refinement

Neumaier (1990), Rump (1999)

Algorithm (certifylss)

Input : $A \in \mathbb{F}^{n\times n}$, $b \in \mathbb{F}^n$

$\tilde{x} = A \setminus b$, $\quad R = inv(A)$, $\quad K = [RA]$

while(not converged)

$\quad r = [Rb - K\,\tilde{x}]$ $\qquad$ % $\quad RA(x^* - \tilde{x})$

$\quad e = K \setminus r$ $\qquad$ % $\quad x^* - \tilde{x} \in e$

$\quad \tilde{x} = \tilde{x} + \mathrm{mid}(e)$, $\quad e = e - \mathrm{mid}(e)$

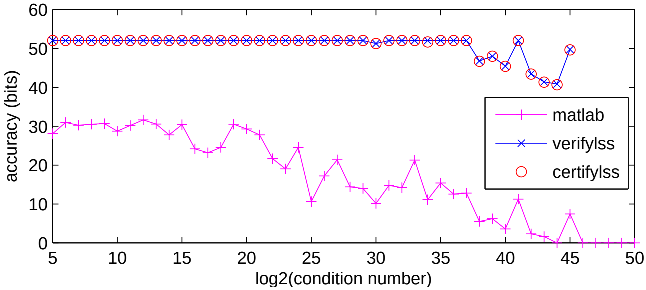
end

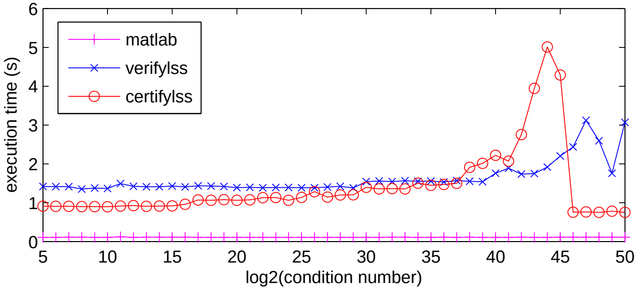
Output : $x = \tilde{x} + e$

This algorithm can fail, if it fails to solve the interval linear system.

Experimental Results

Dimension : 1000 $\qquad b = [1, \ldots, 1]^T$





Method : contractant iteration
Relaxed interval iterative refinement

Algorithm (certifylss_relaxed)

Input : $A \in \mathbb{F}^{n \times n}, b \in \mathbb{F}^n$

$\tilde{x} = A \setminus b, \qquad R = inv(A), \qquad \mathbf{K} = [RA]$

while(not converged)

$\quad \mathbf{r} = [Rb - \mathbf{K}\ \tilde{x}] \qquad \% \quad RA(x^* - \tilde{x}) \in \mathbf{r}$

$\quad \mathbf{e} = \mathbf{K} \setminus \mathbf{r} \qquad \% \quad x^* - \tilde{x} \in \mathbf{e}$

$\quad \tilde{x} = \tilde{x} + \mathrm{mid}(\mathbf{e}), \quad \mathbf{e} = \mathbf{e} - \mathrm{mid}(\mathbf{e})$

end

Output : $\mathbf{x} = \tilde{x} + \mathbf{e}$

Method : contractant iteration
Relaxed interval iterative refinement

Algorithm (certifylss_relaxed)

Input : $A \in \mathbb{F}^{n \times n}, b \in \mathbb{F}^n$

$\tilde{x} = A \setminus b, \quad R = inv(A), \quad \mathbf{K} = [RA],$

$\hat{\mathbf{K}} = inflated(\mathbf{K})$ *% $\hat{\mathbf{K}}$ is centered on a diagonal matrix*

while(not converged)

$\mathbf{r} = [Rb - \mathbf{K}\,\tilde{x}]$ % $RA(x^* - \tilde{x}) \in \mathbf{r}$

$\mathbf{e} = \mathbf{K} \setminus \mathbf{r}$ % $x^* - \tilde{x} \in \mathbf{e}$

$\tilde{x} = \tilde{x} + \mathrm{mid}(\mathbf{e}), \quad \mathbf{e} = \mathbf{e} - \mathrm{mid}(\mathbf{e})$

end

Output : $\mathbf{x} = \tilde{x} + \mathbf{e}$

Method : contractant iteration
Relaxed interval iterative refinement

Algorithm (certifylss_relaxed)

Input : $A \in \mathbb{F}^{n\times n}$, $b \in \mathbb{F}^n$

$\tilde{x} = A \setminus b$, $\quad R = inv(A)$, $\quad \boldsymbol{K} = [RA]$,

$\hat{\boldsymbol{K}} = inflated(\boldsymbol{K})$ $\quad$ % $\hat{\boldsymbol{K}}$ *is centered on a diagonal matrix*

while(not converged)

$\quad \boldsymbol{r} = [Rb - \boldsymbol{K}\,\tilde{x}]$ $\quad$ % $\quad RA(x^* - \tilde{x}) \in \boldsymbol{r}$

$\quad \boldsymbol{e} = \hat{\boldsymbol{K}} \setminus \boldsymbol{r}$ $\quad$ % *cost : 1 floating-point matrix-vector product*

$\quad \tilde{x} = \tilde{x} + \mathrm{mid}(\boldsymbol{e})$, $\quad \boldsymbol{e} = \boldsymbol{e} - \mathrm{mid}(\boldsymbol{e})$

end

Output : $\boldsymbol{x} = \tilde{x} + \boldsymbol{e}$

Method : contractant iteration
Relaxed interval iterative refinement

Algorithm (certifylss_relaxed)

Input : $A \in \mathbb{F}^{n\times n}, b \in \mathbb{F}^n$

$\tilde{x} = A \setminus b, \qquad R = inv(A), \qquad \mathbf{K} = [RA],$

$\hat{\mathbf{K}} = inflated(\mathbf{K})$ $\qquad$ % $\hat{\mathbf{K}}$ *is centered on a diagonal matrix*

while(not converged)

$\quad \mathbf{r} = [Rb - \mathbf{K}\,\tilde{x}]$ $\qquad$ % $\quad RA(x^* - \tilde{x}) \in \mathbf{r}$

$\quad \mathbf{e} = \hat{\mathbf{K}} \setminus \mathbf{r}$ $\qquad$ % *cost : 1 floating-point matrix-vector product*

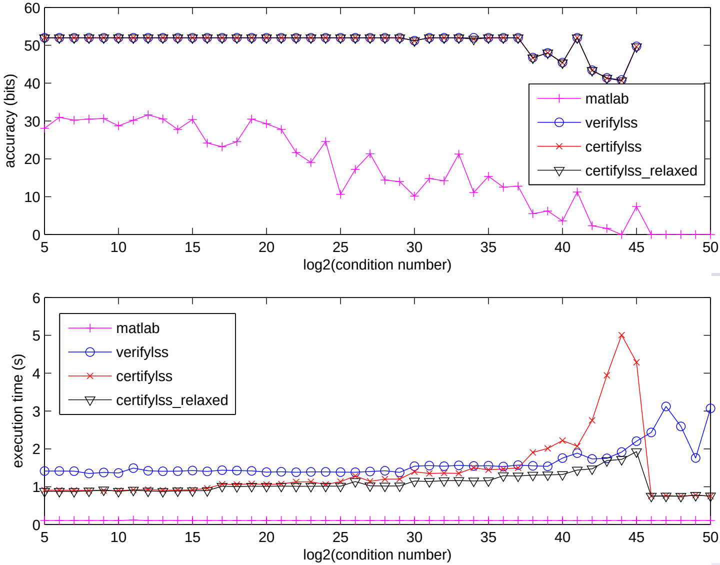
$\quad \tilde{x} = \tilde{x} + \mathrm{mid}(\mathbf{e}), \quad \mathbf{e} = \mathbf{e} - \mathrm{mid}(\mathbf{e})$

end

Output : $\mathbf{x} = \tilde{x} + \mathbf{e}$

Relaxed method : Results

Dimension : 1000 $\qquad b = [1, \ldots, 1]^T$



Method : contractant iteration Extra-precise relaxed interval iterative refinement

Algorithm (certifylssx)

Input : $A \in \mathbb{F}^{n \times n}, b \in \mathbb{F}^n$

$\tilde{x} = A \setminus b, \qquad R = inv(A), \qquad \boldsymbol{K} = [RA],$

$\hat{\boldsymbol{K}} = inflated(\boldsymbol{K}) \qquad \% \ \hat{\boldsymbol{K}}$ *is centered on a diagonal matrix*

while(not converged)

$\quad \boldsymbol{r} = [Rb - \boldsymbol{K}\ \tilde{x}]$

$\quad \boldsymbol{e} = \hat{\boldsymbol{K}} \setminus \boldsymbol{r}$

$\quad \tilde{x} = \tilde{x} + \mathrm{mid}(\boldsymbol{e})$

$\quad \boldsymbol{e} = \boldsymbol{e} - \mathrm{mid}(\boldsymbol{e})$

end

Output : $\boldsymbol{x} = \tilde{x} + \boldsymbol{e}$

Method : contractant iteration
Extra-precise relaxed interval iterative refinement

Algorithm (certifylssx)

Input : $A \in \mathbb{F}^{n\times n}, b \in \mathbb{F}^n$

$\tilde{x} = A \setminus b, \qquad R = inv(A), \qquad \mathbf{K} = [RA],$

$\hat{\mathbf{K}} = inflated(\mathbf{K})$ % $\hat{\mathbf{K}}$ *is centered on a diagonal matrix*

while(not converged)

$\quad \mathbf{r} = [Rb - \mathbf{K}\ \tilde{x}]$ % $\mathbf{r}$ *in twice the working precision*

$\quad \mathbf{e} = \hat{\mathbf{K}} \setminus \mathbf{r}$

$\quad \tilde{x} = \tilde{x} + \mathrm{mid}(\mathbf{e})$

$\quad \mathbf{e} = \mathbf{e} - \mathrm{mid}(\mathbf{e})$

end

Output : $\mathbf{x} = \tilde{x} + \mathbf{e}$

Method : contractant iteration Extra-precise relaxed interval iterative refinement

Algorithm (certifylssx)

Input : $A \in \mathbb{F}^{n \times n}, b \in \mathbb{F}^n$

$\tilde{x} = A \setminus b, \qquad R = inv(A), \qquad \boldsymbol{K} = [RA],$

$\hat{\boldsymbol{K}} = inflated(\boldsymbol{K})$ % $\hat{\boldsymbol{K}}$ *is centered on a diagonal matrix*

while(not converged)

$\quad \boldsymbol{r} = [Rb - \boldsymbol{K}\ \tilde{x}]$ % $\boldsymbol{r}$ *in twice the working precision*

$\quad \boldsymbol{e} = \hat{\boldsymbol{K}} \setminus \boldsymbol{r}$

$\quad \tilde{x} = \tilde{x} + \mathrm{mid}(\boldsymbol{e})$ % $\tilde{x}$ *in twice the working precision*

$\quad \boldsymbol{e} = \boldsymbol{e} - \mathrm{mid}(\boldsymbol{e})$

end

Output : $\boldsymbol{x} = \tilde{x} + \boldsymbol{e}$

Method : contractant iteration Extra-precise relaxed interval iterative refinement

Algorithm (certifylssx)

Input : $A \in \mathbb{F}^{n\times n}, b \in \mathbb{F}^n$

$\tilde{x} = A \setminus b, \qquad R = inv(A), \qquad \boldsymbol{K} = [RA],$

$\hat{\boldsymbol{K}} = inflated(\boldsymbol{K})$ % $\hat{\boldsymbol{K}}$ *is centered on a diagonal matrix*

while(not converged)

$\quad \boldsymbol{r} = [Rb - \boldsymbol{K}\,\tilde{x}]$ % $\boldsymbol{r}$ *in twice the working precision*

$\quad \boldsymbol{e} = \hat{\boldsymbol{K}} \setminus \boldsymbol{r}$

$\quad \tilde{x} = \tilde{x} + \mathrm{mid}(\boldsymbol{e})$ % $\tilde{x}$ *in twice the working precision*

$\quad \boldsymbol{e} = \boldsymbol{e} - \mathrm{mid}(\boldsymbol{e})$

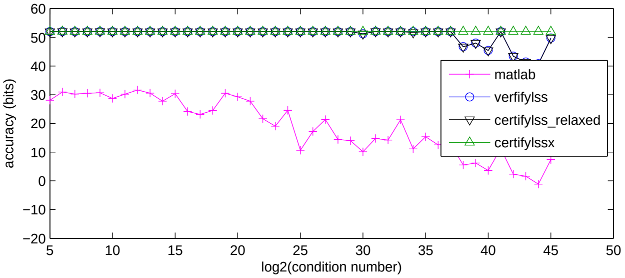
end

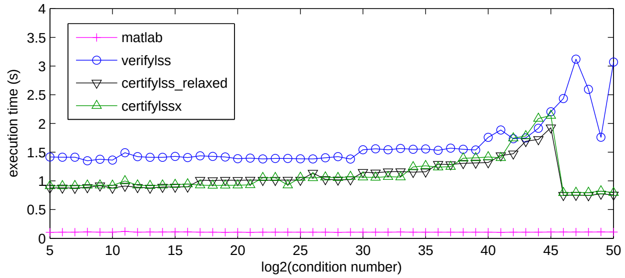
Output : $\boldsymbol{x} = \tilde{x} + \boldsymbol{e}$

Implementation : careful tuning of the precision of each variable (no doubling for $\boldsymbol{e}$: useless and costly).

Extra-precise relaxed method : Results

Dimension : 1000 $\qquad b = [1, \ldots, 1]^T$





Vérification numérique : problèmes à résoudre

Compromis temps / précision :

- temps :
  - utiliser les bibliothèques optimisées pour le calcul flottant,
  - effectuer des opérations à gros grain,
  - penser en centre-rayon ?
- précision :
  - utiliser les bibliothèques optimisées et précises pour le calcul flottant,
  - identifier les opérations sensibles et n'effectuer que celles-ci en précision augmentée,
  - penser en centre-rayon ?

Calcul ensembliste : problèmes à résoudre

De nouveaux algorithmes à inventer :

- algèbre linéaire
- optimisation globale
- résolution de contraintes (égalités et inégalités)
- intégration d'équations différentielles, ordinaires ou partielles
- intégration de systèmes hybrides
- ...

Calcul ensembliste : problèmes à résoudre

De nouveaux algorithmes à inventer :

- algèbre linéaire
- optimisation globale
- résolution de contraintes (égalités et inégalités)
- intégration d'équations différentielles, ordinaires ou partielles
- intégration de systèmes hybrides
- ...

Être plus systématique ?

Calcul ensembliste : problèmes à résoudre

De nouveaux algorithmes à inventer :

- algèbre linéaire
- optimisation globale
- résolution de contraintes (égalités et inégalités)
- intégration d'équations différentielles, ordinaires ou partielles
- intégration de systèmes hybrides
- ...

Être plus systématique ? ou plus rigoureux ?

- établir la complexité théorique de chaque algorithme, ne pas concevoir que des méthodes exactes pour des problèmes NP-durs.
- établir la précision de chaque algorithme,
- établir des benchmarks pour se comparer : jeux de tests et jeux de temps d'exécution.

Arithmétique par intervalles normalisation : The IEEE-P1788 Project

Credo : l'arithmétique par intervalles est assez mature pour que l'on puisse entreprendre en commun de figer les définitions.

Objectif : normaliser l'arithmétique par intervalles.

Création : par 15 participants à Dagstuhl, janvier 2008. Projet accepté par IEEE sous le nom de code IEEE-WG-P1788 en juin 2008.

Comment travaille IEEE-WG-P1788 ?

- L'essentiel du travail se fait par courriel, avec votes électroniques ;
- des *motions* sont proposées, soutenues, s'ensuivent 3 semaines de discussion puis 3 semaines de vote ;
- IEEE a accordé 4 ans (extensibles) ;
- une réunion en chair et en os par an – la prochaine aura lieu pendant la conférence Arith à Tübingen, fin juillet 2011 ;
- le parrainage d'IEEE comporte un rapport et une audio-conférence par trimestre.

Les motions proposées jusquà présent

1 Provisional standard notation for intervals

2-14 Levels structure for standardisation process

3 Standard is based on R not R^*

4 Restrict standard to 754 systems

5-10 Arithmetic operations and elementary functions

6 Multi- & mixed-format interval support

-15-18 Exception handling

9 Exact dot product

11-12 Reverse Arithmetic Operations

-14-20 Comparison Relations

-19-23 inf/sup and mid/rad

17 IO

21 Overlapping intervals

22 Continuity Bit

Standardized IA and IA in libraries

What is already implemented everywhere :

- work in R not R^*,
- arithmetic operations without exceptions ($+$, $-$, $\times$, $/\ldots$),
- use of outward rounding (apart from `intpakX`, `AWA` and `COSY`),
- elementary functions (at least some of them) : not controversial.

What is implemented nowhere :

- conversion between formats (apart from `MPFI`).

Selection of motions

Motions 7, 8, 15, 18 on exceptions handling. discussion of possible results and consequences. Example of the division by 0, cf. **Motion 5**.

Motion 9 on the exact dot-product. Which implementations are already compliant ?

Motions 11 and 12 on "reverse" or "inner" operators.

Motions 13 and 14 on comparisons

Exception : arguments outside the domain

How should $f(\boldsymbol{x})$ be handled when $\boldsymbol{x}$ is not included in the domain of f ? E.g. $\sqrt{[-1,2]}$?

- exit ?
- return NaI (Not an Interval) ? Ie. handle exceptional values such as NaI and infinities ?
- return the set of every possible limits $\lim_{y \to x} f(y)$ for every possible x in the domain of f (but not necessarily y) ?
- intersect $\boldsymbol{x}$ with the domain of f prior to the computation, silently ?
- intersect $\boldsymbol{x}$ with the domain of f prior to the computation and mention it

Motions 7, 8, 15 and 18 : Exceptions

Issue : How to handle exceptions efficiently.

- Typical examples :
  (a) Invalid interval constructor
  `interval(3,2)` `interval("[2.4,3;5]")`
  —interface between interval world and numbers or text strings.
  (b) Elementary function evaluated partly or wholly outside domain
  `sqrt([-1,4])` `log([-4,-1])` `[1,2]/[0,0]`
- Type (a) can simply cause nonsense if ignored.
- Type (b) are crucial for applications that depend on fixed-point theorems ; but can be ignored by others, e.g. some optimisation algorithms.

Digression about extended interval arithmetic
Division by an interval containing 0

Main concern : Newton iteration to solve $f(x) = 0$ without losing any solution.

Proposals :

- Jaulin et al. : $1/[-2, 2] = (-\infty, +\infty)$ but $[3, 4]/[0, 0] = \emptyset$;
- $[1, 2]/[0, 1] = [1, +\infty)$ since only nonnegative terms can be produced (Ratschek & Rokne 1988) ;
- $[1, 2]/[0, 1] = \{-\infty\} \cup [1, +\infty]$ (cset theory)
- $[1, 2]/[-1, 1] = (-\infty, -1] \cup [1, +\infty)$ (Ratz)

Division by 0 :

$$[a_1, a_2]/[b_1, b_2] = (-\infty, +\infty)$$
$$\text{when } a_1 < 0 < a_2 \text{ and } b_1 < 0 < b_2$$

Motions 11 and 12 : "reverse" or "inner" or "relational" operations

Example :

$$\begin{array}{rlcl} & [1,2] + x & = & [3,5] \\ \Rightarrow & x & = & [3,5] - [1,2] = [1,4] \end{array}$$

Problem : one does not have $[1,2] + x = [3,5]$, but $[1,2] + x = [1,2] + [1,4] = [2,6] \supsetneq [3,5]$.

Wanted : $x = [2,3]$ because $[1,2] + [2,3] = [3,5]$.

Inner subtraction : $[3,5] -^{-} [1,2] \equiv [2,3]$.
Operation must be total and closed.

Motions 11 and 12 : "reverse" or "inner" or "relational" operations

Another example :

$$\begin{array}{rlll} & [1,8] + x & = & [3,5] \\ \Rightarrow & x & = & [3,5] - [1,8] = [-5,4] \end{array}$$

Problem : one does not have $[1,8] + x = [3,5]$, but $[1,8] + x = [1,8] + [-5,4] = [-4,12] \supsetneq [3,5]$.

Motions 11 and 12 : "reverse" or "inner" or "relational" operations

Another example :

$$\begin{array}{rlcl} & [1,8] + x & = & [3,5] \\ \Rightarrow & x & = & [3,5] - [1,8] = [-5,4] \end{array}$$

Problem : one does not have $[1,8] + x = [3,5]$, but $[1,8] + x = [1,8] + [-5,4] = [-4,12] \supsetneq [3,5]$.
But one cannot get x : it should have a negative width.

Definition for this case : $x = [3,5] - [1,8] = [-5,4]$ because one cannot do better.

With this definition, the **inner subtraction** is total and closed.

IEEE 1788 WG : in short

Web site : http://grouper.ieee.org/groups/1788/
(or google "IEEE 1788").

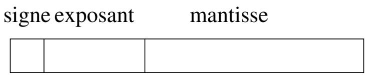
To participate :

- subscribe to the mailing list
- subscribe to the voting roster

Standard expected in December 2011. . .
I hope we will have produced at least a substantial part of it !-)

Compliant implementations :
they all should be compliant for my next Sieste.

Norme IEEE-754 : représentation d'un nombre



$$x = (-1)^{\text{signe}} \times \text{mantisse} \times 2^{\text{exposant}}$$

Codage :

- **signe** s (codé sur un bit : 0 pour positif et 1 pour négatif)
- **exposant** e entier de k chiffres compris entre $e_{\min}$ et $e_{\max}$
- **mantisse** de $n+1$ chiffres : $m = x_0.x_1x_2 \cdots x_n$
  où les x_i sont les chiffres (compris entre 0 et $\beta - 1$).

Norme IEEE-754 : formats de base

Pour des questions de précision, on exige que le premier chiffre de la mantisse soit non nul : **normalisation**.
En base 2, la normalisation revient à dire que le premier bit de la mantisse est égal à 1 ; on ne le stocke pas : **bit implicite** ou **bit caché**.

	nombre de bits			
format	total	signe	exposant	mantisse
simple	32	1	8	$23 + 1$
double	64	1	11	$52 + 1$

Norme IEEE-754 : mantisse

Pour des questions de précision, on exige que le premier chiffre de la mantisse soit non nul : **normalisation**.

La mantisse normalisée est codée sur $n+1$ bits :

$$m = 1.x_1 x_2 \cdots x_n$$

où les x_i sont des **bits** (pour *binary digits*).

Il faut alors un codage spécial pour 0.

Norme IEEE-754 : exposant

Exposant e : entier signé de k chiffres compris entre $e_{\min}$ et $e_{\max}$.

Représentation biaisée : on représente $e + b$ où b est le biais $\Rightarrow e + b \geq 0$.

Avantage : on peut faire des comparaisons entre flottants dans l'ordre lexicographique (sauf pour le signe).

format	taille k	biais b	exp. non biaisé	exp. biaisé
simple	8	127	$[-126, 127]$	$[1, 254]$
double	11	1023	$[-1022, 1023]$	$[1, 2046]$

Remarque : il reste des valeurs réservées, qui sont 0 et $2^k - 1$ en représentation biaisée.

Norme IEEE-754 : représentation de zéro

- **signe** : positif ou négatif
- **exposant** biaisé à 0
- **mantisse** ovec tous ses chiffres à 0

Remarque : on a deux représentations pour 0, selon la valeur du signe.

Norme IEEE-754 : valeurs spéciales

les infinis : $-\infty$ et $+\infty$

codés en utilisant le plus grand exposant possible et que des 0 pour la mantisse.
Le signe correspond à celui de l'infini.

Not a Number : NaN

- permet de représenter le résultat d'une opération invalide telle que $\sqrt{-2}$, $0/0$ ou $\infty - \infty$
- est codé en utilisant le plus grand exposant possible et une mantisse contenant au moins un 1 ;
- se propage dans les calculs, en contaminant les résultats.

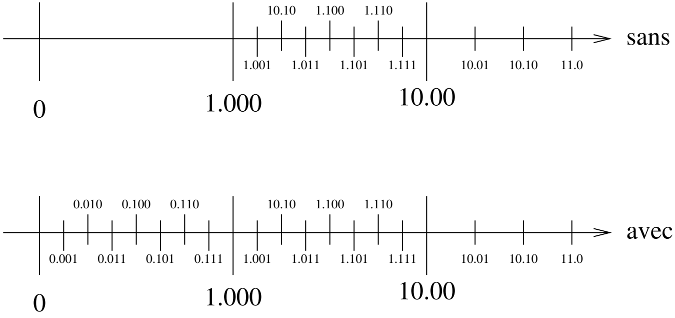
En simple précision

$-\infty$	1	11111111	00000000000000000000000	
$+\infty$	0	11111111	00000000000000000000000	
NaN	0	11111111	00100010000010000001000	(par exemple)

Norme IEEE-754 : subnormals

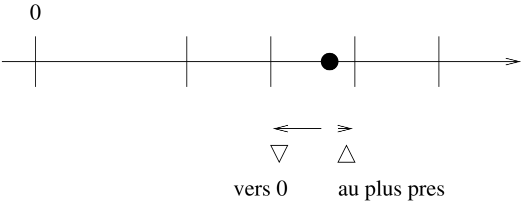
Pour uniformiser la répartition des nombres représentables autour de 0, on autorise la représentation de nombres qui s'écrivent

$$x = (-1)^{\text{signe}} \times 0.x_1 x_2 \cdots x_n \times 2^{\text{exposant}}$$



Norme IEEE-754 : modes d'arrondi

- **calculs**
  π : non représentable par un flottant
  produit de 2 nombres flottants : ne rentre pas dans un flottant
- **arrondis**



Norme IEEE-754 : modes d'arrondis

Quatre modes d'arrondi :

- **arrondi vers $+\infty$** ou par excès, noté $\triangle(x)$: retourne le plus petit nombre machine supérieur ou égal au résultat exact x ;
- **arrondi vers $-\infty$** ou par défaut noté $\bigtriangledown(x)$: retourne le plus grand nombre machine inférieur ou égal au résultat exact x ;
- **arrondi vers 0** : retourne $\triangle(x)$ pour les nombres négatifs et $\bigtriangledown(x)$ pour les nombres positifs ;
- **arrondi au plus près** : retourne le nombre machine le plus proche du résultat exact x, celui dont la mantisse se termine par 0 pour le milieu de deux nombres machine consécutifs (on parle d'arrondi pair).

Norme IEEE-754 : arrondi correct

Arrondi correct : soient a et b deux nombres machine et $\odot$ une opération parmi $\{+, -, \times, /\}$ avec o l'un des 4 arrondis prévus par la norme,
le résultat $a \odot b$ doit être $o(a \dot{} b)$ l'arrondi du résultat exact.

On a la même exigence pour la racine carrée.

Norme IEEE-754 : conversions

Les conversions normalisées sont :

- flottant vers entier
- entier vers flottant
- flottant vers entier stocké dans un flottant
- entre flottants de différents formats
- entre formats binaire et décimal.

The P1788 Project

Belief : interval arithmetic is mature enough to undergo a common definition.

Goal : standardize interval arithmetic.

Creation : Initiated by 15 attenders at Dagstuhl, Jan 2008. Project authorised as IEEE-WG-P1788, Jun 2008.

How P1788's work is done

- The bulk of work is carried out by email, with electronic voting.
- Motions are proposed, seconded ; three weeks discussion period ; three weeks voting period.
- IEEE has given us a four year deadline (not cast in stone).
- One "in person" meeting per year is planned—next one will be October 1st, afternoon, in Lyon, France, right after SCAN 2010.
- IEEE auspices : 1 report + 1 teleconference quarterly

IEEE-1788 WG : some facts

Since October 2008 : very active mailing list
over 140 participants, over 20 nationalities, over 2800 messages

Work already done :
adoption of officers, of procedures and policy
roster of voting members : 83 members, 18 nationalities
17 motions handled
currently : 1 motion under vote, 4 motions in discussion.

Selection of libraries

- **General-purpose libraries** : `IntLib` used in `GlobSol`, `Profil-BIAS`, `Sun Studio-C` compiler, `C-XSC`, `fi_lib`, `IntpakX`, `Mathemagix`, `MPFI` ;

Selection of libraries

- **General-purpose libraries** : `IntLib` used in `GlobSol`, `Profil-BIAS`, `Sun Studio-C` compiler, `C-XSC`, `fi_lib`, `IntpakX`, `Mathemagix`, `MPFI` ;
- **Libraries for linear algebra** : `C-XSC`, `IntLab`, `Profil-BIAS` ;

Selection of libraries

- **General-purpose libraries :** `IntLib` used in `GlobSol`, `Profil-BIAS`, `Sun Studio-C` compiler, `C-XSC`, `fi_lib`, `IntpakX`, `Mathemagix`, `MPFI` ;
- **Libraries for linear algebra :** `C-XSC`, `IntLab`, `Profil-BIAS` ;
- **Libraries for the integration of ODEs :** `AWA`, `CAPD`, `COSY`, `VSPODE`, (`VNODE` : based on `fi_lib` or `Profil-BIAS`) ;

Selection of libraries

- **General-purpose libraries :** IntLib used in GlobSol, Profil-BIAS, Sun Studio-C compiler, C-XSC, fi_lib, IntpakX, Mathemagix, MPFI ;
- **Libraries for linear algebra :** C-XSC, IntLab, Profil-BIAS ;
- **Libraries for the integration of ODEs :** AWA, CAPD, COSY, VSPODE, (VNODE : based on fi_lib or Profil-BIAS) ;
- **Libraries for constraints solving :** Gaol, IBEX-Quimper, Realpaver, (Alias : based on Profil-BIAS) ;

Selection of libraries

- **General-purpose libraries :** IntLib used in GlobSol, Profil-BIAS, Sun Studio-C compiler, C-XSC, fi_lib, IntpakX, Mathemagix, MPFI ;
- **Libraries for linear algebra :** C-XSC, IntLab, Profil-BIAS ;
- **Libraries for the integration of ODEs :** AWA, CAPD, COSY, VSPODE, (VNODE : based on fi_lib or Profil-BIAS) ;
- **Libraries for constraints solving :** Gaol, IBEX-Quimper, Realpaver, (Alias : based on Profil-BIAS) ;
- **Libraries for unconstrained or constrained global optimization :** COSY-GO, GlobSol, (Baron : NEOS, Coconut : based on fi_lib and Jail, GloptLab : on IntLab) ;

Selection of libraries

- **General-purpose libraries :** IntLib used in GlobSol, Profil-BIAS, Sun Studio-C compiler, C-XSC, fi_lib, IntpakX, Mathemagix, MPFI ;
- **Libraries for linear algebra :** C-XSC, IntLab, Profil-BIAS ;
- **Libraries for the integration of ODEs :** AWA, CAPD, COSY, VSPODE, (VNODE : based on fi_lib or Profil-BIAS) ;
- **Libraries for constraints solving :** Gaol, IBEX-Quimper, Realpaver, (Alias : based on Profil-BIAS) ;
- **Libraries for unconstrained or constrained global optimization :** COSY-GO, GlobSol, (Baron : NEOS, Coconut : based on fi_lib and Jail, GloptLab : on IntLab) ;
- **Libraries using variants of interval arithmetic :** Chebyshev models in ChebModels (based on IntpakX), Taylor models in COSY and C-XSC, affine arithmetic in

Exception : arguments outside the domain

How should $f(\boldsymbol{x})$ be handled when $\boldsymbol{x}$ is not included in the domain of f ? E.g. $\sqrt{[-1,2]}$?

- exit ?
- return NaI (Not an Interval) ? Ie. handle exceptional values such as NaI and infinities ?
- return the set of every possible limits $\lim_{y \to x} f(y)$ for every possible x in the domain of f (but not necessarily y) ?
- intersect $\boldsymbol{x}$ with the domain of f prior to the computation, silently ?
- intersect $\boldsymbol{x}$ with the domain of f prior to the computation and mention it

Exception : arguments outside the domain

How should $f(\boldsymbol{x})$ be handled when $\boldsymbol{x}$ is not included in the domain of f ? E.g. $\sqrt{[-1, 2]}$?

- exit ? `IntLib, Profil-BIAS...`
- return NaI (Not an Interval) ? `MPFI`
- return the set of every possible limits $\lim_{y \to x} f(y)$ for every possible x in the domain of f (but not necessarily y) ? `IntLib, Sun Studio-C...`
- intersect $\boldsymbol{x}$ with the domain of f prior to the computation, silently ? `Profil-BIAS...`
- intersect $\boldsymbol{x}$ with the domain of f prior to the computation and mention it : `C-XSC, fi_lib, Profil-BIAS...`

Every solution exists in the libraries !

Motions 7, 8, 15 and 18 : Exceptions

Issue : How to handle exceptions efficiently.

- Typical examples :
  (a) Invalid interval constructor
  `interval(3,2)` `interval("[2.4,3;5]")`
  —interface between interval world and numbers or text strings.
  (b) Elementary function evaluated partly or wholly outside domain
  `sqrt([-1,4])` `log([-4,-1])` `[1,2]/[0,0]`
- Type (a) can simply cause nonsense if ignored.
- Type (b) are crucial for applications that depend on fixed-point theorems ; but can be ignored by others, e.g. some optimisation algorithms.

Motions 7 and 8 : Exceptions, cont.

What to do ? A complicated issue.

- Risk that (Level 3) code to handle exceptions will slow down interval applications that don't need it.
- One approach to type (a) is to define an NaI "Not an Interval" datum at level 2, encoded at level 3 within the two FP numbers that represent an interval.

Motion 8 : Exceptions by Decorations

- Alternative (Motion 8) : An extra tag or decoration field (1 byte ?) in level 3 representation.
- Divided into subfields that record different kinds of exceptional behaviour.
- Decoration is optional, can be added and dropped.
  – To compute at full speed, use "bare" intervals and corresponding "bare" elementary function library.
  – "Decorated" library records exceptions separate from numbers, hence code has fewer IFs & runs fast too. (We hope !)

Motions 8, 15 and 18 : Decoration issues

Decorations look promising but many Qs exist :

- Bare (double) interval is 16-byte object. Decoration increases this. Values are trits or tetrits.Can compilers efficiently allocate memory for large arrays of such objects ?
- Some proposed decoration-subfields record events in the past ; others are properties of the current interval. Can semantic inconsistencies arise ?
- Can decoration semantics be specified at Level 2 . . .
- . . . such that correctness of code can be proven . . .
- . . . and K.I.S.S. is preserved ?

Much work on exceptions remains : list, propagation rules.

Remark : arguments outside the domain

Problematic example (Rump, Dagstuhl seminar 09471, 2009).

$$\begin{aligned} f(x) &= |x-1| \\ g(x) &= (\sqrt{x-1})^2 \end{aligned}$$ I know, it is not the best way of writing it...

What happens if $\boldsymbol{x} = [0, 1]$?
With the adopted definitions of operations,

$$\begin{aligned} f(\boldsymbol{x}) &= [0, 1] \\ g(\boldsymbol{x}) &= [0] \end{aligned}$$

Without exception handling, **the *Thou shalt not lie* principle is not valid.**
One has to check whether there has been a *possibly undefined* operation...

Remark : arguments outside the domain

Problematic example (Rump, Dagstuhl seminar 09471, 2009).

$$\begin{array}{rcl} f(x) & = & |x-1| \\ g(x) & = & (\sqrt{x-1})^2 \end{array}$$

I know, it is not the best way of writing it...

What happens if $\boldsymbol{x} = [0, 1]$?
With the adopted definitions of operations,

$$\begin{array}{rcl} f(\boldsymbol{x}) & = & [0, 1] \\ g(\boldsymbol{x}) & = & [0] \end{array}$$

Without exception handling, **the *Thou shalt not lie* principle is not valid.**
One has to check whether there has been a *possibly undefined* operation... Unexperienced programmers will not do it.

Digression about extended interval arithmetic
Division by an interval containing 0

Main concern : Newton iteration to solve $f(x) = 0$ without losing any solution.

Proposals :

- Jaulin et al. : $1/[-2, 2] = (-\infty, +\infty)$ but $[3, 4]/[0, 0] = \emptyset$;
- $[1, 2]/[0, 1] = [1, +\infty)$ since only nonnegative terms can be produced (Ratschek & Rokne 1988) ;
- $[1, 2]/[0, 1] = \{-\infty\} \cup [1, +\infty]$ (cset theory)
- $[1, 2]/[-1, 1] = (-\infty, -1] \cup [1, +\infty)$ (Ratz)

Division by 0 :

$$[a_1, a_2]/[b_1, b_2] = (-\infty, +\infty)$$
$$\text{when } a_1 < 0 < a_2 \text{ and } b_1 < 0 < b_2$$

Motions 11 and 12 : "reverse" or "inner" or "relational" operations

Example :

$$\begin{array}{rlcl} & [1,2] + \boldsymbol{x} & = & [3,5] \\ \Rightarrow & \boldsymbol{x} & = & [3,5] - [1,2] = [1,4] \end{array}$$

Problem : one does not have $[1,2] + \boldsymbol{x} = [3,5]$, but $[1,2] + \boldsymbol{x} = [1,2] + [1,4] = [2,6] \supsetneq [3,5]$.

Wanted : $\boldsymbol{x} = [2,3]$ because $[1,2] + [2,3] = [3,5]$.

Inner subtraction : $[3,5] -^{-} [1,2] \equiv [2,3]$.
Operation must be total and closed.

Motions 11 and 12 : "reverse" or "inner" or "relational" operations

Another example :

$$\begin{array}{rlcl} & [1,8] + \boldsymbol{x} & = & [3,5] \\ \Rightarrow & \boldsymbol{x} & = & [3,5] - [1,8] = [-5,4] \end{array}$$

Problem : one does not have $[1,8] + \boldsymbol{x} = [3,5]$, but $[1,8] + \boldsymbol{x} = [1,8] + [-5,4] = [-4,12] \supsetneq [3,5]$.

Motions 11 and 12 : "reverse" or "inner" or "relational" operations

Another example :

$$\begin{array}{rlcl} & [1,8] + x & = & [3,5] \\ \Rightarrow & x & = & [3,5] - [1,8] = [-5,4] \end{array}$$

Problem : one does not have $[1,8] + x = [3,5]$, but $[1,8] + x = [1,8] + [-5,4] = [-4,12] \supsetneq [3,5]$.
But one cannot get x : it should have a negative width.

Definition for this case : $x = [3,5] - [1,8] = [-5,4]$ because one cannot do better.

With this definition, the **inner subtraction** is total and closed.

Motions 11 and 12 : "reverse" or "inner" or "relational" operations

Libraries that implement it :

- libraries for constraint programming (constrained optimization, constraints solving) : `Coconut`, `GlobtLab`, `Gaol`, `RealPaver` :
- library for modal arithmetic (Nate Hayes)
- others : `AWA`

Variants of interval arithmetic : Taylor or Chebyshev models, affine arithmetic

Taylor models : represent a function by a polynomial approximation + an interval remainder.
Available in `C-XSC`, `IntLib`, `IntLab`, `AWA`, `COSY`, `VSPODE`, `GlobSol`

Chebyshev or Bernstein models : represent a function by a polynomial approximation in the Chebyshev or Bernstein basis + an interval remainder. Available in `ChebModels`, `Zumkeller's`

Affine arithmetic : represent a function by a linear approximation, where first-order terms represent variables and errors. Available in `Fluctuat...`

Libraries that **evaluate in the small**, for ODE integration or numerical certification.

IEEE 1788 WG : in short

Web site : http://grouper.ieee.org/groups/1788/
(or google "IEEE 1788").

To participate :

- subscribe to the mailing list
- subscribe to the voting roster

Standard expected in December 2011...
I hope we will have produced at least a substantial part of it !-)

Compliant implementations :
they all should be compliant for my next Sieste.

Appendix

Motions so far

#1. (Pryce/Kreinovich) Provisional standard notation for intervals
#2. (Pryce/Corliss) Levels structure for standardisation process
#3. (W. von Gudenberg/Hack) Standard is based on $\boldsymbol{R}$ not $\boldsymbol{R}^*$
#4. (Pryce/Zuras) Restrict standard to 754 systems (withdrawn)
#5. (Kulisch/Einarsson) Tables for arithmetic operations
#6. (Pryce/Zuras) Multi- & mixed-format interval support
#7. (W. von Gudenberg/Cunha) Have just one NaI (withdrawn)
#8. (Hayes/Kreinovich) Decorations : a way to handle exceptions
#9. (Kulisch/Einarsson) Exact dot product
#10. (W. von G./Kreinovich) Supported elementary functions
#11. (Nehmeier/Munoz) Basic Reverse Arithmetic Operations (withdrawn)

Motions so far

#13. (Kulisch/Pryce) Comparison Relations (discussion)

#14. (Pryce/Einarsson) Voting on section 6.1 - 6.2 Wording (waiting)

#15. (Zuras/Hu) Definition of Decorations (withdrawn)

#16. (Neumaier/Zuras/Kreinovich) Status of inf/sup and mid/rad

#17. (Pryce/Zuras) IO

#18. (Hayes/Kreinovich) The "domain" tetrit and bool_set semantics

#19. (Pryce/Zuras) Implicit formats (voting)

#20. (Lohez/Zuras) Framework for comparisons (discussion)

#21. (W. von G./Hayes) Interval overlapping (discussion)

#22. (Pryce/Zuras) Continuity Bit (discussion)

#23. (Neumaier/Hayes) NoMidRad (discussion)

To join IEEE WG P1788

Send to : Chair Nathalie.Revol@ens-lyon.fr or to Vice-Chair R. Baker Kearfott rbk@louisiana.edu an e-mail with

- your first name and name
- your affiliation
- your complete adress
- your e-mail address
- whether you plan to just subscribe to the mailing list or also to be an active (voting) member of the working group.

Motion 1 : Standardized notations

Cf. "Standardized notation in interval analysis" by R.B. Kearfott, M.T. Nakao, A. Neumaier, S.M. Rump, S.P. Shary, and P. van Hentenryck,
http://www.mat.univie.ac.at/~neum/papers.html
(open to amendment after sufficient experience of using it)

A *box* of dimension n is a pair $\boldsymbol{x} = [\underline{x}, \overline{x}]$ consisting of two real column vectors $\underline{x}$ and $\overline{x}$ of length n with $\underline{x} \leq \overline{x}$. The set of all boxes of dimension n is denoted by $\mathbb{IR}^n$.

$f([-1,1])$ denotes the image of $[-1,1]$ under f ; f$([-1,1])$ specifies it as the result of applying the operations in f to the interval $[-1,1]$ in its intrinsic arithmetic.

Motion 2 : Levels Structure of P1788

Issue : manage complexity.
Following good software engineering practice, we decided to arrange the standard in 4 levels, or layers :

- Level 1 : mathematical level
- Level 2 : datum level : *IF* the set of machine intervals
- Level 3 : representation level ($\emptyset$, NaI,. . .)
- Level 4 : bit strings

Motion 2 : Levels structure of P1788

Level 1. Mathematical objects. Intervals are subsets of $\boldsymbol{R}$. Operation on intervals $\boldsymbol{x}, \boldsymbol{y}$ defined by $\boldsymbol{x} \bullet \boldsymbol{y} = \text{hull}\{\, x \bullet y \mid x \in \boldsymbol{x},\ y \in \boldsymbol{y},\ x \bullet y \text{ exists} \,\}$.

Level 2. Finite set of *machine intervals* (interval datums). $\boldsymbol{x} \bullet \boldsymbol{y}$ defined as "machine hull" of ($\boldsymbol{x} \bullet \boldsymbol{y}$ at level 1). *Decorations* for exception handling—see later—exist at this level.

Level 3. Representation. Exposes the variables in terms of which interval operations are implemented.

Level 4. Encoding. The level of bit strings and memory locations.

Motion 2 : Levels structure of P1788

Relationships between different speciÂ£cation levels : interval version

Level 1	Number system $\mathbb{R}$. Set $\overline{\mathbb{IR}}$ of allowed intervals over $\mathbb{R}$. Principles of how $+ - * /$ and standard functions are extended to intervals.	Mathematical Model level. No NaN or NaI at this level.
many-to-one	$\downarrow$ interval hull identity map, except NaI $\uparrow$	one-to-one
Level 2	The set $\mathbb{F}$ of "machine intervals" in $\overline{\mathbb{IR}} \cup \{$NaI, if used $\}$	Interval data level.
one-to-many ?	$\downarrow$ representation specification $\uparrow$	many-to-one ?
Level 3	Representation of nonempty $[\underline{x}, \bar{x}]$ as two FP numbers $\underline{x}$, $\bar{x}$ or alternative. Representation of $\emptyset$ and NaI.	Representations of interval data.
one-to-many ?	$\downarrow$ encoding speciÂ£cation $\uparrow$	many-to-one ?
Level 4	0111000 . . .	Bit strings.

Motion 3 : Intervals are sets of reals

Issue : What is an interval, anyway ?

Intervals are connected closed sets of reals
(and not of extended reals, i.e. infinites are not included)
(no reversed interval either).

They comprise the empty set.

Motion 3 : Consequences

This **excludes** some alternatives :

- Nonstandard intervals (Kaucher/modal) : intervals are not sets, but formal objects, ordered pairs $[\underline{x}, \overline{x}]$ of real numbers, allowing $\underline{x} > \overline{x}$.
  Nonstandard intervals *can* be supported as an add-on. . .
- Interval operations are defined purely algebraically (no limits) :
  containment set (cset) theory, where intervals are subsets of extended reals R^*, and operations defined by limits are quite incompatible with the standard.
- Support for open and half-open intervals.
- Complex intervals.