Algorithme de Horner compensé en précision finie et applications

Stef Graillat

LIP6/PEQUAN - Université Pierre et Marie Curie (Paris 6)

Séminaire SPIRAl/SALSA

1 juin 2007, Paris

Plan de l'exposé

Plan de l'exposé

Motivations générales

Proposer des algorithmes et des logiciels :

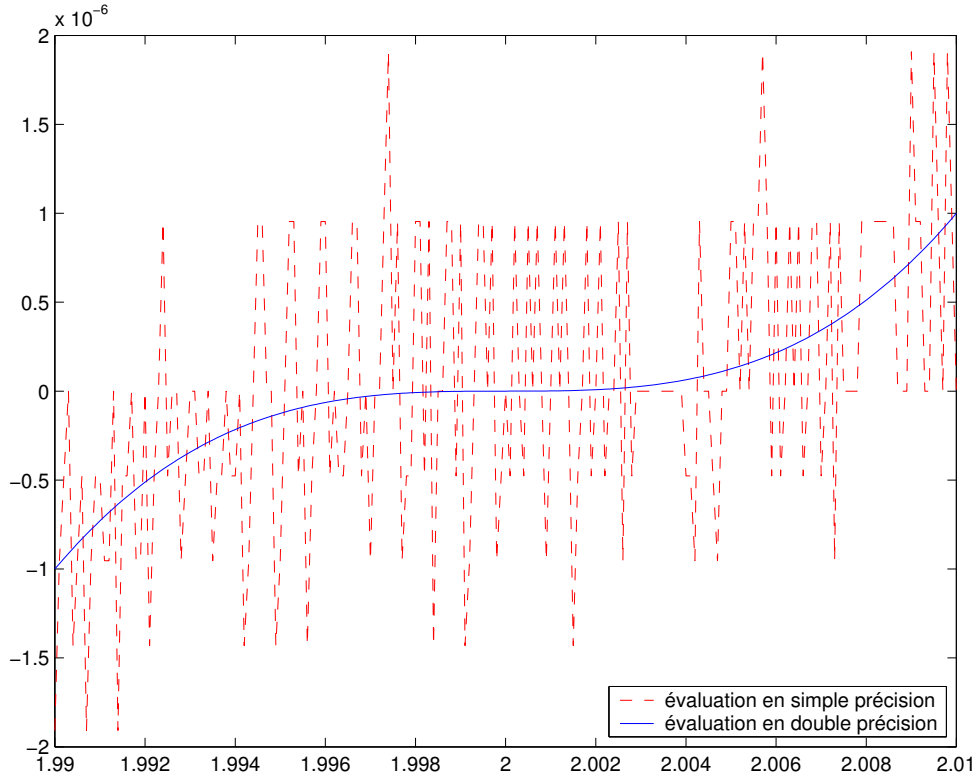
- plus précis que ceux utilisant les précisions de la norme IEEE 754
  ▷ on veut que la précision vérifie une «version améliorée» de l'estimation empirique classique
- efficace en terme de performance sans sacrifier la portabilité
  ▷ on utilise seulement la précision IEEE 754 simple ou double
- avec une borne d'erreur pour contrôler la précision du résultat
  ▷ borne d'erreur dynamique et certifiée calculable en précision finie

Un exemple : le schéma de Horner pour l'évaluation polynomiale
$\longrightarrow$ le schéma de Horner compensé[1]

[1] SG, N. Louvet, Ph. Langlois. Compensated Horner Scheme. Research Report, 2005

Imprécision dans l'évaluation polynomiale

Évaluation du polynôme $p(x) = (x-2)^3 = x^3 - 6x^2 + 12x - 8$ pour environ 200 points au voisinage de $x = 2$ en simple et double précision



Problématique en précision finie (1/2)

But : Résoudre des problèmes numériques en étant précis et fiable

- Comprendre l'influence de la précision finie sur la qualité numérique du logiciel scientifique pour contrôler et limiter ses effets néfastes
  - résultat imprécis ;
  - instabilité numérique.
- Améliorer la précision des résultats

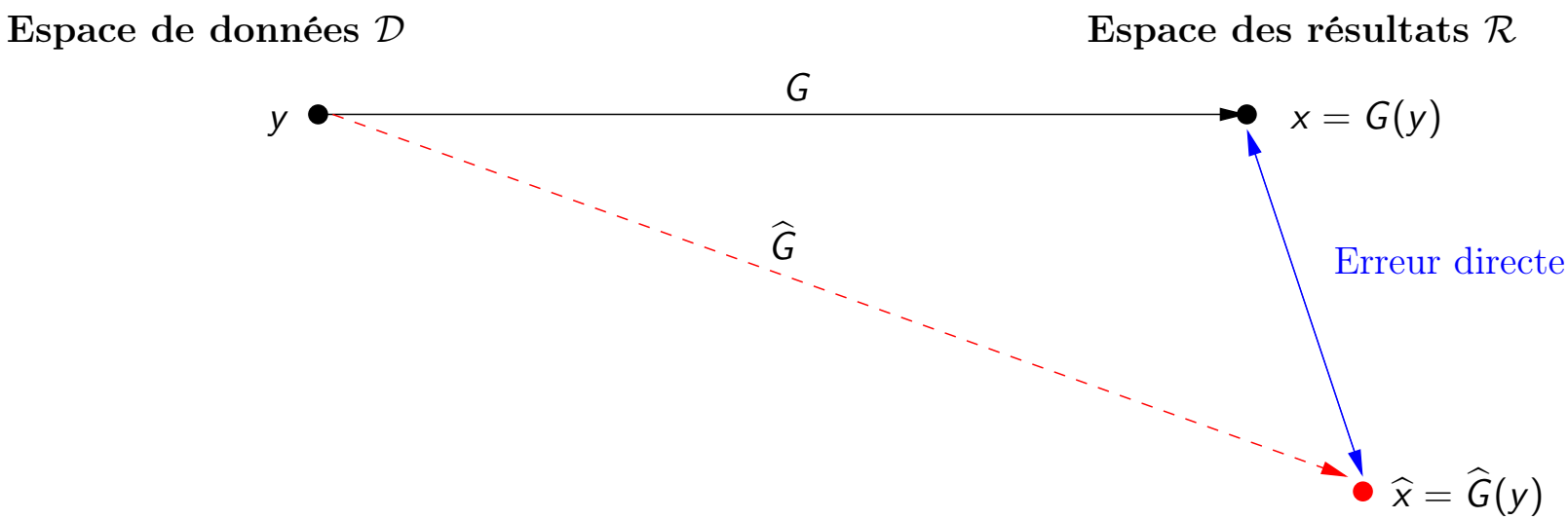
Comment être plus précis à faible coût

Problématique en précision finie (2/2)

Comprendre l'influence de la précision finie sur la qualité numérique du logiciel scientifique pour contrôler et limiter ses effets néfastes :

- Contrôler les effets de la précision finie :
  - Comment mesurer la difficulté de résolution d'un problème ?
  - Comment apprécier la fiabilité de l'algorithme de résolution ?
  - Comment estimer la précision de la solution calculée ?
- Limiter les effets de la précision finie
  - Comment améliorer la précision de la solution ?

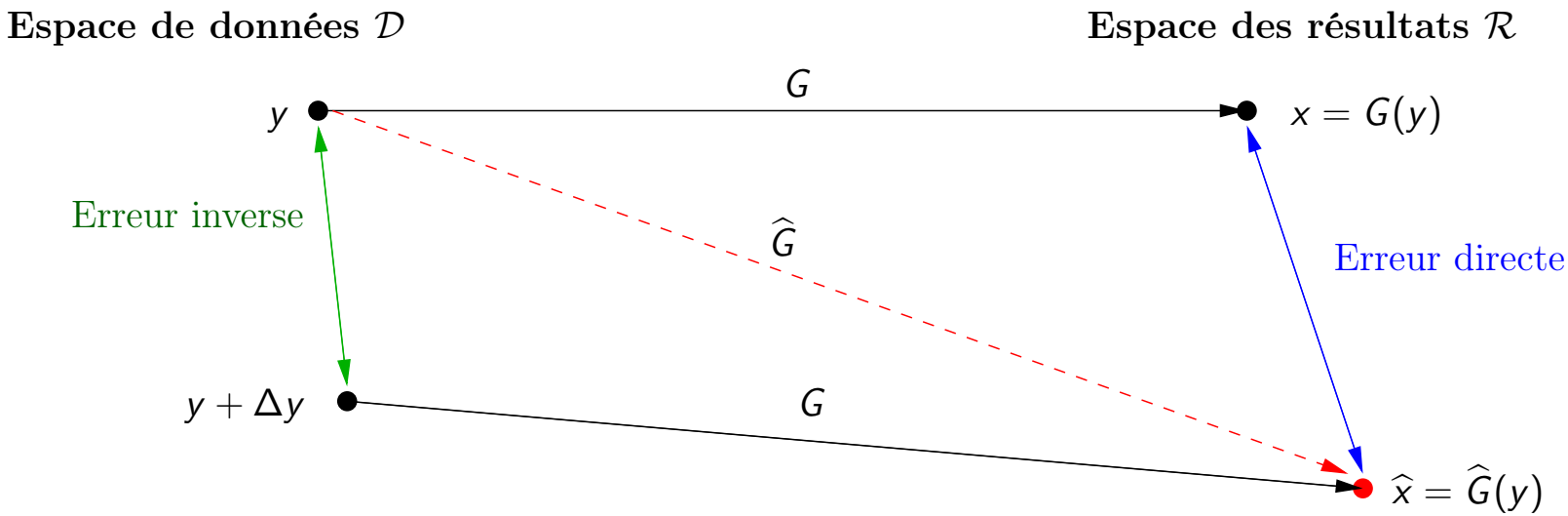
Analyse d'erreur



- Analyse directe
- Analyse inverse

  consiste à identifier $\widehat{x}$ à la solution d'un problème perturbé : $\widehat{x} = G(y + \Delta y)$.

Analyse d'erreur



- Analyse directe
- Analyse inverse
  consiste à identifier $\widehat{x}$ à la solution d'un problème perturbé :
  $\widehat{x} = G(y + \Delta y)$.

Intérêt de l'analyse inverse

- **Comment estimer la précision de la solution calculée ?**
  Au premier ordre, on a l'estimation empirique :

  $$\text{erreur directe} \lesssim \text{conditionnement} \times \text{erreur inverse.}$$

- **Comment mesurer la difficulté de résolution d'un problème ?**
  Le conditionnement caractérise la sensibilité de la solution d'un problème à des perturbations sur les données.

  $$\text{Conditionnement} : K(P, y) := \lim_{\varepsilon \to 0} \sup_{\Delta y \in \mathcal{P}(\varepsilon)} \left\{ \frac{\|\Delta x\|_{\mathcal{R}}}{\|\Delta y\|_{\mathcal{D}}} \right\}$$

- **Comment apprécier la fiabilité de l'algorithme de résolution ?**
  L'erreur inverse mesure la distance entre le problème que l'on a effectivement résolu et le problème initial.

  $$\text{Erreur inverse} : \eta(\widehat{x}) = \min_{\Delta y \in \mathcal{D}} \{ \|\Delta y\|_{\mathcal{D}} : \widehat{x} = G(y + \Delta y) \}$$

Intérêt de l'analyse inverse

- **Comment estimer la précision de la solution calculée ?**
  Au premier ordre, on a l'estimation empirique :

  $$\text{erreur directe} \lesssim \text{conditionnement} \times \text{erreur inverse.}$$

- **Comment mesurer la difficulté de résolution d'un problème ?**
  Le conditionnement caractérise la sensibilité de la solution d'un problème à des perturbations sur les données.

  $$\text{Conditionnement} : K(P, y) := \lim_{\varepsilon \to 0} \sup_{\Delta y \in \mathcal{P}(\varepsilon)} \left\{ \frac{\|\Delta x\|_{\mathcal{R}}}{\|\Delta y\|_{\mathcal{D}}} \right\}$$

- **Comment apprécier la fiabilité de l'algorithme de résolution ?**
  L'erreur inverse mesure la distance entre le problème que l'on a effectivement résolu et le problème initial.

  $$\text{Erreur inverse} : \eta(\widehat{x}) = \min_{\Delta y \in \mathcal{D}} \{ \|\Delta y\|_{\mathcal{D}} : \widehat{x} = G(y + \Delta y) \}$$

Intérêt de l'analyse inverse

- **Comment estimer la précision de la solution calculée ?**
  Au premier ordre, on a l'estimation empirique :

  $$\text{erreur directe} \lesssim \text{conditionnement} \times \text{erreur inverse.}$$

- **Comment mesurer la difficulté de résolution d'un problème ?**
  Le conditionnement caractérise la sensibilité de la solution d'un problème à des perturbations sur les données.

  $$\text{Conditionnement} : K(P, y) := \lim_{\varepsilon \to 0} \sup_{\Delta y \in \mathcal{P}(\varepsilon)} \left\{ \frac{\|\Delta x\|_{\mathcal{R}}}{\|\Delta y\|_{\mathcal{D}}} \right\}$$

- **Comment apprécier la fiabilité de l'algorithme de résolution ?**
  L'erreur inverse mesure la distance entre le problème que l'on a effectivement résolu et le problème initial.

  $$\text{Erreur inverse} : \eta(\widehat{x}) = \min_{\Delta y \in \mathcal{D}} \{ \|\Delta y\|_{\mathcal{D}} : \widehat{x} = G(y + \Delta y) \}$$

Plan de l'exposé

Nombres à virgule flottante

Un nombre flottant normalisé $x \in \mathbb{F}$ est un nombre qui s'écrit sous la forme

$$x = \pm \underbrace{x_0.x_1 \ldots x_{p-1}}_{\textit{mantisse}} \times b^e, \quad 0 \leq x_i \leq b-1, \quad x_0 \neq 0$$

b : la base, p : précision, e : exposant vérifiant $e_{\min} \leq e \leq e_{\max}$

Précision machine $\epsilon = b^{1-p}$, $|1^+ - 1| = \epsilon$

Approximation de $\mathbb{R}$ par $\mathbb{F}$, arrondi $\mathrm{fl} : \mathbb{R} \longrightarrow \mathbb{F}$
Soit $x \in \mathbb{R}$ alors

$$\mathrm{fl}(x) = x(1+\delta), \quad |\delta| \leq \mathbf{u}.$$

L'unité d'arrondi $\mathbf{u}$ vaut $\mathbf{u} = \epsilon/2$ pour l'arrondi au plus près.

Modèle standard de l'arithmétique à virgule flottante

Soient $x, y \in \mathbb{F}$,

$$\mathrm{fl}(x \circ y) = (x \circ y)(1+\delta), \quad |\delta| \leq \mathbf{u}, \quad \circ \in \{+, -, \cdot, /\}$$

Norme IEEE 754 (1985)

Type	Taille	Mantisse	Exposant	Unité d'arrondi	Intervalle
Simple	32 bits	23+1 bits	8 bits	$\mathbf{u} = 2^{-24} \approx 5,96 \times 10^{-8}$	$\approx 10^{\pm 38}$
Double	64 bits	52+1 bits	11 bits	$\mathbf{u} = 2^{-53} \approx 1,11 \times 10^{-16}$	$\approx 10^{\pm 308}$

Pour une évaluation plus précise

- Évaluation de $p(x)$ plus précise : le schéma de Horner compensé et l'estimation empirique compensée
- Une borne améliorée et certifiée de l'erreur
- Les résultats théoriques et expérimentaux montrent que
  - précision : identique à celle obtenue si les calculs avaient été effectués avec deux fois la précision courante,
  - vitesse : deux fois plus rapide que l'implémentation double-double de référence

Plus de précision, comment ?

Augmenter la précision interne des calculs :

- de façon matérielle
  - précision étendue sur l'architecture x86
- de façon logicielle
  - expansions de longueur finie : double-double (Briggs, Bailey, Hida, Li), quad-double (Bailey, Hida, Li)
  - expansions de longueur variable : Priest, Shewchuk
  - précision arbitraire : MP, MPFUN/ARPREC, MPFR

Corriger les erreurs d'arrondis :

- sommation compensée (Kahan,1965) et doublement compensée (Priest,1991), etc.
- sommation et produit scalaire : Ogita, Rump et Oishi (2005)
  $\longrightarrow$ résultat avec une précision identique à celle obtenue si on faisait les calculs avec 2 fois la précision courante

Plus de précision, comment ?

Augmenter la précision interne des calculs :

- de façon matérielle
  - précision étendue sur l'architecture x86
- de façon logicielle
  - expansions de longueur finie : double-double (Briggs, Bailey, Hida, Li), quad-double (Bailey, Hida, Li)
  - expansions de longueur variable : Priest, Shewchuk
  - précision arbitraire : MP, MPFUN/ARPREC, MPFR

Corriger les erreurs d'arrondis :

- sommation compensée (Kahan,1965) et doublement compensée (Priest,1991), etc.
- sommation et produit scalaire : Ogita, Rump et Oishi (2005)
  $\longrightarrow$ résultat avec une précision identique à celle obtenue si on faisait les calculs avec 2 fois la précision courante

À précision courante ...

Estimation empirique pour les algorithmes inverses-stables :

$$\boxed{\text{précision du résultat} \approx \text{conditionnement} \times \text{précision de calcul}}$$

1. précision IEEE-754 : double ($\mathbf{u} = 2^{-53} \approx 10^{-16}$)
2. Conditionnement pour l'évaluation de $p(x) = \sum_{i=0}^{n} a_i x^i$:

$$\text{cond}(p, x) = \frac{\sum_{i=0}^{n} |a_i||x|^i}{|\sum_{i=0}^{n} a_i x^i|} = \frac{\widetilde{p}(|x|)}{|p(x)|}, \text{ toujours } \geq 1.$$

3. Précision de la solution $\widehat{p}(x)$:

$$\boxed{\frac{|p(x) - \widehat{p}(x)|}{|p(x)|} \leq \alpha(n) \times \text{cond}(p, x) \times \mathbf{u}}$$

avec $\alpha(n) \approx 2n$

Que signifie doubler la précision courante ?

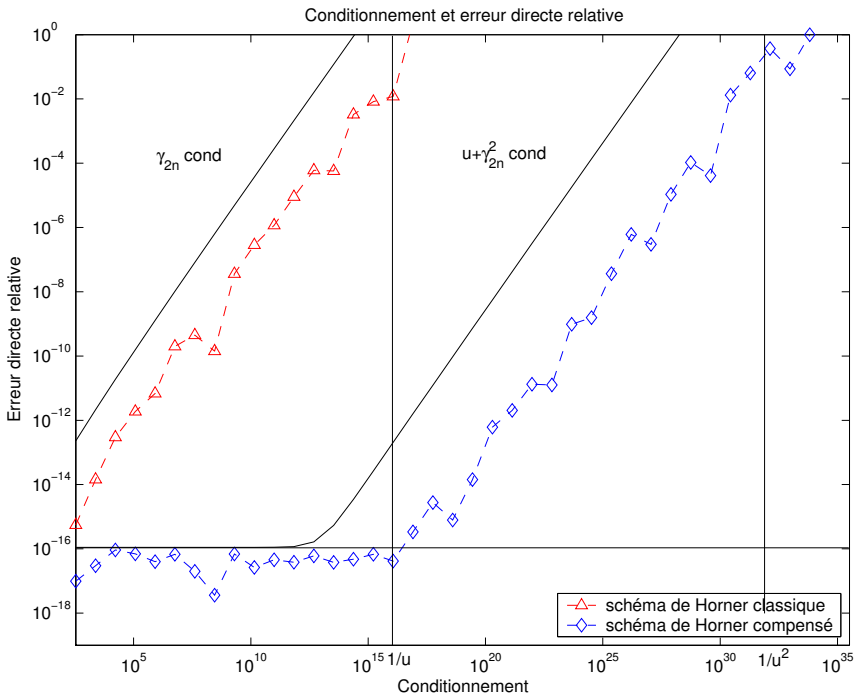
Estimation empirique compensée :

$$\text{précision du résultat} \lesssim \text{précision} + \text{conditionnement} \times \text{précision}^2$$

Trois régimes pour la précision de l'évaluation de $\widehat{p}(x)$:

1) conditionnement $\leq 1/\mathbf{u}$: la précision de $\widehat{p}(x)$ est optimale

$$\frac{|\widehat{p}(x)-p(x)|}{|p(x)|} \approx \mathbf{u}$$



Que signifie doubler la précision courante ?

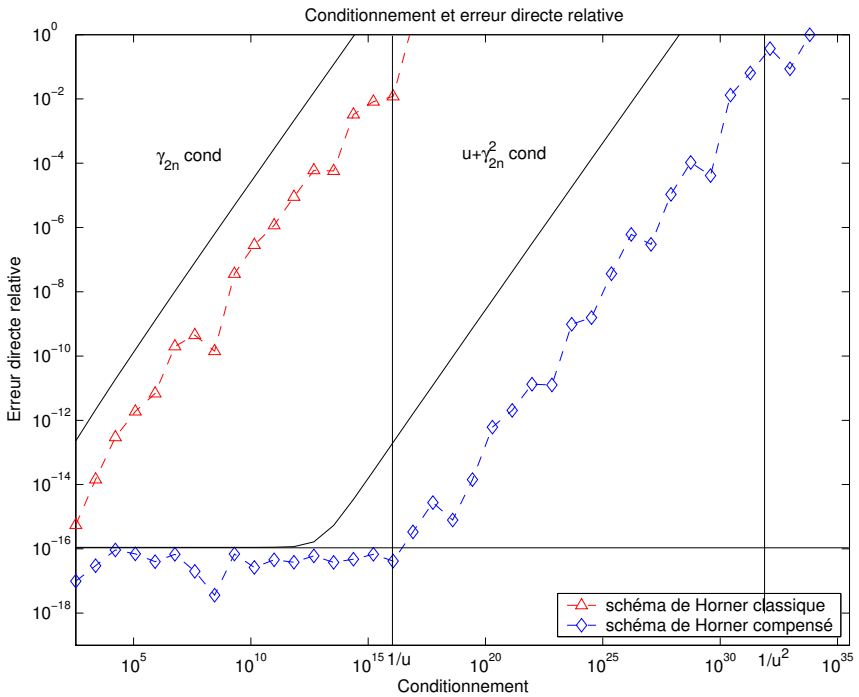
Estimation empirique compensée :

précision du résultat $\lesssim$ précision + conditionnement $\times$ précision^2

Trois régimes pour la précision de l'évaluation de $\widehat{p}(x)$:

2) $1/\mathbf{u} \leq$ conditionnement $\leq 1/\mathbf{u}^2$: la précision de $\widehat{p}(x)$

$$\frac{|\widehat{p}(x)-p(x)|}{|p(x)|} \approx \text{cond} \times \mathbf{u}^2$$



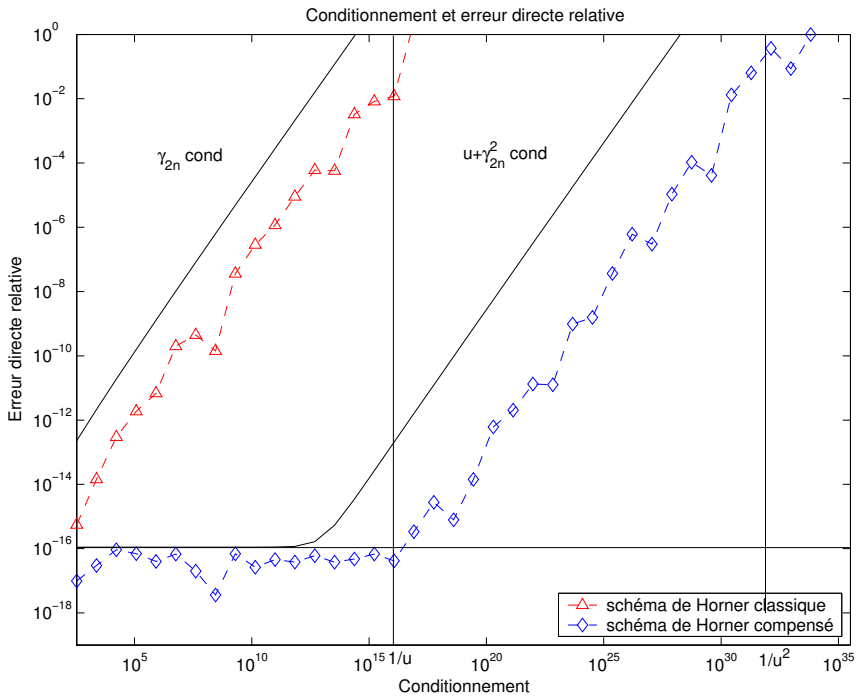
Que signifie doubler la précision courante ?

Estimation empirique compensée :

$$\text{précision du résultat} \lesssim \text{précision} + \text{conditionnement} \times \text{précision}^2$$

Trois régimes pour la précision de l'évaluation de $\widehat{p}(x)$:

3) perte totale de précision quand conditionnement $> 1/\mathbf{u}^2$.



Algorithme de Horner classique

Algorithme 1 (Algorithme de Horner classique)

function $\texttt{res} = \texttt{Horner}(p, x)$

$s_n = a_n$

for $i = n - 1 : -1 : 0$

$p_i = \mathsf{fl}(s_{i+1} \cdot x)$ % erreur d'arrondi π_i

$s_i = \mathsf{fl}(p_i + a_i)$ % erreur d'arrondi σ_i

end

$\texttt{res} = s_0$

$\gamma_n = n\mathbf{u}/(1 - n\mathbf{u}) \approx n\mathbf{u}$

$$\frac{|p(x) - \texttt{Horner}(p, x)|}{|p(x)|} \leq \gamma_{2n}\, \mathsf{cond}(p, x)$$

Transformations exactes pour la somme

$$x = \mathrm{fl}(a \pm b) \quad \Rightarrow \quad a \pm b = x + y \quad \text{avec } y \in \mathbb{F},$$

Algorithmes de Dekker (1971) et Knuth (1974)

Algorithme 2 (Transformation exacte pour la somme de 2 flottants, nécessite $|a| \geq |b|$)

function $[x, y] = \texttt{FastTwoSum}(a, b)$

$x = \mathrm{fl}(a + b)$

$y = \mathrm{fl}((a - x) + b)$

Algorithme 3 (Transformation exacte pour la somme de 2 flottants)

function $[x, y] = \texttt{TwoSum}(a, b)$

$x = \mathrm{fl}(a + b)$

$z = \mathrm{fl}(x - a)$

$y = \mathrm{fl}((a - (x - z)) + (b - z))$

Transformations exactes pour le produit (1/2)

$$x = \mathrm{fl}(a \cdot b) \quad \Rightarrow \quad a \cdot b = x + y \quad \text{avec } y \in \mathbb{F},$$

Algorithme `TwoProduct` de Veltkamp et Dekker (1971)

$$a = x + y \quad \text{et} \quad x \text{ et } y \text{ ne se chevauchent pas avec } |y| \leq |x|.$$

Algorithme 4 (Séparation exacte d'un flottant en deux parties)

function $[x, y] = \texttt{Split}(a, b)$

$\texttt{factor} = \mathrm{fl}(2^s + 1)$ % $\mathbf{u} = 2^{-p}$, $s = \lceil p/2 \rceil$

$c = \mathrm{fl}(\texttt{factor} \cdot a)$

$x = \mathrm{fl}(c - (c - a))$

$y = \mathrm{fl}(a - x)$

Transformations exactes pour le produit (2/2)

Algorithme 5 (Transformation exacte pour le produit de 2 flottants)

function $[x, y] = \texttt{TwoProduct}(a, b)$

$x = \mathrm{fl}(a \cdot b)$

$[a_1, a_2] = \texttt{Split}(a)$

$[b_1, b_2] = \texttt{Split}(b)$

$y = \mathrm{fl}(a_2 \cdot b_2 - (((x - a_1 \cdot b_1) - a_2 \cdot b_1) - a_1 \cdot b_2))$

Transformations exactes pour la somme et le produit

Étant donnés $a, b, c \in \mathbb{F}$,

- $\texttt{FMA}(a, b, c)$ est l'arrondi au plus près de $a \cdot b + c \in \mathbb{F}$
- $\texttt{ADD3}(a, b, c)$ est l'arrondi au plus près de $a + b + c \in \mathbb{F}$

Algorithme 6 (Transformation exacte pour la somme de 2 flottants)

function $[x, y] = \texttt{TwoSumADD3}(a, b)$
$x = \mathrm{fl}(a + b)$
$y = \texttt{ADD3}(a, b, -x)$

Algorithme 7 (Transformation exacte pour le produit de 2 flottants)

function $[x, y] = \texttt{TwoProductFMA}(a, b)$
$x = \mathrm{fl}(a \cdot b)$
$y = \texttt{FMA}(a, b, -x)$

Transformation exacte pour le schéma de Horner

Algorithme 8 (Transformation exacte pour le schéma de Horner)

function $[\texttt{Horner}(p,x), p_\pi, p_\sigma] = \texttt{EFTHorner}(p,x)$

$s_n = a_n$

for $i = n-1 : -1 : 0$

  $[p_i, \pi_i] = \texttt{TwoProduct}(s_{i+1}, x)$

  $[s_i, \sigma_i] = \texttt{TwoSum}(p_i, a_i)$

  Soit π_i le coefficient de degré i de p_π

  Soit σ_i le coefficient de degré i de p_σ

end

$\texttt{Horner}(p,x) = s_0$

$$\boxed{p(x) = \texttt{Horner}(p,x) + (p_\pi + p_\sigma)(x)}$$

avec $p_\pi(x) = \sum_{i=0}^{n-1} \pi_i x^i$ et $p_\sigma(x) = \sum_{i=0}^{n-1} \sigma_i x^i$ avec π_i et σ_i dans $\mathbb{F}$

Algorithme de Horner compensé

Algorithme 9 (Algorithme de Horner compensé)

function $\texttt{res} = \texttt{CompHorner}(p, x)$
$[h, p_\pi, p_\sigma] = \texttt{EFTHorner}(p, x)$
$c = \texttt{Horner}(p_\pi + p_\sigma, x)$
$\texttt{res} = \mathsf{fl}(h + c)$

Précision de l'algorithme de Horner compensé

Théorème 1

Soient p un polynôme de degré n à coefficients flottants et x un nombre flottant. Alors en absence d'underflow, on a

$$\frac{|\texttt{CompHorner}(p,x) - p(x)|}{|p(x)|} \leq \mathbf{u} + \gamma_{2n}^2 \operatorname{cond}(p,x).$$

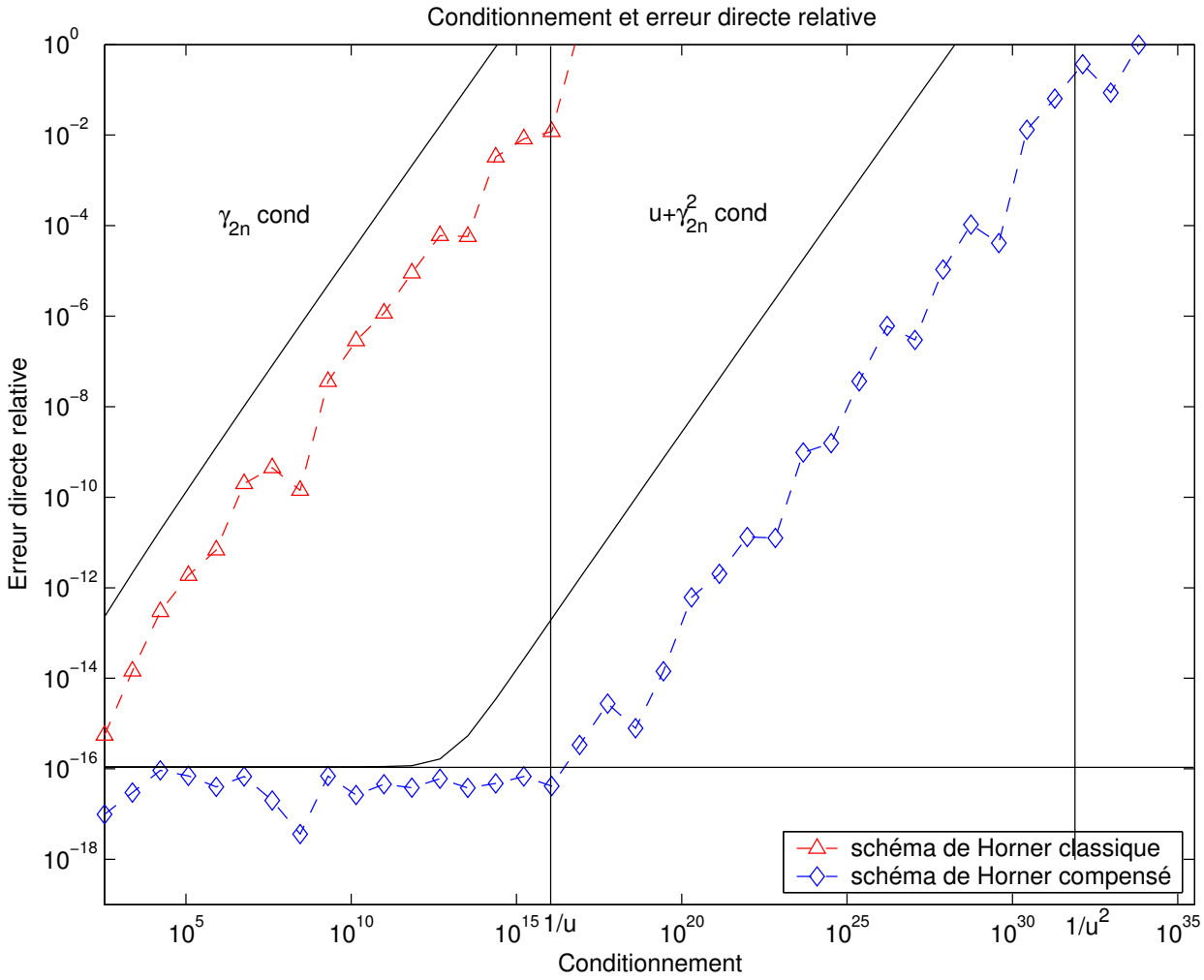
- argument clé de la preuve :

$$(\widetilde{p_\pi} + \widetilde{p_\sigma})(|x|) \leq \gamma_{2n}\widetilde{p}(|x|)$$

- on a une borne similaire en présence d'underflow

Simulations numériques

Évaluation de $p_n(x) = (x-1)^n$ pour $x = \mathrm{fl}(1.333)$ et $n = 3, \ldots, 42$



Mesure de performances

Nous comparons

- `Horner` : algorithme de Horner en double précision IEEE 754
- `CompHorner` : algorithme de Horner compensé
- `DDHorner` : algorithme de Horner avec les double-double

Tous les tests de performances ont été fait en langage C avec la double précision IEEE 754
Pour chaque polynôme p_n avec n variant 3 à 42 :

- on effectue 100 tests mesurant le nombre de cycles,
- on garde la valeur moyenne, la valeur minimum et la valeur maximum des 10 plus petits nombres de cycles.

Mesure de performances

Pentium 4 : 3.0GHz, 1024kB cache L2 - GCC 3.4.1				
ratio	minimum	moyen	maximum	théorique
CompHorner/Horner	1.5	2.9	3.2	13
DDHorner/Horner	2.3	8.4	9.4	17

Intel Celeron : 2.4GHz, 256kB cache L2 - GCC 3.4.1				
ratio	minimum	moyen	maximum	théorique
CompHorner/Horner	1.4	3.1	3.4	13
DDHorner/Horner	2.3	8.4	9.4	17

$\longrightarrow$ algorithme de Horner compensé = algorithme de Horner avec les double-double sans renormalisation

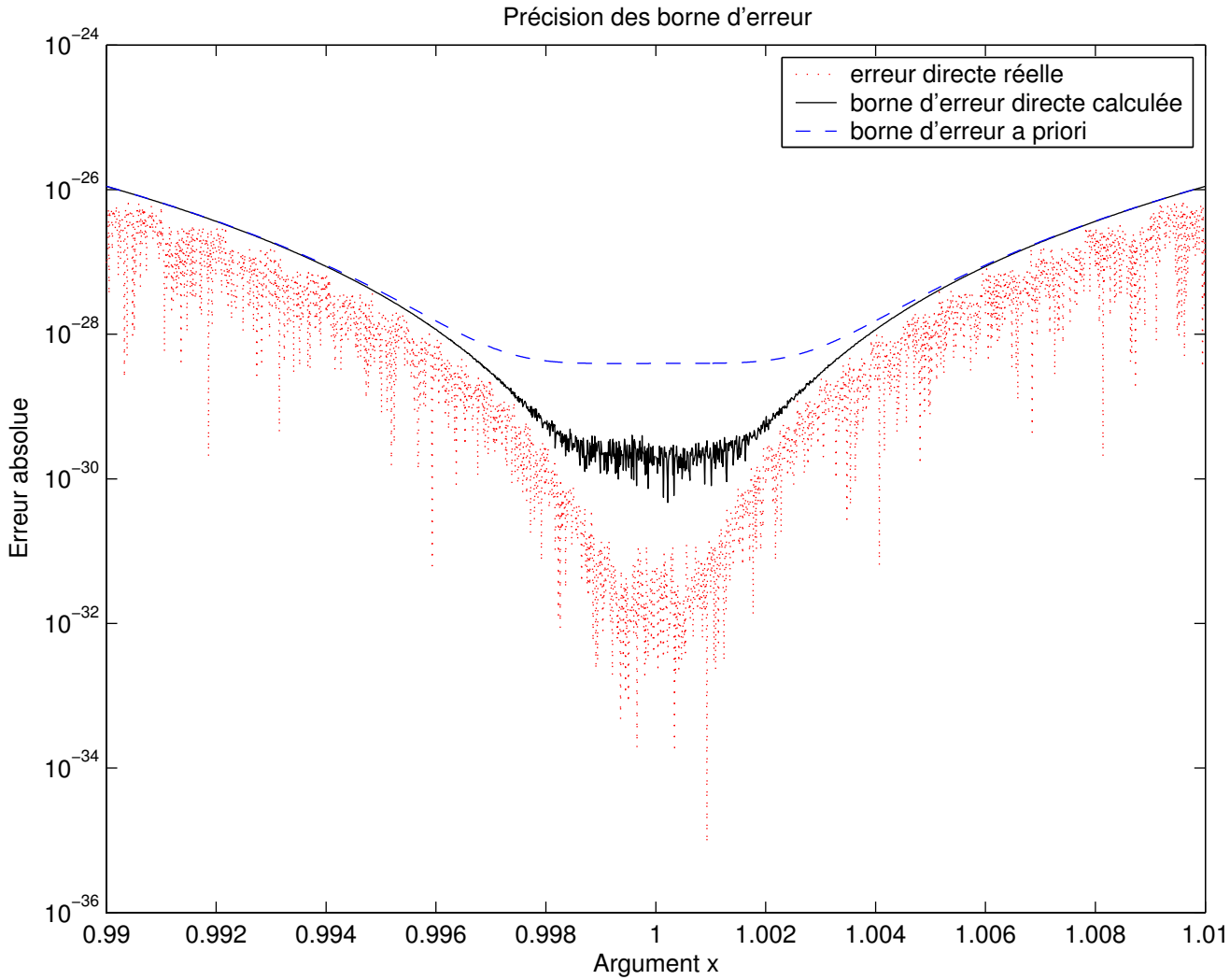
Borne d'erreur certifiée en arithmétique flottante

Théorème 2

Soient p un polynôme de degré n à coefficients flottants, x un nombre flottant et $\texttt{res} = \texttt{CompHorner}(p, x)$ *l'évaluation précise par l'algorithme de Horner compensé de p en x. L'erreur directe absolue commise lors de l'évaluation vérifie*

$$|\texttt{CompHorner}(p, x) - p(x)| \leq \mathrm{fl}((\mathbf{u}|\texttt{res}| + (\gamma_{4n+2}\texttt{HornerSum}(|p_\pi|, |p_\sigma|, |x|) + 2\mathbf{u}^2|\texttt{res}|))).$$

Précision de la borne certifiée pour $p_5(x) = (x-1)^5$



Plan de l'exposé

Applications

L'évaluation polynomiale précise est utilisée dans

- l'évaluation des fonctions élémentaires
- l'évaluation par interpolation polynomiale
- la géométrie algorithmique (intersection de surface, prédicats géométriques)
- le calcul scientifique (calcul des racines de polynômes)
- robotique (polynômes à plusieurs variables)

Prédicats géométriques

Le prédicat Orient3D : détermine si un point D est du côté droit ou gauche du plan orienté défini par A, B et C. Le résultat dépend du signe du déterminant

$$\text{Orient3D}(a,b,c,d) \quad = \quad \text{signe} \begin{vmatrix} a_x & a_y & a_z & 1 \\ b_x & b_y & b_z & 1 \\ c_x & c_y & c_z & 1 \\ d_x & d_y & d_z & 1 \end{vmatrix}$$

$\longrightarrow$ revient à évaluer un polynôme à plusieurs variables.

Approximation de racines simples par la méthode de Horner (1/2)

Polynôme $p(z) = \sum_{i=0}^{n} a_i z^i$ avec une racine simple x

Définition 1

Étant donné un polynôme p et un racine simple x, on définit le conditionnement de la racine par

$$\mathrm{cond}(p, x) = \lim_{\varepsilon \longrightarrow 0} \sup \left\{ \frac{|\Delta x|}{\varepsilon |x|} : |\Delta a_i| \leq \varepsilon |a_i| \right\}.$$

Théorème 3

On a la relation suivante

$$\mathrm{cond}(p, x) = \frac{\widetilde{p}(|x|)}{|x||p'(x)|}.$$

Approximation de racines simples par la méthode de Horner (2/2)

Algorithme de Newton pour les racines simples :

Algorithme 10

$x_0 = \xi$

$x_{i+1} = x_i - \frac{p(x_i)}{p'(x_i)}$; $\quad x_{i+1} = x_i - \frac{\texttt{CompHorner}(p,x_i)}{p'(x_i)}$

Théorème 4

Si ξ est suffisamment proche de x alors l'erreur relative décroît jusqu'au premier i pour lequel

$$\frac{|\widehat{x_i} - x|}{|x|} \leq \mathbf{u} + \mathrm{cond}(p, x)\mathbf{u}^2$$

Et les polynômes à coefficients complexes ?

$$p(z) = \sum_{j=0}^{n} a_j z^j, \qquad a_j \in \mathbb{C}, z = x + iy \in \mathbb{C}$$

$\longrightarrow$ Écrire $p(z) = p_r(x, y) + iq_i(x, y)$ avec p_r et q_r à coefficients réels puis évaluer p_r et q_r avec l'algorithme de Horner précédent

Pb : nécessite des manipulations formelles

$\Rightarrow$ il faut de nouvelles transformations exactes pour les complexes

Transformations exactes complexes (1/2)

Pour $x, y \in \mathbb{F} + i\mathbb{F}$,

$$\mathsf{fl}(x \circ y) = (x \circ y)(1 + \varepsilon_1), \text{ pour } \circ \in \{+, -\} \text{ et } |\varepsilon_\nu| \leq \mathbf{u},$$

et

$$\mathsf{fl}(x \cdot y) = (x \cdot y)(1 + \varepsilon_1), |\varepsilon_1| \leq \sqrt{2}\gamma_2.$$

Algorithme 11 (Transformation exacte pour la somme de 2 flottants $x = a + ib$ and $y = c + id$)

function $[s, e] = \texttt{TwoSumCplx}(x, y)$
  $[s_1, e_1] = \texttt{TwoSum}(a, c)$
  $[s_2, e_2] = \texttt{TwoSum}(b, d)$
  $s = s_1 + is_2$
  $e = e_1 + ie_2$

$$x + y = s + e, \quad s = \mathsf{fl}(x + y), \quad |e| \leq \mathbf{u}|s|, \quad |e| \leq \mathbf{u}|x + y|$$

Transformations exactes complexes (2/2)

Algorithme 12 (Transformation exacte pour le produit de 2 flottants $x = a + ib$ and $y = c + id$)

function $[p, e, f, g] = \texttt{TwoProductCplx}(x, y)$

$[z_1, h_1] = \texttt{TwoProduct}(a, c)$

$[z_2, h_2] = \texttt{TwoProduct}(b, d)$

$[z_3, h_3] = \texttt{TwoProduct}(a, d)$

$[z_4, h_4] = \texttt{TwoProduct}(b, c)$

$[z_5, h_5] = \texttt{TwoSum}(z_1, -z_2)$

$[z_6, h_6] = \texttt{TwoSum}(z_3, z_4)$

$p = z_5 + iz_6$

$e = h_1 + ih_3$

$f = -h_2 + ih_4$

$g = h_5 + ih_6$

$$x \cdot y = p + e + f + g \quad p = \mathsf{fl}(x \cdot y), \quad |e + f + g| \leq \sqrt{2}\gamma_2 |x \cdot y|$$

Conclusion et perspectives

- Le schéma de Horner compensé fournit
  - une précision double de la précision courante,
  - un algorithme deux fois plus rapide que la version double-double,
  - une borne d'erreur dynamique et certifiée.
- Développements passés, en cours et futurs
  - schéma de Horner compensé : underflow, avec `FMA`, pour le `FMA`
  - même technique avec la méthode de Newton pour d'autres problèmes

La future révision de la norme IEEE 754 devrait inclure les fonctions `tailadd`, `tailsubtract` et `tailmultiply` qui calculent l'erreur d'arrondi pour l'addition, la soustraction et la multiplication.

Références I

Nicholas J. Higham.
Accuracy and stability of numerical algorithms.
Society for Industrial and Applied Mathematics (SIAM), Philadelphia, PA, second edition, 2002.

Takeshi Ogita, Siegfried M. Rump, and Shin'ichi Oishi.
Accurate sum and dot product.
SIAM J. Sci. Comput., 26(6) :1955–1988, 2005.

Douglas M. Priest.
On Properties of Floating Point Arithmetics : Numerical Stability and the Cost of Accurate Computations.
PhD thesis, Mathematics Department, University of California, Berkeley, CA, USA, November 1992.

Références II

Jonathan Richard Shewchuk.
Adaptive precision floating-point arithmetic and fast robust geometric predicates.
Discrete Comput. Geom., 18(3) :305–363, 1997.

Françoise Tisseur.
Newton's method in floating point arithmetic and iterative refinement of generalized eigenvalue problems.
SIAM J. Matrix Anal. Appl., 22(4) :1038–1057, 2001.