Quand les boucles deviennent des polynômes

ou

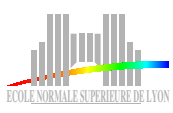
l'implantation automatique de fonctions

Présentation aux R.A.I.M. 2008

Christoph Quirin Lauter

Équipe-projet Arénaire
Laboratoire de l'Informatique et du Parallélisme
École Normale Supérieure de Lyon

Lille, 04 juin 2008



Introduction

Fonctions mathématiques correctement arrondies - crlibm

crlibm[1]: bibliothèque de fonctions correctement arrondies

[1] http://lipforge.ens-lyon.fr/www/crlibm/

Fonctions mathématiques correctement arrondies - crlibm

crlibm[1]: bibliothèque de fonctions correctement arrondies

- Fonctions mathématiques typiques comme dans une libm traditionnelle:
  - exp
  - sin
  - cos
  - ...

[1] http://lipforge.ens-lyon.fr/www/crlibm/

Fonctions mathématiques correctement arrondies - `crlibm`

`crlibm`[1]: bibliothèque de fonctions correctement arrondies

- Fonctions mathématiques typiques comme dans une `libm` traditionnelle:
  - `exp`
  - `sin`
  - `cos`
  - `...`
- Résultats exacts bit-à-bit: $\texttt{f(x)} = \circ(f(x))$

[1] `http://lipforge.ens-lyon.fr/www/crlibm/`

Fonctions mathématiques correctement arrondies - `crlibm`

`crlibm`[1]: bibliothèque de fonctions correctement arrondies

- Fonctions mathématiques typiques comme dans une `libm` traditionnelle:
  - `exp`
  - `sin`
  - `cos`
  - `...`
- Résultats exacts bit-à-bit: $\texttt{f(x)} = \circ(f(x))$
- Performance en moyenne égalisant les `libms` traditionnelles

[1] `http://lipforge.ens-lyon.fr/www/crlibm/`

Fonctions mathématiques correctement arrondies - `crlibm`

`crlibm`[1]: bibliothèque de fonctions correctement arrondies

- Fonctions mathématiques typiques comme dans une `libm` traditionnelle:
  - `exp`
  - `sin`
  - `cos`
  - `...`
- Résultats exacts bit-à-bit: $\texttt{f(x)} = \circ(f(x))$
- Performance en moyenne égalisant les `libms` traditionnelles
- Temps d'évaluation garanti dans le pire cas

[1] `http://lipforge.ens-lyon.fr/www/crlibm/`

Fonctions mathématiques correctement arrondies - `crlibm`

`crlibm`[1]: bibliothèque de fonctions correctement arrondies

- Fonctions mathématiques typiques comme dans une `libm` traditionnelle:
  - `exp`
  - `sin`
  - `cos`
  - `...`
- Résultats exacts bit-à-bit: $\texttt{f(x)} = \circ(f(x))$
- Performance en moyenne égalisant les `libms` traditionnelles
- Temps d'évaluation garanti dans le pire cas
- Défi: l'arrondi correct demande des dizaines implantations à haute précision avec des preuves complètes

[1] `http://lipforge.ens-lyon.fr/www/crlibm/`

Developpement de bibliothèques libm

Développement de fonctions en Arénaire – 1

Première fonction dans `crlibm`

Développement de fonctions en Arénaire – 1

Première fonction dans `crlibm`

- `exp(x)` par David Defour

Développement de fonctions en Arénaire – 1

Première fonction dans `crlibm`

- `exp(x)` par David Defour
- arrondi correct en deux étapes

Développement de fonctions en Arénaire – 1

Première fonction dans `crlibm`

- `exp(x)` par David Defour
- arrondi correct en deux étapes
- code `C` portable
- blibliothèque de calcul sur entiers pour la deuxième étape

Développement de fonctions en Arénaire – 1

Première fonction dans `crlibm`

- `exp(x)` par David Defour
- arrondi correct en deux étapes
- code `C` portable
- blibliothèque de calcul sur entiers pour la deuxième étape
- preuve complexe, écrite à la main

Développement de fonctions en Arénaire – 1

Première fonction dans `crlibm`

- `exp(x)` par David Defour
- arrondi correct en deux étapes
- code `C` portable
- blibliothèque de calcul sur entiers pour la deuxième étape
- preuve complexe, écrite à la main
- temps de développement: une thèse

Développement de fonctions en Arénaire – 2

Une implantation alternative

Développement de fonctions en Arénaire – 2

Une implantation alternative

- `exp(x)` par moi-même

Développement de fonctions en Arénaire – 2

Une implantation alternative

- `exp(x)` par moi-même
- arrondi correct en une étape

Développement de fonctions en Arénaire – 2

Une implantation alternative

- `exp(x)` par moi-même
- arrondi correct en une étape
- utilisation d'instructions spéciales Itanium

Développement de fonctions en Arénaire – 2

Une implantation alternative

- `exp(x)` par moi-même
- arrondi correct en une étape
- utilisation d'instructions spéciales Itanium
- preuve complexe, écrite à la main, fausse

Développement de fonctions en Arénaire – 2

Une implantation alternative

- `exp(x)` par moi-même
- arrondi correct en une étape
- utilisation d'instructions spéciales Itanium
- preuve complexe, écrite à la main, fausse
- temps de développement: un stage de 3 mois

Développement de fonctions en Arénaire – 3

Fonctions suivantes en `crlibm`: `atan(x)`, `log(x)`...

Développement de fonctions en Arénaire – 3

Fonctions suivantes en `crlibm`: `atan(x)`, `log(x)`...

- Début de l'automatisation : scripts Maple pour les fichiers en-tête

Développement de fonctions en Arénaire – 3

Fonctions suivantes en `crlibm`: `atan(x)`, `log(x)`...

- Début de l'automatisation : scripts Maple pour les fichiers en-tête
- Calcul de normes infinies en Maple

Développement de fonctions en Arénaire – 3

Fonctions suivantes en `crlibm`: `atan(x)`, `log(x)`...

- Début de l'automatisation : scripts Maple pour les fichiers en-tête
- Calcul de normes infinies en Maple
- Preuves en Gappa, écrites à la main

Développement de fonctions en Arénaire – 3

Fonctions suivantes en `crlibm`: `atan(x)`, `log(x)`...

- Début de l'automatisation : scripts Maple pour les fichiers en-tête
- Calcul de normes infinies en Maple
- Preuves en Gappa, écrites à la main
- temps de développement: à peu près 1 mois par fonction

Coût d'une bibliothèque (cr)libm

- Coût d'une fonction: 1 homme-mois

Coût d'une bibliothèque (cr)libm

- Coût d'une fonction: 1 homme-mois
- Une libm typique contient à peu près 35 fonctions

Coût d'une bibliothèque (cr)libm

- Coût d'une fonction: 1 homme-mois

- Une libm typique contient à peu près 35 fonctions
- Avec l'approche CRLibm, on implante deux algorithmes (phases rapide et précise)

Coût d'une bibliothèque (cr)libm

- Coût d'une fonction: 1 homme-mois

- Une libm typique contient à peu près 35 fonctions
- Avec l'approche CRLibm, on implante deux algorithmes (phases rapide et précise)
- Une optimisation est nécessaire pour

Coût d'une bibliothèque (cr)libm

- Coût d'une fonction: 1 homme-mois

- Une libm typique contient à peu près 35 fonctions
- Avec l'approche CRLibm, on implante deux algorithmes (phases rapide et précise)
- Une optimisation est nécessaire pour
  - IA32 avec une FPU x87

Coût d'une bibliothèque (cr)libm

- Coût d'une fonction: 1 homme-mois

- Une libm typique contient à peu près 35 fonctions
- Avec l'approche CRLibm, on implante deux algorithmes (phases rapide et précise)
- Une optimisation est nécessaire pour
  - IA32 avec une FPU x87
  - IA32 avec SSEx

Coût d'une bibliothèque (cr)libm

- Coût d'une fonction: 1 homme-mois

- Une libm typique contient à peu près 35 fonctions
- Avec l'approche CRLibm, on implante deux algorithmes (phases rapide et précise)
- Une optimisation est nécessaire pour
  - IA32 avec une FPU x87
  - IA32 avec SSEx
  - IA64

Coût d'une bibliothèque (cr)libm

- Coût d'une fonction: 1 homme-mois

- Une libm typique contient à peu près 35 fonctions
- Avec l'approche CRLibm, on implante deux algorithmes (phases rapide et précise)
- Une optimisation est nécessaire pour
  - IA32 avec une FPU x87
  - IA32 avec SSEx
  - IA64
  - PowerPC

Coût d'une bibliothèque (cr)libm

- Coût d'une fonction: 1 homme-mois

- Une libm typique contient à peu près 35 fonctions
- Avec l'approche CRLibm, on implante deux algorithmes (phases rapide et précise)
- Une optimisation est nécessaire pour
  - IA32 avec une FPU x87
  - IA32 avec SSEx
  - IA64
  - PowerPC
  - processeurs embarqués

Coût d'une bibliothèque (cr)libm

- Coût d'une fonction: 1 homme-mois

- Une libm typique contient à peu près 35 fonctions
- Avec l'approche CRLibm, on implante deux algorithmes (phases rapide et précise)
- Une optimisation est nécessaire pour
  - IA32 avec une FPU x87
  - IA32 avec SSEx
  - IA64
  - PowerPC
  - processeurs embarqués
  - en matériel pour certains processeurs

Coût d'une bibliothèque (cr)libm

- Coût d'une fonction: 1 homme-mois

- Une libm typique contient à peu près 35 fonctions
- Avec l'approche CRLibm, on implante deux algorithmes (phases rapide et précise)
- Une optimisation est nécessaire pour
  - IA32 avec une FPU x87
  - IA32 avec SSEx
  - IA64
  - PowerPC
  - processeurs embarqués
  - en matériel pour certains processeurs
  - la latence ou pour le débit

Coût d'une bibliothèque (cr)libm

- Coût d'une fonction: 1 homme-mois

- Une libm typique contient à peu près 35 fonctions
- Avec l'approche CRLibm, on implante deux algorithmes (phases rapide et précise)
- Une optimisation est nécessaire pour
  - IA32 avec une FPU x87
  - IA32 avec SSEx
  - IA64
  - PowerPC
  - processeurs embarqués
  - en matériel pour certains processeurs
  - la latence ou pour le débit
  - et on repète tout quand le matériel/le compilateur/le système change

Coût d'une bibliothèque (cr)libm

- Coût d'une fonction: 1 homme-mois

- Une libm typique contient à peu près 35 fonctions
- Avec l'approche CRLibm, on implante deux algorithmes (phases rapide et précise)
- Une optimisation est nécessaire pour
  - IA32 avec une FPU x87
  - IA32 avec SSEx
  - IA64
  - PowerPC
  - processeurs embarqués
  - en matériel pour certains processeurs
  - la latence ou pour le débit
  - et on repète tout quand le matériel/le compilateur/le système change

$\Rightarrow$ On devrait faire l'implantation automatiquement!

Un générateur d'implantation automatique

- Un générateur d'implantations automatique prototypé

Un générateur d'implantation automatique

- Un générateur d'implantations automatique prototypé
- Entrées:
  - une fonction f

Un générateur d'implantation automatique

- Un générateur d'implantations automatique prototypé
- Entrées:
  - une fonction f
  - un domaine $I = [a; b]$

Un générateur d'implantation automatique

- Un générateur d'implantations automatique prototypé
- Entrées:
  - une fonction f
  - un domaine $I = [a; b]$
  - une erreur cible $\overline{\varepsilon} \in \mathbb{R}^+$

Un générateur d'implantation automatique

- Un générateur d'implantations automatique prototypé
- Entrées:
  - une fonction f
  - un domaine $I = [a; b]$
  - une erreur cible $\overline{\varepsilon} \in \mathbb{R}^+$
- Sortie:
  - Un code F tel que

$$\forall x \in I, \left| \frac{F(x)}{f(x)} - 1 \right| \leq \overline{\varepsilon}$$

Un générateur d'implantation automatique

- Un générateur d'implantations automatique prototypé
- Entrées:
  - une fonction f
  - un domaine $I = [a; b]$
  - une erreur cible $\overline{\varepsilon} \in \mathbb{R}^+$
- Sortie:
  - Un code F tel que

$$\forall x \in I, \left| \frac{F(x)}{f(x)} - 1 \right| \leq \overline{\varepsilon}$$

- Langage d'implantation: Sollya

Un générateur d'implantation automatique

- Un générateur d'implantations automatique prototypé
- Entrées:
  - une fonction f
  - un domaine $I = [a; b]$
  - une erreur cible $\overline{\varepsilon} \in \mathbb{R}^+$
- Sortie:
  - Un code F tel que

$$\forall x \in I, \left| \frac{F(x)}{f(x)} - 1 \right| \leq \overline{\varepsilon}$$

- Langage d'implantation: Sollya
- $\Rightarrow$ Temps de développement pour $\sin \pi$, $\cos \pi$, $\tan \pi$:

Un générateur d'implantation automatique

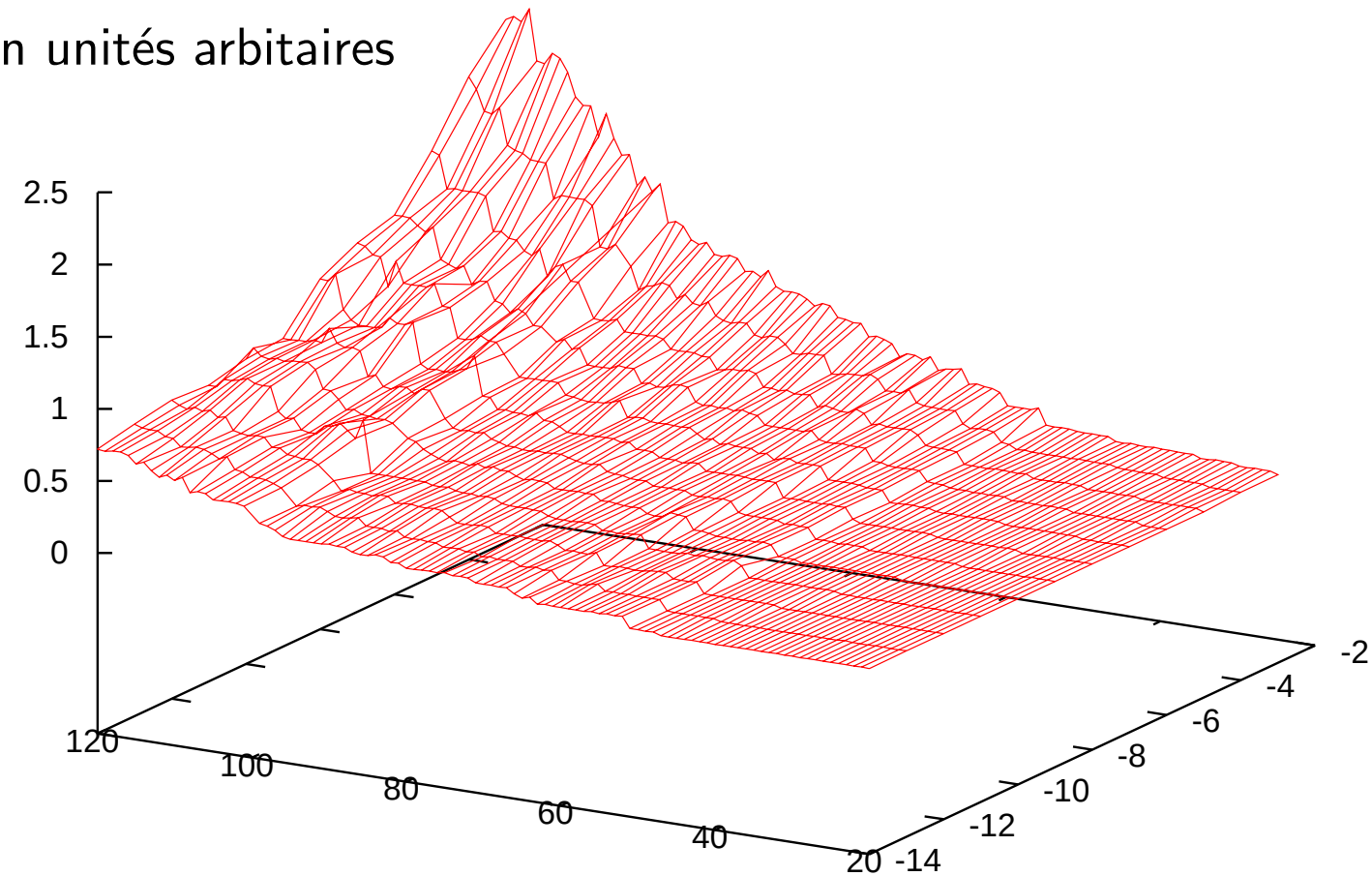
- Un générateur d'implantations automatique prototypé
- Entrées:
  - une fonction f
  - un domaine $I = [a; b]$
  - une erreur cible $\overline{\varepsilon} \in \mathbb{R}^+$
- Sortie:
  - Un code F tel que

$$\forall x \in I, \left| \frac{F(x)}{f(x)} - 1 \right| \leq \overline{\varepsilon}$$

- Langage d'implantation: Sollya
- $\Rightarrow$ Temps de développement pour $\sin \pi$, $\cos \pi$, $\tan \pi$: 2 jours

Resultats

- $\log(1+x)$
- Argument réduit x dans un intervalle de largeur entre 2^{-13} et 2^{-1}
- Précision entre 20 et 120 bits
- 1203 implantations, tous prouvés formellement en Gappa
- Temps en unités arbitaires



Élargissement de l'espace de recherche

- L'implantation de bibliothèques libm a été

Élargissement de l'espace de recherche

- L'implantation de bibliothèques libm a été
  - un savoir-faire experimental connu par quelques spécialistes

Élargissement de l'espace de recherche

- L'implantation de bibliothèques libm a été
  - un savoir-faire experimental connu par quelques spécialistes
  - beaucoup de recettes de cuisine

Élargissement de l'espace de recherche

- L'implantation de bibliothèques libm a été
  - un savoir-faire experimental connu par quelques spécialistes
  - beaucoup de recettes de cuisine
  - une lutte avec la certification fastidieuse

Élargissement de l'espace de recherche

- L'implantation de bibliothèques libm a été
  - un savoir-faire experimental connu par quelques spécialistes
  - beaucoup de recettes de cuisine
  - une lutte avec la certification fastidieuse
  - beaucoup de travail manuel

Élargissement de l'espace de recherche

- L'implantation de bibliothèques libm a été
  - un savoir-faire experimental connu par quelques spécialistes
  - beaucoup de recettes de cuisine
  - une lutte avec la certification fastidieuse
  - beaucoup de travail manuel

  $\Rightarrow$ on ratait beaucoup d'optimisations

Élargissement de l'espace de recherche

- L'implantation de bibliothèques libm a été
  - un savoir-faire experimental connu par quelques spécialistes
  - beaucoup de recettes de cuisine
  - une lutte avec la certification fastidieuse
  - beaucoup de travail manuel

  $\Rightarrow$ on ratait beaucoup d'optimisations

- Maintenant, on peut produire
  - des approximations polynomiales adaptées

Élargissement de l'espace de recherche

- L'implantation de bibliothèques libm a été
  - un savoir-faire experimental connu par quelques spécialistes
  - beaucoup de recettes de cuisine
  - une lutte avec la certification fastidieuse
  - beaucoup de travail manuel

  $\Rightarrow$ on ratait beaucoup d'optimisations

- Maintenant, on peut produire
  - des approximations polynomiales adaptées
  - des implantations avec une précision ajusté au minimum

Élargissement de l'espace de recherche

- L'implantation de bibliothèques libm a été
  - un savoir-faire experimental connu par quelques spécialistes
  - beaucoup de recettes de cuisine
  - une lutte avec la certification fastidieuse
  - beaucoup de travail manuel

  $\Rightarrow$ on ratait beaucoup d'optimisations

- Maintenant, on peut produire
  - des approximations polynomiales adaptées
  - des implantations avec une précision ajusté au minimum
  - des preuves formelles de correction

Élargissement de l'espace de recherche

- L'implantation de bibliothèques libm a été
  - un savoir-faire experimental connu par quelques spécialistes
  - beaucoup de recettes de cuisine
  - une lutte avec la certification fastidieuse
  - beaucoup de travail manuel

  $\Rightarrow$ on ratait beaucoup d'optimisations

- Maintenant, on peut produire
  - des approximations polynomiales adaptées
  - des implantations avec une précision ajusté au minimum
  - des preuves formelles de correction
  - automagiquement en quelques secondes

Élargissement de l'espace de recherche

- L'implantation de bibliothèques libm a été
  - un savoir-faire experimental connu par quelques spécialistes
  - beaucoup de recettes de cuisine
  - une lutte avec la certification fastidieuse
  - beaucoup de travail manuel

  $\Rightarrow$ on ratait beaucoup d'optimisations

- Maintenant, on peut produire
  - des approximations polynomiales adaptées
  - des implantations avec une précision ajusté au minimum
  - des preuves formelles de correction
  - automagiquement en quelques secondes

  $\Rightarrow$ parcourir un espace de recherche large est possible

Transformations numériques de codes

Quand les boucles deviennent des polynômes...

- Le générateur d'implantation ne fait qu'échantillonner la fonction à implanter

Quand les boucles deviennent des polynômes...

- Le générateur d'implantation ne fait qu'échantillonner la fonction à implanter
- Plus précisement:

Quand les boucles deviennent des polynômes...

- Le générateur d'implantation ne fait qu'échantillonner la fonction à implanter
- Plus précisement:
  - $f \in \mathcal{C}^2$

Quand les boucles deviennent des polynômes...

- Le générateur d'implantation ne fait qu'échantillonner la fonction à implanter
- Plus précisement:
  - $f \in \mathcal{C}^2$
  - possiblité de calculer f, f' et f'' en des points demandés

Quand les boucles deviennent des polynômes...

- Le générateur d'implantation ne fait qu'échantillonner la fonction à implanter
- Plus précisement:
  - $f \in \mathcal{C}^2$
  - possiblité de calculer f, f' et f'' en des points demandés
  - avec une précision suffisante

Quand les boucles deviennent des polynômes...

- Le générateur d'implantation ne fait qu'échantillonner la fonction à implanter
- Plus précisement:
  - $f \in \mathcal{C}^2$
  - possiblité de calculer f, f' et f'' en des points demandés
  - avec une précision suffisante
- Sollya permet de lier un code externe à un symbôle de fonction

Quand les boucles deviennent des polynômes...

- Le générateur d'implantation ne fait qu'échantillonner la fonction à implanter
- Plus précisement:
  - $f \in \mathcal{C}^2$
  - possiblité de calculer f, f' et f'' en des points demandés
  - avec une précision suffisante
- Sollya permet de lier un code externe à un symbôle de fonction
- Cette fonction externe peut être un code à optimiser

Quand les boucles deviennent des polynômes...

- Le générateur d'implantation ne fait qu'échantillonner la fonction à implanter
- Plus précisement:
  - $f \in \mathcal{C}^2$
  - possiblité de calculer f, f' et f'' en des points demandés
  - avec une précision suffisante
- Sollya permet de lier un code externe à un symbôle de fonction
- Cette fonction externe peut être un code à optimiser

La génération d'implantations devient une transformation de codes

Quand les boucles deviennent des polynômes...

- Le générateur d'implantation ne fait qu'échantillonner la fonction à implanter
- Plus précisement:
  - $f \in \mathcal{C}^2$
  - possiblité de calculer f, f' et f'' en des points demandés
  - avec une précision suffisante
- Sollya permet de lier un code externe à un symbôle de fonction
- Cette fonction externe peut être un code à optimiser

La génération d'implantations devient une transformation de codes

Les boucles deviennent des polynômes...

Un exemple...

$$argerf = erf^{-1} \text{ sur } I = \left[-2^{-4}; 2^{-4}\right] \text{ avec 20 bits}$$

Un exemple...

$$argerf = erf^{-1} \text{ sur } I = \left[-2^{-4}; 2^{-4}\right] \text{ avec 20 bits}$$

```
void argerfMpfr(mpfr_t rop, mpfr_t
     op, mp_rnd_t rnd, mp_prec_t
     prec) {
  mpfr_t temp, absOp, temp2, temp3;
  node *func, *deriv;
  mpfr_init2(temp, prec + 10);
  mpfr_set_d(temp,1.0,GMP_RNDN);
  mpfr_init2(absOp,mpfr_get_prec(op
      ));
  mpfr_abs(absOp,op,GMP_RNDN);
  mpfr_init2(temp2,prec + 10);
  mpfr_init2(temp3,prec + 10);
  func = makeSub(makeErf(
      makeVariable()),
      makeConstant(absOp));
  deriv = differentiate(func);
  mpfr_set_d(temp3, 1000000.0,
      GMP_RNDN);
  mpfr_set_d(temp2, 0.0, GMP_RNDN);
  mpfr_nextabove(temp2);
  newtonMPFRWithStartPoint(temp,
      func, deriv, temp2, temp3,
      temp2, prec + 20);
  ....
```

```
#define p_coeff_1h
     8.86226681313185...e-01
#define p_coeff_3h
     2.32513194330223...e-01
void p(double *p_resh, double x) {
 double p_x_0_pow2h;
 double p_t_1_0h;
 double p_t_2_0h;
 double p_t_3_0h;
 double p_t_4_0h;

 p_x_0_pow2h = x * x;
 p_t_1_0h = p_coeff_3h;
 p_t_2_0h = p_t_1_0h * p_x_0_pow2h;
 p_t_3_0h = p_coeff_1h + p_t_2_0h;
 p_t_4_0h = p_t_3_0h * x;
 *p_resh = p_t_4_0h;
}
```

Conclusion

Conclusion et extensions...

- Implantion d'une fonction libm automatisée

Conclusion et extensions...

- Implantion d'une fonction libm automatisée
- Temps de développement réduit d'un facteur 15

Conclusion et extensions...

- Implantion d'une fonction libm automatisée
- Temps de développement réduit d'un facteur 15
- Plus de confiance dans des preuves automatiquement générées

Conclusion et extensions...

- Implantion d'une fonction libm automatisée
- Temps de développement réduit d'un facteur 15
- Plus de confiance dans des preuves automatiquement générées

- Transformation de codes possibles:

Conclusion et extensions...

- Implantion d'une fonction libm automatisée
- Temps de développement réduit d'un facteur 15
- Plus de confiance dans des preuves automatiquement générées

- Transformation de codes possibles:
  - Prototypage rapide, ensuite optimisation automatique

Conclusion et extensions...

- Implantion d'une fonction libm automatisée
- Temps de développement réduit d'un facteur 15
- Plus de confiance dans des preuves automatiquement générées

- Transformation de codes possibles:
  - Prototypage rapide, ensuite optimisation automatique
  - Adapter la précision d'un code automatiquement

Conclusion et extensions...

- Implantion d'une fonction libm automatisée
- Temps de développement réduit d'un facteur 15
- Plus de confiance dans des preuves automatiquement générées

- Transformation de codes possibles:
  - Prototypage rapide, ensuite optimisation automatique
  - Adapter la précision d'un code automatiquement
  - Codes transformés automatiquement et formellement certifiés

Conclusion et extensions...

- Implantion d'une fonction libm automatisée
- Temps de développement réduit d'un facteur 15
- Plus de confiance dans des preuves automatiquement générées

- Transformation de codes possibles:
  - Prototypage rapide, ensuite optimisation automatique
  - Adapter la précision d'un code automatiquement
  - Codes transformés automatiquement et formellement certifiés

- Investigations:

Conclusion et extensions...

- Implantion d'une fonction libm automatisée
- Temps de développement réduit d'un facteur 15
- Plus de confiance dans des preuves automatiquement générées

- Transformation de codes possibles:
  - Prototypage rapide, ensuite optimisation automatique
  - Adapter la précision d'un code automatiquement
  - Codes transformés automatiquement et formellement certifiés

- Investigations:
  - Approximations non-polynomiales

Conclusion et extensions...

- Implantion d'une fonction libm automatisée
- Temps de développement réduit d'un facteur 15
- Plus de confiance dans des preuves automatiquement générées

- Transformation de codes possibles:
  - Prototypage rapide, ensuite optimisation automatique
  - Adapter la précision d'un code automatiquement
  - Codes transformés automatiquement et formellement certifiés

- Investigations:
  - Approximations non-polynomiales
  - Évaluations autres que Horner

Conclusion et extensions...

- Implantion d'une fonction libm automatisée
- Temps de développement réduit d'un facteur 15
- Plus de confiance dans des preuves automatiquement générées

- Transformation de codes possibles:
  - Prototypage rapide, ensuite optimisation automatique
  - Adapter la précision d'un code automatiquement
  - Codes transformés automatiquement et formellement certifiés

- Investigations:
  - Approximations non-polynomiales
  - Évaluations autres que Horner
  - Arithmétiques diverses

Merci!

Merci pour votre attention !

Des questions ?