Rapport sur tâche 7
Qualité des fonctions élémentaires
Cas de la précision infinie

Paul Zimmermann

October 30, 1997

1 Logiciels existants

La référence principale sur les logiciels existants est la page *Interval Computations*.[1] Parmi les systèmes de calcul formel, MATHEMATICA et MAPLE ont une arithmétique d'intervalle en précision arbitraire.

1.1 Fonction `Interval` de MATHEMATICA

La fonction `Interval` permet de définir des intervalles (à bornes symboliques ou numériques) :

```
In[1]:= a=Interval[{0.001,0.0011}];

In[2]:= Exp[a]

Out[2]= Interval[{1.001, 1.0011}]

In[3]:= N[Exp[a],16]

Out[3]= Interval[{1.001000500166708, 1.001100605221894}]
```

Cependant les calculs numériques sont limités aux fonctions élémentaires. Les fonctions spéciales comme la fonction Zeta de Riemann ne sont pas supportées. De plus il ne semble pas possible d'avoir des arrondis dirigés en entrée et surtout en sortie, ce qui est plus gênant.

1.2 Module `Intpak` de MAPLE

Ce module est dans la *share-library* de MAPLE depuis la version V.3 (au moins). Pour le charger, il faut faire :

```
> with(share):
> readshare(intpak,numerics):
> with(intpak):
> a:=construct(0.001,0.0011):
> &exp(a);
                    [1.001000499, 1.001100606]
```

Il est dit expérimental par ses auteurs, A. E. Connell et R. M. Corless :

[1] `http://cs.utep.edu/interval-comp/main.html`

```
> ?Interval,&exp
See Interval[logarithmic]

LOCATION: readshare(intpak,numerics);

> ?Interval[logarithmic]
Help error, help for `Interval,logarithmic` not found
```

2 Calcul de $\exp x$

Nous décrivons ici une implantation en MuPAD des différentes méthodes indiquées par Jean-Michel Muller sur sa page Fiable.[2]

2.1 Développement de Taylor

Une première méthode pour calculer $\exp(x)$ en grande précision est d'effectuer un développement de Taylor, après une réduction d'argument :

$$\exp(x) = (1 + r + \frac{r^2}{2!} + \cdots)^{2^k} \cdot 2^n$$

où $2^k r = x - n \log 2$. Le code MuPAD correspondant est le suivant (la fonction `lntwo` renvoie $\log(2)$, que l'on suppose tabulé, avec la précision courante) :

```
exp1:=proc(x,K) local d,n,r,s,t,k,e,dd,DIGITS;
begin
   x:=float(x);
   d:=DIGITS; DIGITS:=9; n:=round(x/0.6931471805);
   if args(0)=1 then K:=round(sqrt(5*d)) end_if;
   dd:=d+round(K*0.6931471805); DIGITS:=dd;
   r:=(if n<>0 then n*lntwo() else 0 end_if);
   r:=x-r; r:=r/2^K;
   k:=0; t:=1; s:=1.0; DIGITS:=9; e:=10.0^(-d-3-K*0.3);
   repeat
      DIGITS:=dd;
      k:=k+1;
      t:=t*r/k;
      s:=s+t;
   until (DIGITS:=9;abs(t)<=s*e*(1-r/k)) end_repeat;
   DIGITS:=dd;
   for k from 1 to K do s:=s*s end_for;
   if n<>0 then s:=s*2^n end_if; DIGITS:=d; float(s)
end_proc:
```

Il semble que la valeur optimale de k soit de l'ordre de $\sqrt{d}$ où d est le nombre désiré de décimales.

2.2 Méthode de Newton

Une autre méthode consiste à utiliser la méthode de Newton pour calculer $\exp(x)$ à partir du logarithme, en utilisant l'itération

$$y_{k+1} = y_k(x + 1 - \log y_k).$$

Cela donne le code MuPAD suivant :

[2] `http://www.ens-lyon.fr/~jmmuller/Fiable.html`

```
newton:=proc(f,x) local oldx,l,k;
begin
   l:=[DIGITS]; while l[1]>9 do l:=[(l[1]+1) div 2].l end_while;
   for k from 2 to nops(l) do DIGITS:=l[k]; oldx:=x; x:=f(x) end_for; x
end_proc:

exp2:=proc(x) local d;
begin
   d:=DIGITS; DIGITS:=9; exp(x); DIGITS:=d;
   newton(subs(fun(args(1)*(x+1-ln(args(1)))),hold(x)=x),%2)
end_proc:
```

3 Calcul de $\log x$

3.1 Moyenne arithmético-géométrique

Le calcul du logarithme peut se faire par la formule

$$\log x = \frac{\pi}{2A(1,4/s)} - m \log 2$$

où $s = x2^m > 2^{p/2}$ et p est le nombre de bits cherchés. Cette méthode nécessite le calcul de π et $\log 2$.

```
AG:=proc(a,b)
begin
   repeat
      (a+b)/2.0; b:=(a*b)^(1/2); a:=%2
   until b-a=0.0 end_repeat;
   b
end_proc:

ln1:=proc(x) local d,m,s,k;
begin
   d:=DIGITS; x:=float(x);
   DIGITS:=9; m:=(1.1513*d-ln(x))/0.693; m:=ceil(m+ln(m));
   DIGITS:=(d:=d+3); x:=float(x); x:=x*2^m;
   x:=fPI()/2/AG(1.0,4/x)-m*lntwo();
   DIGITS:=d-3; float(x)
end_proc:
```

4 Résultats

Le tableau ci-dessous indique les temps de calcul des fonctions standards MuPAD `exp` et `ln` et des différentes méthodes indiquées ci-dessus.

DIGITS	1000	2000	5000	10000
`exp`(a)	0.07	0.35	1.9	8.4
`exp1`(a)/k	0.5/71	1.3/100	7.7/158	64.5/224
`exp2`(a)	0.14	0.6	2.6	21.2
`ln`(a)	0.08	0.40	2.1	12.6
`ln1`(a)	0.13	0.45	2.5	5.5

On peut constater que la méthode basée sur la moyenne arithmético-géométrique, malgré une programmation non optimisée, est plus rapide que la fonction standard `ln` pour plus de 5000 chiffres environ.

Le tableau ci-dessous compare les temps de calcul et la précision de plusieurs valeurs sur un Pentium Pro 200 Mhz. On note $a = \pi/3$. Chaque entrée contient deux temps t/u où t est le temps en flottant et u celui en arithmétique d'intervalle.

valeur	précision	Mma 3.0	Maple V.4	MuPAD 1.3
$\exp(a)$	1000	0.06/0.11	0.13/0.14	0.07/
$\exp(a)$	2000	0.28/0.55	0.46/0.47	0.33/
$\exp(a)$	5000	2.6/5.0	2.4/2.8	1.9/
$\exp(a)$	10000	14.0/28.0	4.3/7.7	8.4/
$\log(a)$	1000	0.6/0.4	0.11/0.12	0.14/
$\log(a)$	2000	1.1/1.9	0.45/0.44	0.4/
$\log(a)$	5000	8.4/20	2.0/2.7	2.8/
$\log(a)$	10000	61/138	7.7/3.8	6.5/