pmaxima

```
Maxima 5.27.0 http://maxima.sourceforge.net
using Lisp GNU Common Lisp (GCL) GCL 2.6.7 (a.k.a. GCL)
Distributed under the GNU Public License. See the file COPYING.
Dedicated to the memory of William Schelter.
```

Sommes de Riemann

1 Présentation

Une *subdivision* σ d'un intervalle $[a,b]$ de $\mathbf{R}$ est un ensemble $(x_i)_{0\leqslant i\leqslant n}$ $(n\in\mathbf{N}^*)$ de réels tels que :

$$x_0 = a < x_1 < \ldots < x_n = b$$

Une suite de points $(\xi_i)_{1\leqslant i\leqslant n}$ de la subdivision σ est une suite finie de réels tels que :

$$\forall i \in [\![1,n]\!],\ x_{i-1} \leqslant \xi_i \leqslant x_i$$

À toute fonction f définie sur $[a,b]$ on associe la somme suivante, dite **somme de Riemann**, liée à la subdivision σ et à la suite de points (ξ_i) :

$$S = \sum_{i=1}^{i=n} f(\xi_i) \times (x_i - x_{i-1})$$

C'est la somme des aires (*algébriques*) des rectangles dont les bases sont prises entre deux points consécutifs de la subdivision σ et dont les hauteurs sont mesurées par les valeurs de f au point correspondant de (ξ_i).

Si la fonction f est continue par morceaux sur $[a,b]$ alors les sommes de Riemann sont d'autant plus proches de l'intégrale que l'écart maximal entre deux points consécutifs de la subdivision est proche de 0.

2 Calculs

2.1 Première phase

Pour simplifier, nous allons considérer un intervalle $[a,b]$ que l'on divisera en n intervalles de longueurs égales. Cela définira notre subdivision σ.

```
▷ (a:1,b:2,n:10)$
```

```
▷ s:makelist(a+(b-a)/n*i,i,1,n);
2 :      [11/10, 6/5, 13/10, 7/5, 3/2, 8/5, 17/10, 9/5, 19/10, 2]
```

Définissons une fonction sur $[a,b]$.

```
▷ f(x):=log(x);
```

Calculons la valeurs de f aux points de σ.

```
▷ v:map(f,s);
```

$$4: \quad \left[\log\left(\frac{11}{10}\right), \log\left(\frac{6}{5}\right), \log\left(\frac{13}{10}\right), \log\left(\frac{7}{5}\right), \log\left(\frac{3}{2}\right), \log\left(\frac{8}{5}\right), \log\left(\frac{17}{10}\right), \log\left(\frac{9}{5}\right), \log\left(\frac{19}{10}\right), \log 2\right]$$

Nous pouvons maintenant calculer la somme de Riemann.

```
▷ r:lsum(i*(b-a)/n,i,v);
```

$$5: \quad \frac{\log 2}{10} + \frac{\log\left(\frac{19}{10}\right)}{10} + \frac{\log\left(\frac{9}{5}\right)}{10} + \frac{\log\left(\frac{17}{10}\right)}{10} + \frac{\log\left(\frac{8}{5}\right)}{10} + \frac{\log\left(\frac{3}{2}\right)}{10} + \frac{\log\left(\frac{7}{5}\right)}{10} + \frac{\log\left(\frac{13}{10}\right)}{10} + \frac{\log\left(\frac{6}{5}\right)}{10} + \frac{\log\left(\frac{11}{10}\right)}{10}$$

Il ne reste qu'à évaluer numériquement ce nombre.

```
▷ float(r);
```

$$6: \quad 0.42053529577375$$

On peut maintenant le comparer avec la valeur approchée de l'intégrale fournie par ***Maxima***.

```
▷ valeur:float(integrate(f(x),x,a,b));
```

$$7: \quad 0.38629436111989$$

L'approximation est ce qu'elle est...

2.2 Seconde phase

Des calculs précédents nous pouvons *extraire* une procédure, ce qui va nous permettre de multiplier les calculs pour *apprécier* la convergence des sommes de Riemann vers l'intégrale lorsque n tend vers $+\infty$.

```
▷ SommeRiemann(f,a,b,n) := block(
      /* les variables locales                    */
      [s,v,r],
      /* la suite de points de la subdivision     */
      s:makelist(a+(b-a)/n*i,i,1,n),
      /* les valeurs de la fonction en ces points */
      v:map(f,s),
      /* la somme de Riemann                      */
      r:lsum(i*(b-a)/n,i,v),
      /* une évaluation comme résultat final      */
      float(r)
  );
```

Voici les sommes de Riemann avec $n = 10, 30, 50, 70$.

```
▷ t:makelist(SommeRiemann(f,1,2,10+20*n),n,0,3);
```

$$9: \quad [0.42053529577375, 0.39780052083256, 0.39320916664766, 0.39123690910947]$$

La variation est *lente*. Passons à de plus grandes valeurs de n.

```
▷ t:makelist(SommeRiemann(f,1,2,10^n),n,1,3);
```

$$10: \quad [0.42053529577375, 0.38975593038033, 0.38664089304351]$$

Finalement nous n'avons que trois décimales exactes après avoir divisé l'intervalle d'intégration en 1000 parties. Le gain est faible.

Le choix d'une progression géométrique pour les calculs précédents n'était pas fortuit, nous allons

tirer profit des valeurs calculées pour approcher l'intégrale de *beaucoup* plus près et cela à l'aide de la méthode de **Richardson**.

Nous partons de l'hypothèse que les sommes de Riemann s_n, comme elles sont calculées ici, sont développables en $\frac{1}{n}$ (n étant le nombre de divisions de l'intervalle d'intégration). Plus précisément supposons qu'il existe des réels α_1, α_2, α_3 tels que

$$s_n = \int_a^b f(x)dx + \frac{\alpha_1}{n} + \frac{\alpha_2}{n^2} + \frac{\alpha_3}{n^3} + o\left(\frac{1}{n^3}\right).$$

On voit ainsi que si l'on calcule $\frac{10s_{10n}-s_n}{9}$ nous obtiendrons une quantité s'_n qui se développera sous la forme

$$s'_n = \int_a^b f(x)dx - \frac{\alpha_2}{10n^2} - \frac{11\alpha_3}{100n^3} + o\left(\frac{1}{n^3}\right).$$

Le constat est simple : le terme d'ordre 1 du développement est éliminé (on a tout fait pour), l'approximation est donc meilleure. Et rien n'empêche de continuer ainsi, tant que l'on dispose d'un nombre de valeurs suffisantes.

```
▷ t:makelist((10*t[i+1]-t[i])/9,i,1,length(t)-1);
11 :      [0.38633600089217, 0.38629477778386]

▷ t:makelist((100*t[i+1]-t[i])/99,i,1,length(t)-1);
12 :      [0.38629436138883]

▷ abs(valeur-t);
13 :      [2.6893515192583095 × 10^-10]
```

Au final nous disposons de 9 décimales exactes et cela sans *recalculer* une seule somme !

L'hypothèse du départ se justifie donc *à posteriori*, sa démonstration dans le cas de fonctions suffisamment régulières est assez accessible.