wxMaxima pour l'analyse en mathématiques
Volume II

Zachary Hannan
Solano Community College

Traduction française : Michel Gosse

Table des matières

Préface

Logiciel de calcul formel :

Un système de calcul formel (CAS) est un ensemble de logiciels conçu principalement pour effectuer des calculs symboliques. Un CAS peut effectuer presque tous les calculs symboliques qu'une personne pourrait faire « à la main », mais il est beaucoup plus rapide, plus précis et capable de gérer une complexité plus grande. Les calculs complexes peuvent être divisés en parties gérables grâce à l'utilisation de fonctions, et les modélisations peuvent être explorées en modifiant rapidement leurs paramètres. En plus des calculs symboliques, un CAS peut produire des graphiques de qualité, faire des approximations numériques de divers types et exécuter des algorithmes pour résoudre des problèmes qui ne peuvent être résolus symboliquement.

wxMaxima :

wxMaxima est une interface utilisateur pour le système de calcul formel Maxima. Cette interface permet à l'utilisateur de créer, éditer et sauvegarder un document (un fichier .wxm ou .wxmx selon les cas) contenant de nombreux calculs et graphiques, et *la plupart* des opérations peuvent être accessibles via l'interface graphique, si souhaité. Maxima et wxMaxima sont des projets open source, ce qui signifie qu'ils seront toujours gratuits et qu'ils s'améliorent constamment grâce au travail bénévole de leurs nombreux passionnés.

La dernière version de wxMaxima pour les ordinateurs Windows et Mac peut être téléchargée ici : http://andrejv.github.io/wxmaxima/.
Lorsque vous cliquez sur le lien de téléchargement pour votre système d'exploitation, vous serez redirigé vers une page de sourceforge.net qui téléchargera automatiquement Maxima, wxMaxima, GNUplot et tout autre programme auxiliaire nécessaire pour que wxMaxima fonctionne sur votre machine. L'installation sur un ordinateur Windows prend généralement environ 5 minutes.

Versions des logiciels

Le texte du document original en anglais a été rédigé avec wxMaxima version 12.04.0 et Maxima version 5.27.0. L'utilisation de versions plus récentes du logiciel ne devrait pas poser de problèmes. Cependant, des bogues peuvent parfois survenir ; c'est pourquoi je recommande que l'enseignant et ses étudiants utilisent tous la même version, au cas où un dépannage serait nécessaire.
Lors de la traduction en français, une actualisation du document a été faite en utilisant la version 20.12 de wxMaxima et Maxima version 5.45.

Autres manuels de la série “wxMaxima pour”

J’ai publié deux livres dans la série “wxMaxima pour” :

“wxMaxima for Calculus I”	June 2015
“wxMaxima for Calculus II”	June 2015

avec le projet de réaliser des documents similaires pour l’algèbre linéaire, les équations différentielles et le calcul à plusieurs variables lors des prochaines années.

Ces livres peuvent être obtenus à l’adresse https://wxmaximafor.wordpress.com/. Les documents sont disponibles sous forme de fichiers pdf gratuits à télécharger ou bien en ”impression à la demande” pour un coût abordable. Les versions françaises de ces livres sont téléchargeables sur le site https://maxima-french-doc.fr/ à la rubrique Documentation.

Ces manuels s’adressent principalement aux étudiants de premier cycle qui suivent des cours de mathématiques destinés aux filières d’ingénieurs, de sciences physiques et de mathématiques appliquées. Les universités attendent de plus en plus que ces étudiants maîtrisent les logiciels mathématiques dès le début de leurs cycles de formation supérieure, et de nombreuses institutions proposent actuellement des laboratoires de mathématiques pour répondre à ce besoin. Chaque texte de la série **≪ wxMaxima pour ≫** peut servir de manuel de référence pour un cours de première année de licence, ou de ressource précieuse ≪ par l’exemple ≫ pour les étudiants apprenant l’informatique de manière autonome.

En se basant seulement sur une expérience minimale avec les ordinateurs (comme être à l’aise avec un système d’exploitation tel que Windows ou Mac OS), chaque texte introduit progressivement l’utilisation de l’informatique scientifique à l’aide d’exemples pertinents pour le cours de mathématiques suivi en parallèle. Les principaux points théoriques de chaque cours sont passés en revue de manière concise, et les commandes sont introduites au fur et à mesure des besoins. Les exemples servent à motiver et renforcer les concepts mathématiques importants tout en illustrant leurs applications dans le contexte de l’informatique. Les commandes textes du logiciel sont utilisées exclusivement pour deux raisons : d’une part, elles sont plus puissantes et flexibles ; d’autre part, se familiariser avec les commandes textes garantit que les compétences acquises ici seront facilement transposables à d’autres logiciels.

Mise en page du document

Chaque texte est divisé en 7 modules, chacun étant composé de plusieurs sections et sous-sections. Chaque sous-section commence généralement par une brève discussion théorique, suivie de plusieurs exemples résolus dans wxMaxima. Chaque module se termine par un court ensemble d’exercices progressant du niveau basique au niveau avancé. Les sections et exercices particulièrement difficiles sont marqués d’un astérisque.

Ce texte n’est pas conçu pour être un manuel de référence encyclopédique, mais chaque module contient une liste de **≪ Commandes clés ≫** sur la page de titre afin de faciliter la recherche d’un exemple utilisant une commande spécifique.

Pour l'étudiant

Pour les étudiants ayant très peu d'expérience avec les ordinateurs, les deux premiers modules progresseront très lentement. Faire des erreurs et déboguer vos commandes fait naturellement partie de l'apprentissage de la syntaxe d'un nouveau programme. Bien que wxMaxima tente de vous aider en cas d'erreur, votre aide la plus précieuse reste la recherche sur Internet. En recherchant un problème particulier sur Google, vous trouverez probablement une variété de forums en ligne où un problème similaire est abordé. Avec le temps, votre connaissance syntaxique de wxMaxima et votre capacité à effectuer des recherches efficaces s'amélioreront. Notez qu'il est judicieux d'inclure **≪ wxMaxima ≫** dans vos requêtes plutôt que **≪ Maxima ≫**, car ce dernier terme a de nombreuses significations autres que le logiciel de calcul formel !

Le manuel officiel de Maxima est disponible à l'adresse suivante :
https://maxima.sourceforge.io/docs/manual/maxima_toc.html, mais il convient de noter qu'il est destiné à un public ayant un haut niveau de connaissances en calcul scientifique informatique. Je consulte occasionnellement le manuel officiel, mais j'ai constaté que rechercher des exemples pertinents est la méthode la plus rapide pour apprendre.

Il est important de travailler les exemples par vous-même, qu'ils soient ou non proposés par votre enseignant. En tapant vous-même les commandes, vous ferez inévitablement des erreurs de syntaxe qu'il faudra corriger. Corriger votre syntaxe sur un exemple résolu est une excellente préparation pour réussir les exercices de manière autonome.

Pour l'enseignant

Ce contenu peut être intégré à votre cours à différents niveaux. Vous pouvez choisir de demander aux étudiants de simplement reproduire et tester tous les exemples du manuel, ou décider de ne proposer que certains exercices et laisser les étudiants utiliser le manuel comme référence en autonomie. Le niveau de difficulté peut être modéré ou très exigeant, en fonction de la quantité d'indications que vous incluez dans votre cours.

Je recommande que les étudiants soumettent leur travail par e-mail sous forme de fichier wxMaxima bien formaté (.wxmx) avec un en-tête clair et les exemples et/ou exercices clairement étiquetés. Les étudiants peuvent insérer des lignes de texte dans leur feuille de travail en sélectionnant **Cell / Insert Text Cell** ou en appuyant sur **Ctrl+1**. Lorsque vous ouvrez la feuille de travail, les commandes devront être ré-exécutées, et cela constitue un moyen rapide de vérifier que tout le code fonctionne et que les solutions attendues sont obtenues. Les étudiants ont la responsabilité de déboguer leur travail jusqu'à ce que leur code fonctionne sans erreur.

Tout commentaire ou suggestion sur ce texte sera grandement apprécié et pris en compte pour les futures éditions.

Merci beaucoup,

Zak Hannan
Enseignant de Mathématiques et de Physique
Solano Community College, Fairfield, CA
zhannan@solano.edu

Traduction du document : Michel Gosse (michel.gosse@free.fr)
En complément du document original, les corrigés des exercices sont proposés dans le fichier correction-volI.pdf, accompagnés des fichiers wxMaxima associés. Le tout est disponible sur https://maxima-french-doc.fr/ à la rubrique Documentation.

Chapitre 0

Introduction à wxMaxima

Commandes essentielles de ce Chapitre

`float`	`trigsimp`	`kill(all)`	`parametric`	`rhs`
`ratsimp`	`trigreduce`	`wxdraw2d`	`dimensions`	`makelist`
`fullratsimp`	`trigexpand`	`wxdraw3d`	`solve`	`for-do`
`expand`	`subst`	`explicit`	`find_root`	`sum`
`factor`	`sublis`	`implicit`	`lhs`	`simpsum`

0.1 Commandes de base

0.1.1 Arithmétique

wxMaxima utilise `+,-,*,/,^,sqrt,log` pour l'addition, la soustraction, la multiplication, la division, les puissances, la racine carrée et le logarithme népérien. On utilise `%` pour rappeler le résultat précédent et `float` pour obtenir une valeur approchée décimale. Pour obtenir les résultats de chaque commande, nous terminons la ligne (ou cellule) d'entrée avec `;` et appuyons simultanément sur les touches `shift+enter`. Si nous souhaitons masquer le résultat du calcul, nous devons terminer la ligne d'entrée par `$` à la place de `;`.

Exemple 0.1.1. Effectuer les opérations numériques suivantes :

1. Calculer $3 \cdot 2 + 5$.
2. Ajouter $\sqrt{2}$ au résultat précédent.
3. Trouver une valeur approchée décimale du résultat précédent.
4. Elever au carré le résultat précédent.

```
(%i1) 3*2+5;
(%o1) 11
(%i2) %+sqrt(2);
(%o2) sqrt(2)+11
(%i3) float(%);
(%o3) 12.4142135623731
(%i4) %^2;
(%o4) 154.1126983722081
```

Vous verrez que l'expression obtenue est de temps en temps "mieux affichée" que le résultat présenté dans ce document ; par exemple, `sqrt(2)+11` peut s'afficher sous la forme $\sqrt{2} + 11$. (Le menu Maxima ▶ Basculer en affichage 2d de wxMaxima gère ces choix d'affichage des résultats). Nous utiliserons le format enrichi seulement quand cela sera nécessaire pour une meilleure compréhension.

Exemple 0.1.2. Additionner $\frac{5}{6}$ and $\frac{7}{15}$ et trouver la forme simplifiée du résultat, puis donner une valeur approchée décimale de cette réponse.

```
(%i5) (5/6)+(7/15);
(%o5) 13/10
(%i6) float(%);
(%o6) 1.3
```

Remarquer que wxMaxima exprime automatiquement la fraction exacte sous forme réduite.

Exemple 0.1.3. wxMaxima utilise le symbole `%e` pour la constante d'Euler e. Trouver une approximation décimale de e et vérifier que $\ln e = 1$.

```
(%i7) float(%e);
(%o7) 2.718281828459045
(%i8) log(%e);
(%o8) 1
```

0.1.2 Calcul algébrique

wxMaxima est capable de réaliser les opérations algébriques basiques sur des expressions contenant des variables tout comme des nombres : rassembler des termes identiques, développer et factoriser, additionner/soustraire/diviser/réduire des expressions rationnelles, etc. Dans quelques cas, la simplification automatique est complète, et dans d'autres cas nous avons à imposer une simplification avec la commande `ratsimp` ou `fullratsimp` (cette dernière commande applique simplement `ratsimp` de manière récursive). Notons que une entrée comme `5x` provoquera une erreur – nous devons écrire la multiplication de manière explicite en écrivant `5*x`.

Exemple 0.1.4. Réaliser les opérations algébriques suivantes :

1. Simplifier : $(3a + b) + 2(2a - b)$

   On saisit l'expression et on applique `ratsimp` pour rassembler les termes :

```
(%i9) (3*a+b)+2*(2*a-b);
(%o9) b+2*(2*a-b)+3*a
(%i10) ratsimp(%);
(%o10) 7*a-b
```

2. Développer $(a + b)^3$

   `expand` développe l'expression :

```
(%i11) (a+b)^3;
(%o11) (b+a)^3
(%i12) expand(%);
(%o12) b^3+3*a*b^2+3*a^2*b+a^3
```

3. Factoriser : $x^2 - 8x + 12$

   On applique simplement `factor` :

```
(%i13) factor(x^2-8*x+12);
(%o13) (x-6)*(x-2)
```

4. Faire la somme et exprimer sous forme réduite et factorisée : $\frac{5}{x^2-1} + \frac{x-2}{x^2+2x-3}$

   On additionne avec `ratsimp`, puis on exprime la réponse sous forme factorisée et réduite avec `factor` :

```
(%i14) 5/(x^2-1)+(x-2)/(x^2+2*x-3);
(%o14) (x-2)/(x^2+2*x-3)+5/(x^2-1)
(%i15) ratsimp(%);
(%o15) (x^2+4*x+13)/(x^3+3*x^2-x-3)
(%i16) factor(%);
(%o16) (x^2+4*x+13)/((x-1)*(x+1)*(x+3))
```

NDT : on obtient en affichage 2d le résultat suivant :

$$\frac{x^2+4\,x+13}{(x-1)\,(x+1)\,(x+3)}$$

5. Réduire : $\frac{\frac{2x^2+xy}{y}}{\frac{xy+x}{y^2}}$

On saisit cette fraction de fractions, on la réduit en utilisant `ratsimp` et on la factorise avec `factor` :

```
(%i17) ((2*x^2+x*y)/y)/((x*y+x)/y^2);
(%o17) (y*(x*y+2*x^2))/(x*y+x)
(%i18) ratsimp(%);
(%o18) (y^2+2*x*y)/(y+1)
(%i19) factor(%);
(%o19) (y*(y+2*x))/(y+1)
```

NDT : résultat en affichage 2d : $\frac{y\,(y+2\,x)}{y+1}$

0.1.3 Trigonometrie

wxMaxima dispose de toutes les fonctions trigonométriques. Les angles doivent être exprimés en radians, aussi tout angle mesuré en degrés doit être converti en radians en le multipliant par $\frac{2\pi}{360}$. `%pi` est le symbole spécial pour le nombre π.

Exemple 0.1.5. Calculer $\sin 85°$ et $\tan \frac{11\pi}{12}$.

```
(%i20) 85*2*%pi/360;
(%o20) (17*%pi)/36
(%i21) sin(%);
(%o21) sin((17*%pi)/36)
(%i22) float(%);
(%o22) 0.99619469809175

(%i23) float(tan(11*%pi/12));
(%o23) -0.26794919243112
```

On se sert des commandes `trigsimp` pour simplifier le résultat à l'aide des égalités $\cos^2 x + \sin^2 x = 1$ et $\cosh^2 x + \sinh^2 x = 1$, `trigreduce` pour linéariser des expressions trigonométriques et `trigexpand` pour développer des fonctions trigonométriques de sommes d'angles.

Exemple 0.1.6. Aperçu des manipulations trigonométriques.

1. Appliquez `trigreduce` à $\sin^2 x + \cos^2 x$. Qu'observe-t-on ? Essayez d'utiliser `trigsimp` à la place.

```
(%i24) trigreduce((sin(x))^2+(cos(x))^2);
(%o24) (cos(2*x)+1)/2+(1-cos(2*x))/2
```

`trigreduce` transforme l'égalité en une expression bien plus complexe – nous appliquons `trigsimp` à la place :

```
(%i25) trigsimp((cos(x))^2+(sin(x))^2);
(%o25) 1
```

2. Exprimer $\sin^2 x$ en fonction du "double de l'angle".
   Cette fois `trigreduce` est la commande adéquate :

```
(%i26) trigreduce((sin(x))^2);
(%o26) (1-cos(2*x))/2
```

3. Trouver une expression pour $\cos(x+y)$ en fonction de $\sin x$ et $\cos x$.
   `trigexpand` va simplifier l'argument de la fonction cosinus :

```
(%i27) trigexpand(cos(x+y));
(%o27) cos(x)*cos(y)-sin(x)*sin(y)
```

0.2 Expressions et fonctions

L'une des fonctionnalités puissantes d'un système de calcul formel est que nous pouvons attribuer un nom à une variété d'objets, puis les « appeler » plus tard en utilisant ce nom. Nous pouvons attribuer un nom à une expression en utilisant « `:` » et attribuer un nom à une fonction en utilisant « `:=` ». Les expressions peuvent être évaluées pour des valeurs spécifiques des variables en utilisant `subst` ou `sublis`, tandis que les fonctions sont évaluées en utilisant la notation fonctionnelle ordinaire. Les expressions et les fonctions présentent divers avantages et inconvénients qui se révéleront au fur et à mesure – souvent, nous pouvons choisir l'un ou l'autre de ces avantages ou inconvénients pour résoudre un problème.

Exemple 0.2.1. Affecter A à l'expression $2x + 5$ et B à l'expression $6 - x^4$, puis calculer $A + B$, $A - B$ and AB sous forme développée.

On affecte A et B sur deux lignes consécutives avant de les faire interpréter par Maxima avec `shift+enter`.

```
(%i1) A:2*x+5$
      B:6-x^4$

(%i3) A+B;
(%o3) -x^4+2*x+11

(%i4) A-B;
(%o4) x^4+2*x-1

(%i5) A*B;
(%o5) (2*x+5)*(6-x^4)
(%i6) expand(%);
(%o6) -2*x^5-5*x^4+12*x+30
```

Exemple 0.2.2. Utiliser `subst` pour évaluer A quand $x = -3$ puis quand $x = B$.

```
(%i7) subst(-3,x,A);
(%o7) -1

(%i8) subst(B,x,A);
(%o8) 2*(6-x^4)+5
```

Exemple 0.2.3. Définir $f(x)$ comme la fonction racine carrée. Calculer $f(4)$, $f(-4)$ et $f(A)$.

```
(%i9) f(x):=sqrt(x);
(%o9) f(x):=sqrt(x)

(%i10) f(4);
(%o10) 2
```

```
(%i11) f(-4);
(%o11) 2*%i

(%i12) f(A);
(%o12) sqrt(2*x+5)
```

Nous voyons que $\sqrt{-4}$ a pour valeur $2i$. Notons que le symbole spécial désignant l'unité imaginaire i est `%i`. wxMaxima n'est pas restreint uniquement aux nombres réels !

Exemple 0.2.4. Utiliser `sublis` pour évaluer $\frac{-b+\sqrt{b^2-4ac}}{2a}$ quand $a = 1$, $b = -3$ and $c = 5$.

```
(%i9) EXPN:(-b+sqrt(b^2-4*a*c))/(2*a)$

(%i10) sublis([a=1,b=-3,c=5],EXPN);
(%o10) (sqrt(11)*%i+3)/2
```

Les fonctions peuvent avoir un nombre quelconque de variables. Quelquefois, nous utilisons une variable comme un *paramètre* afin de générer une famille de fonctions :

Exemple 0.2.5. Définir $f_n(x) = \cos(n\pi x)$, puis expliciter $f_n(x)$ pour plusieurs valeurs de n.

```
(%i13) f(n,x):=cos(n*%pi*x)$
       f(1,x);
       f(2,x);
       f(3,x);
(%o14) cos(%pi*x)
(%o15) cos(2*%pi*x)
(%o16) cos(3*%pi*x)
```

0.3 Graphiques 2D et 3D

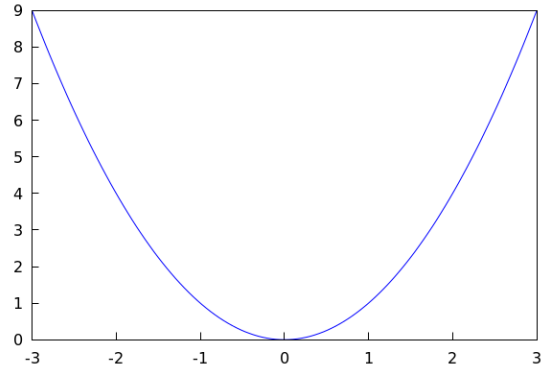
wxMaxima crée des graphiques en appelant un autre programme appelé GNUplot. Dans ce document, nous utilisons exclusivement les commandes `wxdraw2d` et `wxdraw3d` pour créer des graphiques 2D et 3D intégrés. Les commandes associées `draw2d` et `draw3d` créeront le même graphique dans une fenêtre contextuelle de GNUplot. La fenêtre contextuelle est utile pour manipuler des graphiques 3D (nous pouvons faire tourner l'image avec une souris), mais les graphiques intégrés sont plus pratiques pour imprimer notre travail. Dans les anciennes versions de wxMaxima, il peut être nécessaire de saisir `load(draw)$` avant d'utiliser les commandes `wxdraw2d` et `wxdraw3d`.

Les graphiques peuvent être modifiés avec une multitude de paramètres introduits progressivement dans le texte. Nous incluons un petit échantillon d'objets graphiques et de fonctionnalités de tracé ci-dessous.

Exemple 0.3.1. Définir $f(x) = x^2$, puis réaliser un tracé simple de $f(x)$ sur $[-3, 3]$.

Nous commençons cette section en appliquant `kill(all)` pour supprimer toutes les affectations que wxMaxima conserve actuellement en mémoire. En pratique, nous n'avons besoin d'utiliser `kill(all)` que si nos affectations précédentes provoquent une interférence avec un calcul. Nous utilisons `explicit` pour tracer la fonction, car $f(x)$ est donnée *explicitement* comme une fonction de x, puis le domaine est indiqué à côté de la fonction. Nous insérons plusieurs sauts de ligne dans `wxdraw2d` afin d'en faciliter la lecture :

```
(%i1) kill(all)$
(%i1) f(x):=x^2$
(%i2) wxdraw2d(
      explicit(f(x),x,-3,3)
      );
```



NDT : il existe une autre syntaxe pour tracer une courbe cartésienne 2D. Cette syntaxe sépare la ou les fonctions à tracer dans le premier crochet, et l'intervalle où varie x dans le deuxième crochet :

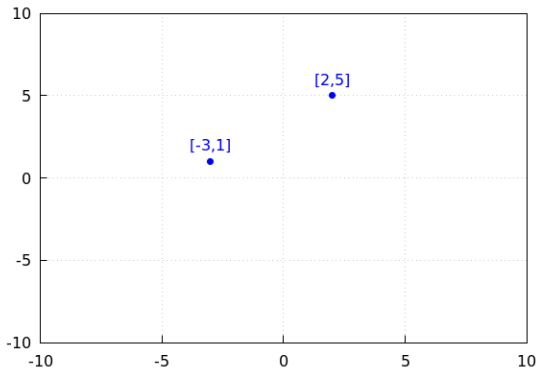
```
wxplot2d([f(x)], [x,-3,3])$
```

Il est aussi possible d'utiliser directement le menu Tracé de Courbe → Courbe 2d de wxMaxima.

Exemple 0.3.2. Dessiner les points $[-3, 1]$ et $[2, 5]$ sur un quadrillage pour x dans $[-10, 10]$ et y dans $[-10, 10]$. Rajouter une étiquette au-dessus de chaque point.

Nous utilison `point_type=7` afin de créer des disques et `points` pour lister les points désirés. `label` est utlisé pour créer les textes de chaque étiquette et les positionner auxs coordonnées voulues (dans ce cas, 1 unité au-dessus des points dessinés) :

```
(%i3) wxdraw2d(
      grid=true,
      xrange=[-10,10],
      yrange=[-10,10],
      point_type=7,
       points([[-3,1],[2,5]]),
      label(["[-3,1]",-3,2],["[2,5]",2,6])
      );
```

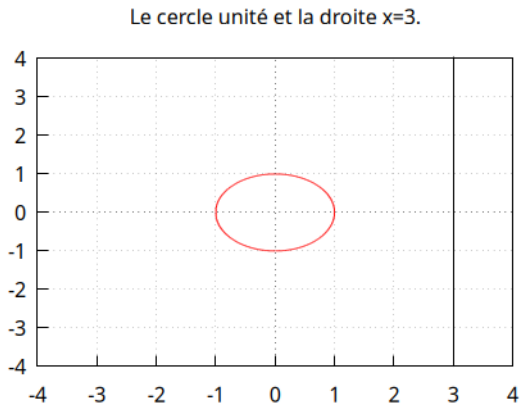


Exemple 0.3.3. Tracer une ligne verticale de couleur noire $x = 3$, et dessiner le cercle unité en rouge avec x variant sur $[-4, 4]$ et y variant sur $[-4, 4]$. Inclure les axes des x et y, un quadrillage et un titre.

Cet exemple illustre bien à quel point un graphique peut rapidement devenir complexe. Une ligne verticale n'est pas une fonction, elle doit donc être définie *paramétriquement*. Nous demandons à wxMaxima de tracer de nombreux points $(3, t)$ lorsque t varie de -4 à 4. De plus, l'équation du cercle unité définit une courbe *implicitement* : il est impossible d'exprimer y en fonction de x. Enfin, le cercle unité sera déformé si nous n'imposons pas un rapport d'aspect carré à l'aide de `dimensions` :

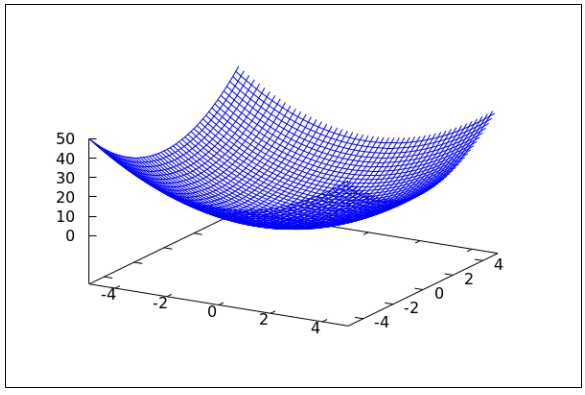
```
(%i4) wxdraw2d(
      grid=true,
      xaxis=true,
      yaxis=true,
      dimensions=[600,600],
      xrange=[-4,4],
      yrange=[-4,4],
```

```
title="Le cercle unité et la droite x=3.",
color=black,
 parametric(3,t,t,-4,4),
color=red,
 implicit(x^2+y^2=1,x,-1,1,y,-1,1)
);
```



Exemple 0.3.4. Effectuer un tracé rapide du paraboloïde $z = x^2 + y^2$ en utilisant `wxdraw3d`. En complément, utiliser `draw3d` pour représenter ce paraboloïde dans une fenêtre GNUplot, puis manipuler cette courbe avec la souris.

```
(%i5) wxdraw3d(
      explicit(x^2+y^2,x,-5,5,y,-5,5)
      );
```



NDT : On peut aussi utiliser les menus de wxMaxima (Tracé de courbes → Courbe 3d pour obtenir ce résultat. wxMaxima génère alors la commande équivalente pour le tracé du paraboloïde :

```
wxplot3d(x^2+y^2, [x,-5,5], [y,-5,5],
 [gnuplot_pm3d,true])$
```

0.4 Définir et résoudre des équations

Dans wxMaxima, le symbole "=" est réservé à la définition des équations. Une fois qu'une équation est définie, nous pouvons utiliser `rhs` et `lhs` pour isoler les membres droit et gauche. De nombreuses équations et systèmes d'équations peuvent être résolus à l'aide de `solve`, mais certaines équations ne peuvent être résolues que numériquement en utilisant `find_root` ou grâce à une autre méthode d'approximation.

Exemple 0.4.1. Attribuer l'étiquette `EQN` à l'équation $3x - 6 = 6x + 5$, puis résoudre l'équation "à la main" en utilisant des opérations algébriques afin d'isoler x. Vérifier votre réponse en remplaçant cette valeur de x dans les membres droit et gauche de l'équation de départ. Enfin, vérifier à nouveau votre réponse en utilisant `solve`.

On effectue les opérations classiques pour résoudre les équations linéaires :

```
(%i1) EQN:3*x-6=6*x+5;
(%o1) 3*x-6=6*x+5
(%i2) %+6;
(%o2) 3*x=6*x+11
(%i3) %-6*x;
(%o3) -3*x=11
(%i4) %/-3;
(%o4) x=-11/3
```

On vérifie notre réponse en utilisant `subst` :

```
(%i5) subst(-11/3,x,rhs(EQN));
(%o5) -17
(%i6) subst(-11/3,x,lhs(EQN));
(%o6) -17
```

Finalement, nous résolvons à nouveau en utilisant `solve` :

```
(%i7) solve(EQN,x);
(%o7) [x=-11/3]
```

Exemple 0.4.2. Essayez de résoudre $\ln x = \sin x$ en utilisant `solve`. Que se passe-t-il ? Reformulez ensuite le problème en termes de recherche d'une *racine* d'une autre fonction. Approximez la solution en utilisant `find_root`.

Commençons par essayer la solution classique, en appelant l'équation `EQN2` :

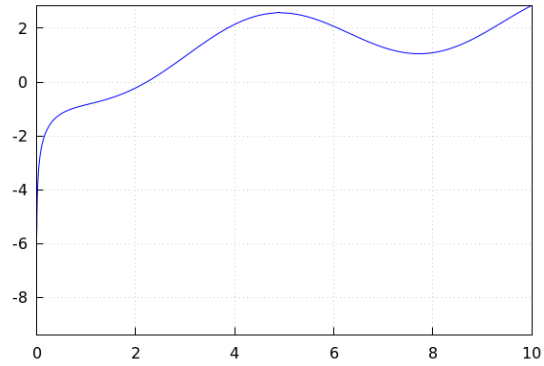
```
(%i8) EQN2:log(x)=sin(x)$
      solve(EQN2,x);
(%o9) [sin(x)=log(x)]
```

wxMaxima répète simplement la question, indiquant ainsi qu'il est en échec pour trouver la solution (en fait, `solve` ne peut résoudre que *certaines* équations polynomiales !). Cependant, nous réalisons que toute solution de $\ln x = \sin x$ est également une solution de

$\ln x - \sin x = 0$. Ainsi, nous pouvons étudier la fonction $f(x) = \ln x - \sin x$ et approcher numériquement ses racines.

Une complication de `find_root` est que nous devons indiquer l'intervalle dans lequel la racine se trouve, et la fonction doit être définie sur l'intervalle que nous avons choisi. Nous pouvons trouver un intervalle en représentant rapidement la fonction :

```
(%i10) wxdraw2d(
       grid=true,
       explicit(log(x)-sin(x),x,0,10)
       );
```



Nous observons l'existence d'une racine quelque part sur $[2, 4]$.
Remarque : si nous choisissons l'intervalle $[0, 4]$, `find_root` échouera car $\ln x$ n'est pas définie en $x = 0$!

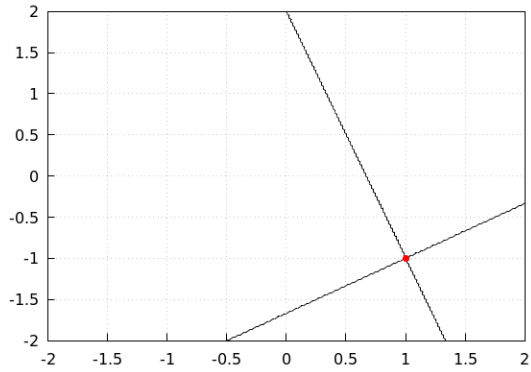
```
(%i11) find_root(log(x)-sin(x),2,4);
(%o11) 2.219107148913746
```

Nous obtenons $x \approx 2.219$ comme valeur approchée de la solution à l'équation.

Exemple 0.4.3. Résoudre le système d'équations $\begin{cases} 2x - 3y &= 5 \\ 3x + y &= 2 \end{cases}$ en utilisant `solve`.
Représentez les deux équations en définissant implicitement les fonctions correspondantes et faire apparaître le point d'intersection sur le graphique.

```
(%i12) L1:2*x-3*y=5$
       L2:3*x+y=2$
       solve([L1,L2],[x,y]);
(%o14) [[x=1,y=-1]]

(%i15) wxdraw2d(
       grid=true,
       color=black,
        implicit(L1,x,-2,2,y,-2,2),
        implicit(L2,x,-2,2,y,-2,2),
       color=red,
```

```
point_type=7,
 points([[1,-1]])
);
```



0.5 Suites et Sommes

0.5.1 Suites

Les suites trouvent une grande variété d'applications, et nous les utilisons fréquemment dans ce texte. wxMaxima génère des suites en utilisant `makelist` ou `for-do`. `makelist` offre l'avantage de permettre l'appel des éléments de la liste plus tard dans le calcul, tandis que `for-do` est beaucoup plus flexible et puissant.

Exemple 0.5.1. Utiliser `makelist` pour générer la suite $L = 1, 3, 5, \ldots 51$. Utiliser wxMaxima pour extraire le dixième élément de la suite.

On se sert de la formule $2n - 1$ avec $n = 1 \ldots 26$ pour générer la suite :

```
(%i1) L:makelist(2*n-1,n,1,26);
(%o1) [1,3,5,7,9,11,13,15,17,19,21,23,25,27,29,31,33,35,
       37,39,41,43,45,47,49,51]
```

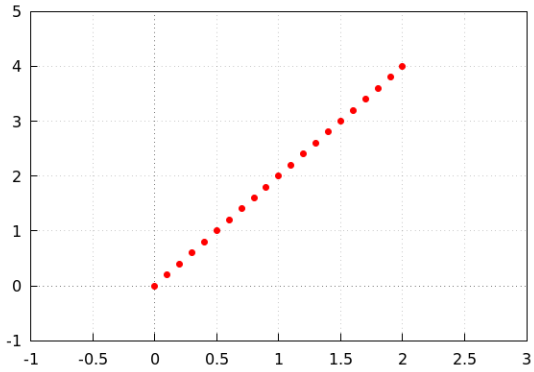
Maintenant, nous obtenons le dixième élément de `L` :

```
(%i2) L[10];
(%o2) 19
```

Exemple 0.5.2. Utiliser `makelist` pour générer une suite de 20 couples ordonnés situés sur la droite $y = 2x$ pour $x = 0.0, 0.1, \ldots, 2$. Transmettre cette liste de couples ordonnés à `wxdraw2d`.

Nous pouvons générer les valeurs de x en utilisant la suite $0.1k$ pour $k = 0, 1, \ldots, 20$. Le résultat de la commande `makelist` est déjà sous la forme d'une "liste", ce qui fait qu'il est déjà formaté pour être un argument de `wxdraw2d` :

```
(%i3) POINTS:makelist([0.1*k,2*(0.1*k)],k,0,20);
(%o3) [[0,0],[0.1,0.2],[0.2,0.4],[0.3,0.6],[0.4,0.8],[0.5,1.0],
      [0.6,1.2],[0.7,1.4],[0.8,1.6],[0.9,1.8],[1.0,2.0],[1.1,2.2],
      [1.2,2.4],[1.3,2.6],[1.4,2.8],[1.5,3.0],[1.6,3.2],[1.7,3.4],
      [1.8,3.6],[1.9,3.8],[2.0,4.0]]

(%i4) wxdraw2d(
      grid=true,
      xaxis=true,
      yaxis=true,
      xrange=[-1,3],
      yrange=[-1,5],
      point_type=7,
      color=red,
      points(POINTS)
      );
```



Exemple 0.5.3. Utiliser une boucle `for-do` pour générer la même suite que dans l'exemple 0.5.1.

Lorsqu'on programme une boucle `for-do` (également appelée simplement « do-loop »), on demande à wxMaxima de répéter un processus jusqu'à ce que le nombre final soit atteint. Dans ce cas, on demande à wxMaxima d'assigner à x la valeur $2n-1$ et d'afficher x, en répétant le calcul pour $n = 1 \ldots 26$:

```
(%i5) (for n:1 thru 26 do
       (x: 2*n-1,
        print(x))
       );
       1
       3
       5
       7
       9
       11
       13
       15
       17
       19
       21
       23
       25
       27
       29
       31
       33
       35
       37
       39
       41
       43
       45
```

```
       47
       49
       51

(%o5)  done
```

Exemple 0.5.4. La suite de Fibonacci est définie par la formule de récurrence $f_n = f_{n-1} + f_{n-2}$; c'est-à-dire que chaque terme est obtenu en additionnant les deux précédents. Si l'on commence la suite par $0, 1, \ldots$, la suite entière est donnée par $0, 1, 1, 2, 3, 5, 8, \ldots$. Utilisez une boucle do-loop pour générer les vingt premiers termes de la suite de Fibonacci.

Notre utilisation de `for-do` est plus importante dans cet exemple : nous avons à répéter un calcul plusieurs fois et à utiliser le résultat obtenu à chaque étape pour calculer l'étape suivante. Nous définissons les deux termes initiaux f_{n-1} and f_{n-2} en premier, puis la boucle calcule le terme suivant de la suite de Fibonnaci X, remplace par le terme d'ordre $(n-1)$ le terme d'ordre $(n-2)$ et assigne le terme d'ordre $(n-1)$ à X. Enfin, le processus est répété 18 fois pour obtenir la totalité des 20 termes de la suite de Fibonacci.

```
(%i6) N_1:1$
      N_2:0$
(%i8)
      (for i:1 thru 18 do
       (X:N_2+N_1,
        print(X),
        N_2:N_1,
        N_1:X)
       );
       1
       2
       3
       5
       8
       13
       21
       34
       55
       89
       144
       233
       377
       610
       987
       1597
       2584
       4181

(%o8)  done
```

0.5.2 Sommes

wxMaxima calcule les sommes en utilisant `sum`. La syntaxe est très similaire à celle de la notation sigma :

Exemple 0.5.5. Utiliser `sum` pour calculer la somme $2 + 4 + 6 + \cdots + 50$.

Avec la notation sigma, la somme s'écrit $\sum_{n=1}^{25} 2n$, et les paramètres de `sum` se réfèrent simplement aux indices de cette notation :

```
(%i9) sum(2*n,n,1,25);
(%o9) 650
```

Exemple 0.5.6. Trouver une formule algébrique donnant la somme $1 + 2 + \cdots + n$, puis utiliser la substitution pour obtenir la somme des 100 premiers entiers naturels.

En utilisant la notation sigma, nous souhaitons calculer $\sum_{k=1}^{n} k$. La formule classique est obtenue en utilisant `sum` suivi de `simpsum`, puis nous effectuons la substitution $n = 100$:

```
(%i10) sum(k,k,1,n),simpsum;
(%o10) (n^2+n)/2
(%i11) subst(100,n,%);
(%o11) 5050
```

0.6 Application : Droite passant par deux points

Pour démontrer l'utilité du calcul symbolique, nous créons une fonction pour représenter rapidement la droite reliant deux points.

Exemple 0.6.1. Créer une fonction 'LINE(a,b,c,d,x)' qui calcule l'équation d'une droite passant par les points (a, b) et (c, d). Complète ce code de manière à ce que l'affectation simple de a, b, c, d produise directement un tracé de qualité de la droite ainsi que des deux points donnés. Appliquer le code aux points $(-0.51, -3.47)$ et $(7.12, 3.94)$.

En premier, nous définissons chaque point comme une fonction de deux variables. Le résultat de chaque fonction est bienn sous une forme adéquate pour être interprété comme un point par `wxdraw2d`. Puis nous calculons la pente entre les points comme une fonction des quatre variables précédemment définies :

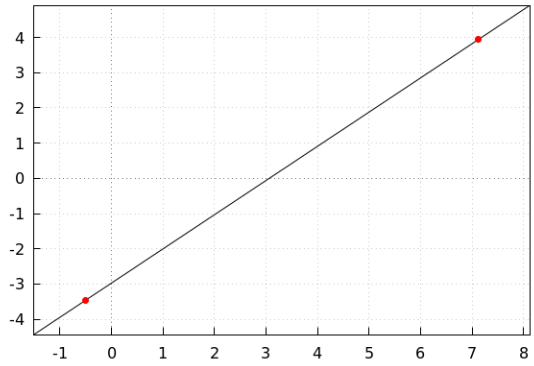
```
(%i1) POINT1(a,b):=[a,b]$
      POINT2(c,d):=[c,d]$
      PENTE(a,b,c,d):=(d-b)/(c-a)$
```

L'étape suivante consiste à remplacer dans la formule de la pente d'une droite $y - y_0 = m(x - x_0)$ et à exprimer y comme fontion de x. Nous utilisons `POINT1` pour (x_0, y_0).

```
(%i4) LINE(a,b,c,d,x):=PENTE(a,b,c,d)*(x-a)+b$
```

Finalement, nous assignons les valeurs pour a, b, c, d et paramétrons `wxdraw2d`. Notez que toutes les commandes restent générales à l'intérieur de `wxdraw2d`, ainsi nous pouvons dessiner la droite entre deux points quelconques de manière immédiate en assignant de nouvelles valeurs à a, b, c, d :

```
(%i5) a:-.51$
      b:-3.47$
      c:7.12$
      d:3.94$

      wxdraw2d(
      grid=true,
      xaxis=true,
      yaxis=true,
      color=black,
       explicit(LINE(a,b,c,d,x),x,a-1,c+1),
      color=red,
      point_type=7,
      points([[a,b],[c,d]])
      );
```


4
3
2
1
0
-1
-2
-3
-4
-1
0
1
2
3
4
5
6
7
8

0.7 Exercices du Chapitre 0

1. Définir les variables $A = \sqrt{3}$ et $B = 5$, puis trouver une approximation numérique pour $A + B$ et A/B.
2. Définir les expressions $A = x^2$ and $B = e^x$. Dans l'expression de A, remplacer x par B, puis évaluer l'expression obtenue pour $x = 0.1$ et en donner une approximation décimale.
3. Utiliser `trigexpand` pour trouver une formule donnant $\sin(x + y)$ en fonction de $\sin x$, $\sin y$, $\cos x$ et $\cos y$. Utiliser ce résultat pour calculer $\sin \frac{5\pi}{12}$ en se servant du fait que $\frac{5\pi}{12} = \frac{\pi}{6} + \frac{\pi}{4}$. Re-calculer $\sin \frac{5\pi}{12}$ directement et utiliser `float` afin de vérifier votre réponse.
4. Additionner et simplifier : $\frac{2x^2-x-6}{x^2-9} + \frac{x^3-x^2-4x+4}{x^2+5x+6}$. Exprimer la réponse sous forme factorisée.
5. Résoudre l'équation $ax^2 + bx + c = 0$ en x.
6. Représenter graphiquement $f(x) = \sin(\ln x) - 0.2$ sur $[10, 30]$ en incluant l'axe des abscisses. Utiliser `find_root` pour trouver une valeur approchée de la solution de $\sin(\ln x) = 0.2$ sur cet intervalle. Vérifier la réponse en calculant $\sin(\ln x)$ à la valeur de x trouvée.
7. Définir $f(x) = x + 2$ et $g(x) = x^2$. Trouver les intersections de ces deux courbes, puis représenter graphiquement les deux fonctions en dessinant les points d'intersection comme des disques. Nommer chaque point d'intersection en utilisant ses coordonnées décimales arrondies à la deuxième décimale.
8. Utiliser `makelist` et `for-do` pour générer les trente premiers termes de la suite $1, \frac{1}{2}, \frac{1}{4}, \ldots$.
9. La formule de récurrence d'une suite est donnée par $f_n = 2f_{n-1}$ avec comme premier terme $f_0 = 3$. Utilisez une boucle 'do' pour générer récursivement les 10 premiers termes de cette suite (comme dans l'exemple 0.5.4). Une fois la suite écrite, essayez de deviner une formule explicite pour f_n. Une fois cette formule trouvée, utilisez `makelist` pour générer la même suite.
10. Utilisez `makelist` pour tracer 40 cercles centrés à l'origine avec des rayons $0.1, 0.2, 0.3, \ldots$ dans un seul graphique. Ce problème est délicat, car cette liste doit produire des éléments que `wxdraw2d` sait interpréter : `implicit` et ses arguments appropriés doivent être inclus dans chaque élément de la liste !
11. Toute parabole peut s'écrire sous la forme $f(x) = ax^2 + bx + c$. Les paramètres a, b, c définissent complètement la parabole. Si l'on vous donne trois points appartenant à une parabole inconnue, ils génèrent un système de trois équations en a, b, c. La commande `solve` permet de résoudre rapidement ce système. Écrivez une solution similaire à l'exemple 0.6.1 pour définir trois points quelconques et tracer à la fois ces points et la parabole qui passe par ces trois points. La syntaxe correcte pour résoudre un système d'équations est : `solve([eqn1,eqn2,eqn3],[var1,var2,var3])`. Appliquer ce programme aux points $(-6.8, -5.5)$, $(0.1, 6.7)$ et $(3.2, -0.9)$.

Chapitre 1

Techniques classiques pour l'intégration

Commandes essentielles de ce Chapitre

`integrate`	`diff`	`ratprint`	`sublis`	`trigexpand`
`float`	`kill(all)`	`logcontract`	`divide`	`dimensions`
`filled_func`	`solve`	`factor`	`partfrac`	`declare`
`wxdraw2d`	`coeff`	`denom`	`limit`	
`subst`	`trigsimp`	`ratsimp`	`trigreduce`	

1.1 Rappel rapide sur l'intégration

1.1.1 Intégrales définies

La notion d'intégration provient du *problème de l'aire* : le problème du calcul de l'aire algébrique délimitée par une fonction $f(x)$ sur un intervalle $[a, b]$. Le problème de l'aire est généralement introduit en divisant l'intervalle $[a, b]$ en de nombreux sous-intervalles et en approximant chaque "tranche" d'aire par un rectangle. L'approximation résultante est appelée *somme de Riemann* (dans le module suivant, nous explorerons les approximations numériques de manière plus approfondie). L'aire *exacte* délimitée sur $[a, b]$ peut être définie comme une limite d'une somme de Riemann lorsque les "tranches" rectangulaires deviennent arbitrairement fines, et nous disons que l'aire est donnée par l'**intégrale définie** :

$$A = \int_a^b f(x)\ \mathrm{d}x$$

wxMaxima calcule les intégrales définies à l'aide de la commande `integrate`.

Exemple 1.1.1. Calculez $\displaystyle\int_{-1}^{2} \frac{1}{1+x^2}\ \mathrm{d}x$ et réalisez un graphique de $f(x)$ en colorant l'aire que vous avez calculée.

Nous définissons $f(x)$ comme une fonction, appliquons la commande `integrate`, puis utilisons `float` afin d'obtenir une approximation numérique :

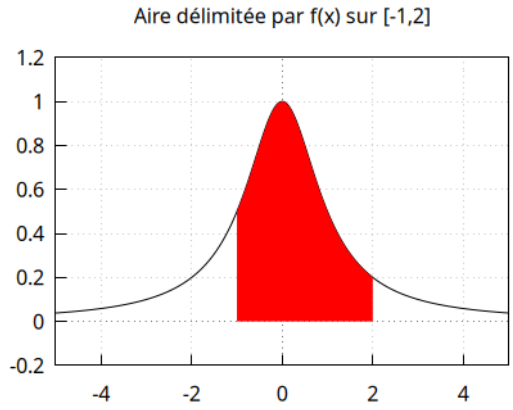
```
(%i1) f(x):=1/(1+x^2)$
(%i2) integrate(f(x),x,-1,2);
(%o2) atan(2)+%pi/4
(%i3) float(%);
(%o3) 1.892546881191539
```

Nous voyons que $\displaystyle\int_{-1}^{2} \frac{1}{1+x^2}\ \mathrm{d}x = \tan^{-1}(2) + \frac{\pi}{4} \approx 1.89.$

Pour produire le graphique incluant l'aire colorée, nous devons utiliser `filled_func`, qui attend que l'on définisse l'aire *entre* deux courbes (nous utilisons $y = 0$ comme seconde courbe). Nous utilisons plusieurs options dans la commande `wxdraw2d` afin produire un graphique soigné :

```
(%i4) load(draw)$
(%i5) wxdraw2d(
      grid=true,
      xaxis=true,
      yaxis=true,
      xrange=[-5,5],
      yrange=[-.2,1.2],
      title="Aire délimitée par f(x) sur [-1,2]",
      color=black,
       explicit(f(x),x,-5,5),
      filled_func=true,
       filled_func=f(x),
```

```
 explicit(0,x,-1,2),
filled_func=false
);
```



1.1.2 Fonctions d'aire

Une **fonction d'aire** est une intégrale définie dont une borne d'intégration est variable. Par exemple, pour trouver l'aire délimitée par f sur $[a, x]$, on écrit :

$$A(x) = \int_a^x f(t)\,\mathrm{d}t$$

où t est considérée comme « variable muette », puisque x est déjà utilisé pour désigner l'extrémité de l'intervalle d'intégration.

Exemple 1.1.2. Définissez la fonction d'aire $A(x) = \int_1^x \sin x\ \mathrm{d}x$. Représentez les aires correspondant à $A(\frac{\pi}{2})$ et $A(\pi)$. Utilisez la différence entre ces fonctions d'aire pour calculer l'aire définie par $f(x) = \sin x$ sur $[\frac{\pi}{2}, \pi]$, puis vérifiez votre réponse en vous servant directement de la commande `integrate`.

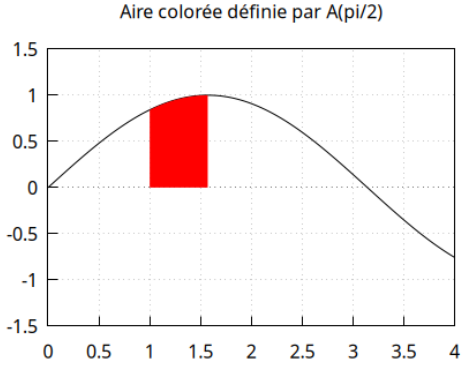
En premier, nous définissons $A(x)$ en sous servant de la variable muette t :

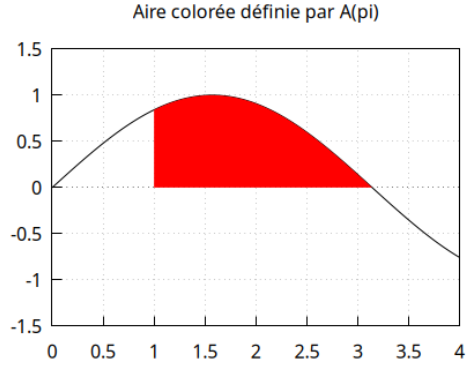
```
(%i6) A(x):=integrate(sin(t),t,1,x);
```

(%o6) $A(x) := \int_1^x \sin t\ \mathrm{d}t$

Nous pouvons maintenant tracer les zones colorées. $A\left(\frac{\pi}{2}\right) = \int_1^{\frac{\pi}{2}} \sin t\,\mathrm{d}t$, il suffit donc de hachurer la région délimitée par $\sin t$ sur l'intervalle $[1, \frac{\pi}{2}]$. De même, la zone colorée correspondant à $A(\pi)$ est simplement la région délimitée sur $[1, \pi]$:

```
(%i7) wxdraw2d(
      grid=true,
      xaxis=true,
      yaxis=true,
      xrange=[0,4],
      yrange=[-1.5,1.5],
      title="Aire colorée définie par A(pi/2)",
      color=black,
       explicit(sin(x),x,0,4),
      filled_func=true,
       filled_func=sin(x),
       explicit(0,x,1,%pi/2),
      filled_func=false
      );

(%i8) wxdraw2d(
      grid=true,
      xaxis=true,
      yaxis=true,
      xrange=[0,4],
      yrange=[-1.5,1.5],
      title="Aire colorée définie par A(pi)",
      color=black,
       explicit(sin(x),x,0,4),
      filled_func=true,
       filled_func=sin(x),
       explicit(0,x,1,%pi),
      filled_func=false
      );
```





Il est clair, d'après les graphiques, que l'aire délimitée sur $[\frac{\pi}{2}, \pi]$ n'est autre que la différence entre les deux aires $A(\pi) - A(\frac{\pi}{2})$. Il apparaît également que le « point de départ » $x = 1$ ne change rien au résultat. Enfin, nous allons calculer l'aire de deux manières différentes :

```
(%i9) A(%pi)-A(%pi/2);
```

```
(%o9) 1

(%i10) integrate(sin(x),x,%pi/2,%pi);
(%o10) 1
```

1.1.3 Le théorème fondamental de l'analyse

Le théorème fondamental du calcul intégral (TFC) trouve son origine géométrique dans l'idée qu'une « fine tranche » d'aire bornée sur l'intervalle $[x, x+h]$ peut être calculée de deux manières différentes : soit comme une approximation de l'aire d'un rectangle $f(x) \cdot h$, soit comme une différence de fonctions d'aire $A(x+h) - A(x)$. En égalant ces deux représentations de l'aire et en prenant la limite lorsque $h \to 0$, on obtient :

$$f(x) \cdot h = A(x+h) - A(x) \implies f(x) = \lim_{h \to 0} \frac{A(x+h) - A(x)}{h} = A'(x)$$

En d'autres termes, la fonction d'aire est la *primitive* de la courbe qui délimite cette aire.

Le théorème fondamental du calcul intégral (TFC) nous indique que l'on peut calculer l'intégrale définie $\int_a^b f(x)\,\mathrm{d}x$ en devinant une primitive de $f(x)$ (une fonction d'aire $A(x)$) et en l'évaluant aux bornes de l'intervalle, ce qui donne $A(b) - A(a)$. On n'a pas à se soucier de la constante arbitraire présente dans la primitive, puisqu'elle s'annule lors du calcul de la différence.
Remarquons que l'**intégrale indéfinie** $\int f(x)\,\mathrm{d}x$ est un synonyme d'une primitive de $f(x)$, et que wxMaxima calcule les primitives à l'aide de la commande `integrate` sans bornes d'intégration.

Exemple 1.1.3. Utilisez le TFC pour calculer l'aire $\int_{\frac{\pi}{2}}^{\pi} \sin x\,\mathrm{d}x$ de l'exemple 1.1.2.

Nous utilisons `integrate` pour calculer rapidement une primitive, $F(x)$. Remarquons que `''(%)` est nécessaire pour affecter une *fonction* à un résultat précédent.

```
(%i11) f(x):=sin(x)$
       integrate(f(x),x);
(%o12) -cos(x)
(%i13) F(x):=''(%);
(%o13) F(x):=-cos(x)
(%i14) F(%pi)-F(%pi/2);
(%o14) 1
```

Nous obtenons le même résultat que celui trouvé en utilisant une différence de fonctions d'aire dans l'exemple 1.1.2.

Exemple 1.1.4. Calculez $\int_1^3 \frac{1}{x^2}\,\mathrm{d}x$ en trouvant une primitive de $\frac{1}{x^2}$ et en l'évaluant aux bornes de l'intervalle d'intégration. Vérifiez votre réponse en calculant directement l'intégrale définie.

Pour illustrer une approche légèrement différente du problème, nous définissons la primitive comme une expression et utilisons `subst` pour l'évaluer aux bornes.

```
(%i15) A:integrate(1/x^2,x);
(%o15) -1/x
(%i16) subst(3,x,A)-subst(1,x,A);
```

```
(%o16) 2/3

(%i17) integrate(1/x^2,x,1,3);
(%o17) 2/3
```

1.2 Transformation d'intégrales par changement de variable

1.2.1 Changement de variable avec *u*

Pour effectuer un changement de variable avec u, nous définissons la variable u en fonction de x (et du en fonction de dx), de manière à obtenir

$$\int f(x)\ \mathrm{d}x = \int g(u)\ \mathrm{d}u$$

On comprend qu'il est plus facile de deviner la primitive $G(u)$. Une fois $G(u)$ déterminée, on utilise la définition de u pour obtenir une primitive de $F(x)$. Le changement de variable avec u peut également s'appliquer aux intégrales définies ; dans ce cas, on peut soit évaluer $F(x)$ aux bornes d'intégration initiales, soit transformer les bornes d'intégration en fonction de u.

Bien que toute technique de substitution soit en réalité conçue pour une intégration « à la main », le processus peut beaucoup nous apprendre sur la manipulation symbolique dans wxMaxima. Pour effectuer un changement de variable avec u, nous :

1. Choisir un changement de variable et utiliser `diff` afin de calculer la différentielle du (appelée `del(u)`) dans wxMaxima), puis exprimer dx en fonction de du à l'aide la commande `solve`.
2. Extraire l'équation qui en découle avec la commande `%[1]` et remplacer `del(x)` par son expression en fonction de `del(u)` à l'intérieur de l'intégrande.
3. Utiliser `subst` pour transformer la totalité de l'intégrande et l'exprimer en fonction de u, puis calculer l'intégrale, en se rappelant que `integrate` n'attend uniquement que le *coefficient* de `del(u)`.
4. Substituer la définition de u en fonction de x dans la primitive obtenue. De manière alternative, dans une intégrale définie, nous pouvons choisir de transformer les bornes d'intégration en fonction de u avant l'évaluation.

Exemple 1.2.1. Utiliser le changement de variable $u = x^2$ pour calculer $\int 5x \cdot \sin(x^2)\ \mathrm{d}x$. Vérifier la réponse en se servant de `diff`.

```
(%i1) INTEGRAND:(5*x)*sin(x^2)*del(x)$

(%i2) solve(diff(u)=diff(x^2),del(x));
(%o2) [del(x)=del(u)/(2*x)]
(%i3) %[1];
(%o3) del(x)=del(u)/(2*x)

(%i4) subst(rhs(%),del(x),INTEGRAND);
(%o4) (5*sin(x^2)*del(u))/2
(%i5) subst(u,x^2,%);
(%o5) (5*sin(u)*del(u))/2
```

L'intégrale est maintenant entièrememement exprimée en fonction de u et s'écrit $\frac{5}{2}\int \sin u \, \mathrm{d}u$, ce qui rend suffisamment facile son calcul. Pour être complet, nous utilisons wxMaxima pour calculer l'intégrale, puis nous transformons le résultat en une fonction de la variable originale, x :

```
(%i6) integrate(coeff(%,del(u)),u);
(%o6) -(5*cos(u))/2

(%i7) subst(x^2,u,%);
(%o7) -(5*cos(x^2))/2
```

On conclut que $\int 5x \cdot \sin\left(x^2\right) \, \mathrm{d}x = -\frac{5}{2}\cos\left(x^2\right) + C$. Vérifions la réponse avec `diff` :

```
(%i8) diff(%,x);
(%o8) 5*x*sin(x^2)
```

Nous voyons que la règle de la chaine génère le facteur x nécessaire, et que la constante multiplicative est bien égale à la valeur attendue 5.

Exemple 1.2.2. Calculer $\int_{-1}^{0} \frac{x}{\sqrt{1-x}} \, \mathrm{d}x$ en utilisant le changement de variable $u = 1 - x$.

En premier, nous exprimons l'intégrale indéfinie en fonction de u :

```
(%i9) kill(all)$
(%i1) INTEGRAND:(x/(sqrt(1-x)))*del(x)$
(%i2) solve(diff(u)=diff(1-x),del(x));
(%o2) [del(x)=-del(u)]
(%i3) %[1];
(%o3) del(x)=-del(u)
(%i4) subst(rhs(%),del(x),INTEGRAND);
(%o4) -(x*del(u))/sqrt(1-x)
(%i5) solve(u=1-x,x);
(%o5) [x=1-u]
(%i6) subst(rhs(%[1]),x,(%o4));
(%o6) -((1-u)*del(u))/sqrt(u)
```

L'intégrande $-\frac{1-u}{\sqrt{u}} \, \mathrm{d}u$ est simple à intégrer en utilisant la « règle des puissances » standard pour les primitives ; autrement dit, la substitution en u conduit naturellement à une primitive que l'on peut "deviner" facilement. Pour plus de commodité, nous utilisons wxMaxima pour trouver une primitive, que nous noterons $G(u)$:

```
(%i7) integrate(coeff(%,del(u)),u);
(%o7) (2*u^(3/2)-6*sqrt(u))/3
(%i8) G(u):=''(%);
(%o8) G(u):=(2*u^(3/2)-6*sqrt(u))/3
```

Nous terminons de calculer l'intégrale définie en exprimant cette primitive de nouveau en fonction de x puis en évaluant la différence de cette primitive prises aux bornes de départ de l'intégrale.

```
(%i9) G(1-x);
(%o9) (2*(1-x)^(3/2)-6*sqrt(1-x))/3
(%i10) F(x):=''(%);
(%o10) F(x):=(2*(1-x)^(3/2)-6*sqrt(1-x))/3
(%i11) F(0)-F(-1);
(%o11) 2^(3/2)/3-4/3
```

De manière alternative, nous pouvons transformer les bornes de l'intégrale en utilisant la variable u :

```
(%i12) u(x):=1-x$
       u_lower:u(-1);
       u_upper:u(0);
(%o13) 2
(%o14) 1
(%i15) G(u_upper)-G(u_lower);
(%o15) 2^(3/2)/3-4/3
```

1.2.2 Substitutions trigonométriques

Une substitution trigonométrique est utilisée lorsqu'on reconnaît, dans l'intégrande, une expression difficile à traiter qui peut être simplifiée selon l'une des identités pythagoriciennes : $\sin^2 x + \cos^2 x = 1$ ou $\tan^2 x + 1 = \sec^2 x$.

Comme les changements de variable en u, les substitutions trigonométriques sont conçues pour des calculs effectués à la main, mais réaliser une substitution trigonométrique avec wxMaxima reste néanmoins instructif.

Exemple 1.2.3. Calculer $\displaystyle\int_{-0.5}^{0.5} \frac{1}{\sqrt{1-x^2}}\,\mathrm{d}x$ à l'aide d'une substitution trigonométrique.

Nous choisissons la substitution $x = \sin t$ afin de tirer parti du fait que $1 - \sin^2 t = \cos^2 t$. Remarquons que la substitution trigonométrique s'accompagne d'un domaine de validité : pour obtenir de manière unique toutes les valeurs possibles de $\sin t$, nous travaillons donc sur le domaine $[-\frac{\pi}{2}, \frac{\pi}{2}]$. Cette restriction de domaine ne fait perdre aucune généralité, car l'intégrande n'est défini que pour les valeurs de x appartenant à $]-1, 1[$, et toutes ces valeurs sont bien atteintes par $\sin t$ sur $[-\frac{\pi}{2}, \frac{\pi}{2}]$.

```
(%i16) kill(all)$
(%i1) INTEGRAND:(1/sqrt(1-x^2))*del(x);
(%o1) del(x)/sqrt(1-x^2)
(%i2) solve(diff(x)=diff(sin(t)),del(x));
(%o2) [del(x)=cos(t)*del(t)]
(%i3) subst(rhs(%[1]),del(x),INTEGRAND);
```

```
(%o3) (cos(t)*del(t))/sqrt(1-x^2)
(%i4) subst(sin(t),x,%);
(%o4) (cos(t)*del(t))/sqrt(1-sin(t)^2)
(%i5) trigsimp(%);
(%o5) (cos(t)*del(t))/abs(cos(t))
```

Nous devons intervenir manuellement, car wxMaxima ne sait pas calculer d'intégrales contenant des valeurs absolues. Avec t restreint à $[-\frac{\pi}{2}, \frac{\pi}{2}]$, $\cos t$ est toujours positif, donc $|\cos t| = \cos t$. L'intégrande se simplifie ainsi en $1 \cdot \mathrm{d}t$. L'intégrale s'évalue alors à $G(t) = t$, et nous remplaçons l'expression de t en fonction de x afin d'obtenir la primitive $F(x)$:

```
(%i6) G:integrate(1,t);
(%o6) t
(%i7) solve(x=sin(t),t);
      solve: using arc-trig functions to get a solution.
      Some solutions will be lost.
(%o7) [t=asin(x)]
(%i8) subst(rhs(%[1]),t,G);
(%o8) asin(x)
(%i9) F(x):=''(%)$
```

Nous ignorons l'avertissement émis par `solve`, car t appartient au domaine standard $[-\frac{\pi}{2}, \frac{\pi}{2}]$. Nous terminons en évaluant $F(x)$ aux bornes de l'intégrale :

```
(%i10) F(.5)-F(-.5);
(%o10) 1.047197551196598
```

Nous masquons une série d'avertissements à l'aide de `ratprint`, puis vérifions notre résultat avec `integrate` :

```
(%i11) ratprint:false$
(%i12) integrate(1/sqrt(1-x^2),x,-.5,.5);
(%o12) 1.047197551196598
```

Exemple 1.2.4. Calculer $\displaystyle\int \frac{1}{x\sqrt{3+x^2}}\,\mathrm{d}x$ à l'aide d'une substitution trigonométrique.

La racine carrée contient une expression qui est *proche* de $1 + \tan^2 x$, mais nous avons besoin que la substitution de x^2 produise un facteur 3 afin qu'il puisse être mis en facteur hors de la racine. Nous choisissons donc la substitution $x = \sqrt{3}\tan t$. Encore une fois, nous travaillons sur un domaine implicite $[-\frac{\pi}{2}, \frac{\pi}{2}]$ pour t, ce qui correspond cette fois à un domaine $[-\infty, \infty]$ pour $x = \sqrt{3}\tan t$.

```
(%i13) kill(all)$
(%i1) INTEGRAND:(1/(x*sqrt(3+x^2)))*del(x);
(%o1) del(x)/(x*sqrt(x^2+3))
(%i2) solve(diff(x)=diff(sqrt(3)*tan(t)),del(x));
(%o2) [del(x)=sqrt(3)*sec(t)^2*del(t)]
(%i3) subst(rhs(%[1]),del(x),INTEGRAND);
(%o3) (sqrt(3)*sec(t)^2*del(t))/(x*sqrt(x^2+3))
```

```
(%i4) subst(sqrt(3)*tan(t),x,%);
(%o4) (sec(t)^2*del(t))/(tan(t)*sqrt(3*tan(t)^2+3))
(%i5) trigsimp(%);
(%o5) (abs(cos(t))*del(t))/(sqrt(3)*cos(t)*sin(t))
```

Encore une fois, nous devons intervenir manuellement pour simplifier le facteur $\cos t$, car wxMaxima ne reconnaît pas que $\cos t$ est toujours positif sur le domaine de définition implicite.

```
(%i6) G:integrate(1/(sqrt(3)*sin(t)),t);
(%o6) (log(cos(t)-1)/2-log(cos(t)+1)/2)/sqrt(3)
```

Enfin, nous remplaçons t dans la dernière expression afin d'obtenir la primitive en fonction de x. Remarquez que l'avertissement de `solve` peut, une fois encore, être ignoré, car nous travaillons implicitement sur le domaine restreint standard de la fonction tangente :

```
(%i7) solve(x=sqrt(3)*tan(t),t);
      solve: using arc-trig functions to get a solution.
      Some solutions will be lost.
(%o7) [t=atan(x/sqrt(3))]
(%i8) subst(rhs(%[1]),t,G);
(%o8) (log(1/sqrt(x^2/3+1)-1)/2-log(1/sqrt(x^2/3+1)+1)/2)/sqrt(3)
(%i9) logcontract(%);
```

$$\text{(\%o9)} \qquad \frac{\log\left(-\frac{\sqrt{x^2+3}-\sqrt{3}}{\sqrt{x^2+3}+\sqrt{3}}\right)}{2\cdot\sqrt{3}}$$

Ce résultat concorde avec les résultats classiques d'intégration, à un signe moins près dans l'argument de `log` (wxMaxima ignore généralement les valeurs absolues dans ce type de primitive).

1.3 D'autres techniques d'intégration

1.3.1 Intégration par parties

La méthode d'intégration par parties est une technique « papier/crayon » utilisée pour intégrer un produit de deux fonctions. Une présentation abrégée de la formule est présentée ci-dessous, en partant de la règle du produit pour les dérivées :

$$(uv)' = u'v + uv'$$

$$\implies uv' = (uv)' - u'v$$

$$\implies u \cdot \mathrm{d}v = (uv)' \cdot \mathrm{d}x - v \cdot \mathrm{d}u$$

$$\implies \int u \ \mathrm{d}v = uv - \int v \ \mathrm{d}u$$

La version pour l'intégrale définie est abordée dans les exercices. Lorsque nous calculons une intégrale à l'aide de la méthode d'intégration par parties, nous devons choisir la fonction u et la différentielle $\mathrm{d}v$ de manière à ce que $\mathrm{d}u$ soit *plus simple* que u, et que v soit relativement facile à déterminer à partir de $\mathrm{d}v$.

Exemple 1.3.1. Calculer $\int x \cdot \sin x \ \mathrm{d}x$ à l'aide d'une intégration par parties. Vérifiez votre résultat en utilisant `diff`.

Nous choisissons $u = x$ ($\mathrm{d}u$ est de manière évidente $\mathrm{d}x$) et $dv = \sin x \cdot \mathrm{d}x$, puis nous trouvons v dans wxMaxima en évaluant $\int \mathrm{d}v$:

```
(%i1) u:x;
      v:integrate(sin(x),x);
(%o1) x
(%o2) -cos(x)
(%i3) u*v-integrate(v,x);
(%o3) sin(x)-x*cos(x)
```

Et finalement, nous vérifions notre réponse avec `diff` :

```
(%i4) diff(%,x);
(%o4) x*sin(x)
```

Exemple 1.3.2. Calculer $\int x^2 \cdot \sin x \ \mathrm{d}x$ à l'aide de deux intégrations par parties. Vérifiez votre réponse en utilisant `diff`.

Nous choisissons $u_1 = x^2$ et $\mathrm{d}v_1 = \sin x \cdot \mathrm{d}x$:

```
(%i5) u1:x^2$
      diff(u1);
      v1:integrate(sin(x),x);
```

```
(%o6) 2*x*del(x)
(%o7) -cos(x)

(%i8) TERM1:u1*v1;
      INTEGRAND1:v1*diff(u1);
(%o8) -x^2*cos(x)
(%o9) -2*x*cos(x)*del(x)
```

La première intégration par parties est réalisée, ce qui nous donne $-x^2 \cdot \cos x - \int -2x \cdot \cos x \ \mathrm{d}x$. Maintenant, nous choisissons $u_2 = -2x$ et $dv_2 = \cos x \cdot \mathrm{d}x$ pour réaliser la deuxième intégration par parties :

```
(%i10) u2:-2*x$
       diff(u2);
       v2:integrate(cos(x),x);
(%o11) -2*del(x)
(%o12) sin(x)

(%i13) TERM2:u2*v2;
       INTEGRAND2:v2*diff(u2);
(%o13) -2*x*sin(x)
(%o14) -2*sin(x)*del(x)
```

Cette deuxième intégration par parties nous donne $-2x \cdot \sin x - \int -2 \sin x \ \mathrm{d}x$.
Finalement, nous assemblons les résultats pour obtenir la réponse finale :
$u_1 \cdot v_1 - \left(u_2 \cdot v_2 - \int v_2 \ \mathrm{d}u_2 \right)$ et nous la vérifions avec `diff` :

```
(%i15) TERM1-(TERM2-integrate(coeff(INTEGRAND2,del(x)),x));
(%o15) 2*x*sin(x)-x^2*cos(x)+2*cos(x)

(%i16) diff(%,x);
(%o16) x^2*sin(x)
```

1.3.2 Décomposition en fractions partielles

La décomposition en fractions partielles est utilisée pour scinder un intégrande rationnel en plusieurs termes plus simples, dont chacun possède une primitive facile à déterminer. En supposant que le degré de $P(x)$ soit inférieur à celui de $Q(x)$ dans l'expression rationnelle $\frac{P(x)}{Q(x)}$, on peut factoriser le dénominateur en facteurs linéaires et quadratiques irréductibles, par exemple $Q(x) = D_1(x) \cdot D_2(x) \cdots D_n(x)$, puis exprimer $\frac{P(x)}{Q(x)}$ comme une somme de fractions plus simples, chacune ayant pour dénominateur un D_i (ou éventuellement une puissance de D_i). Si le degré de $P(x)$ est supérieur ou égal à celui de $Q(x)$, on commence simplement la décomposition en procédant à une division polynomiale.

Il existe plusieurs cas à examiner afin de présenter la décomposition en fractions partielles de manière suffisamment générale. Nous attribuons des numérateurs inconnus N_i à chacune des fractions de la décomposition selon les règles suivantes :

1. Chaque dénominateur linéaire doit être associé à un numérateur constant, et chaque dénominateur quadratique irréductible doit être associé à un numérateur linéaire.
2. Si un facteur répété D_i^n apparaît dans la factorisation de $Q(x)$, alors la décomposition doit contenir des fractions dont les dénominateurs sont $D_i, D_i^2, \ldots, D_i^n$, chacune possédant un numérateur se conformant à la règle 1.

Une fois la décomposition en fractions partielles déterminée, on peut ensuite calculer les N_i algébriquement en résolvant un système d'équations. Les expressions rationnelles résultantes $\frac{N_i}{D_i}$ peuvent alors être intégrées rapidement, nécessitant au plus une substitution en u.

Exemple 1.3.3. Calculer la décomposition en fractions partielles de $\frac{x-4}{3x^3+5x^2+4x+2}$ en réalisant étape par étape les opérations algébriques requises.

Nous commençons par factoriser le dénominateur et écrire la décomposition en fractions partielles sous forme d'équation :

```
(%i17) kill(all)$
(%i1) R:(x-4)/(3*x^3+5*x^2+4*x+2)$
      factor(denom(R));
(%o2) (x+1)*(3*x^2+2*x+2)
(%i3) EQN:R=(A/(x+1))+(B*x+C)/(3*x^2+2*x+2);
(%o3) (x-4)/(3*x^3+5*x^2+4*x+2)=(C+x*B)/(3*x^2+2*x+2)+A/(x+1)
```

Nous multiplions maintenant les deux côtés de l'équation par le dénominateur initial, développons le résultat pour obtenir un polynôme, puis établissons un système d'équations en comparant les coefficients de chaque puissance de x des deux côtés :

```
(%i4) EQN*(denom(R));
(%o4) x-4=(3*x^3+5*x^2+4*x+2)*((C+x*B)/(3*x^2+2*x+2)+A/(x+1))
(%i5) ratsimp(%);
(%o5) x-4=(x+1)*C+(x^2+x)*B+(3*x^2+2*x+2)*A
(%i6) expand(%);
(%o6) x-4=x*C+C+x^2*B+x*B+3*x^2*A+2*x*A+2*A
(%i7) EQN1:coeff(lhs(%o6),x,2)=coeff(rhs(%o6),x,2);
      EQN2:coeff(lhs(%o6),x,1)=coeff(rhs(%o6),x,1);
      EQN3:coeff(lhs(%o6),x,0)=coeff(rhs(%o6),x,0);
(%o7) 0=B+3*A
(%o8) 1=C+B+2*A
(%o9) -4=C+2*A
```

Nous résolvons ce système d'équations en A, B et C et substituons dans la décomposition de départ :

```
(%i10) solve([EQN1,EQN2,EQN3],[A,B,C]);
(%o10) [[A=-5/3,B=5,C=-2/3]]
(%i11) sublis([A=-5/3,B=5,C=-2/3],rhs(EQN));
```

(%o11) $\frac{5x-\frac{2}{3}}{3x^2+2x+2} - \frac{5}{3(x+1)}$

La décomposition en fractions partielles est terminée, ce qui nous laisse deux termes relativement simples à intégrer. L'intégration de cette expression est laissée en exercice.

Vérifions notre réponse :

```
(%i12) ratsimp(%);
```

(%o12) $\frac{x-4}{3x^3+5x^2+4x+2}$

Dans l'exemple suivant, nous utilisons wxMaxima pour calculer automatiquement une décomposition en fractions partielles à l'aide de `partfrac`, mais les intégrales obtenues sont ensuite calculées « à la main » avec plus de détails.

Exemple 1.3.4. Calculer $\int \frac{x^4}{2x^3-2x^2+3x-3}\,\mathrm{d}x$ à l'aide d'une décomposition en fractions partielles.

Nous constatons que l'intégrale est impropre et que l'intégrande doit être transformé, donc nous devons commencer par utiliser la division euclidienne des polynômes. `partfrac` effectue en fait cette opération automatiquement, mais nous faisons d'abord la division pour illustrer la méthode. `divide` renvoie une liste contenant d'abord un polynôme puis un reste. Pour écrire le résultat sous forme correcte, le reste doit apparaître au-dessus du dénominateur initial :

```
(%i13) kill(all)$
(%i1) P:x^4$
      Q:2*x^3-2*x^2+3*x-3$
(%i3) divide(P,Q);
(%o3) [(x+1)/2,-(x^2-3)/2]
(%i4) PROPER:%[1]+%[2]/Q;
(%o4) (x+1)/2-(x^2-3)/(2*(2*x^3-2*x^2+3*x-3))
```

Nous appliquons maintenant `partfrac` afin d'obtenir la décomposition attendue :

```
(%i5) partfrac(PROPER,x);
(%o5) -(9*x+9)/(10*(2*x^2+3))+(x+1)/2+1/(5*(x-1))
```

Nous coupons en 4 termes différents que nous intégrons un par un :

1. $-\frac{9}{10}\int \frac{x}{2x^2+3}\,\mathrm{d}x$
2. $-\frac{9}{10}\int \frac{1}{2x^2+3}\,\mathrm{d}x$
3. $\frac{1}{2}\int (x+1)\,\mathrm{d}x$
4. $\frac{1}{5}\int \frac{1}{(x-1)}\,\mathrm{d}x$

Dans la première intégrale, nous effectuons une substitution informelle, en remarquant qu'un facteur de 4 placé au numérateur permettra de reconnaitre la forme $\int \frac{\mathrm{d}u}{u} = \ln|u|$:

$$-\tfrac{9}{10}\int \frac{x}{2x^2+3}\,\mathrm{d}x = -\frac{9}{10}\cdot\frac{1}{4}\int \frac{4x}{2x^2+3}\,\mathrm{d}x = -\frac{9}{40}\cdot\ln|2x^2+3|$$

La deuxième intégrale nécessite plus de réflexion : nous devons effectuer un changement de variable pour obtenir la forme $\int \frac{\mathrm{d}u}{u^2+1} = \tan^{-1} u$. Nous commençons par diviser par un facteur de 3 le dénominateur pour obtenir que le terme constant soit égal à 1, puis nous devinons et vérifions le u qui nous donne u^2+1 : :

```
(%i6)  P:1$
       Q:2*x^2+3$
       expand(Q/3);
(%o8) (2*x^2)/3+1
(%i9) u:sqrt(2/3)*x$
      u^2;
(%o10) (2*x^2)/3
```

Nous transformons ensuite l'intégrale :

```
(%i11) kill(all)$
(%i1) INTEGRAND:-(9/10)*1/(2*x^2+3)*del(x)$
(%i2) solve(diff(u)=diff(sqrt(2/3)*x),del(x));
(%o2) [del(x)=(sqrt(3)*del(u))/sqrt(2)]
(%i3) subst(rhs(%[1]),del(x),INTEGRAND)$
      subst((3/2)*u^2,x^2,%);
(%o4) -(3^(5/2)*del(u))/(5*2^(3/2)*(3*u^2+3))
(%i5) factor(denom(%));
(%o5) 15*2^(3/2)*(u^2+1)
(%i6) INTEGRAND_u:(-3^(5/2))/%;
```

$$\text{(\%o6)}\quad -\frac{3^{\frac{3}{2}}}{5\,2^{\frac{3}{2}}\,(u^2+1)}$$

Nous pouvons intégrer facilement en reconnaissant cette forme. Nous appelons le résultat $G(u)$. Enfin, nous y substituons la définition de x pour finaliser le résultat :

```
(%i7) G:coeff(%,1/(u^2+1))*atan(u);
(%o7) -(3^(3/2)*atan(u))/(5*2^(3/2))
(%i8) subst(sqrt(2/3)*x,u,%);
```

$$\text{(\%o8)}\quad -\frac{3^{\frac{3}{2}}\,\mathrm{atan}\left(\frac{\sqrt{2}\,x}{\sqrt{3}}\right)}{5\,2^{\frac{3}{2}}}$$

Les deux dernières intégrales peuvent être calculées directement en examinant leur forme :

$$\tfrac{1}{2}\int (x+1)\,\mathrm{d}x = \frac{1}{4}x^2+\frac{1}{2}x \text{ et } \tfrac{1}{5}\int \frac{1}{(x-1)}\,\mathrm{d}x = \frac{1}{5}\ln|x-1|$$

Et pour terminer, nous rassemblons les différents résultats :

$$\int \frac{x^4}{2x^3-2x^2+3x-3}\,\mathrm{d}x = -\frac{9}{40}\cdot\ln|2x^2+3| - \frac{3^{\frac{3}{2}}\,\mathrm{atan}\left(\frac{\sqrt{2}\,x}{\sqrt{3}}\right)}{5\,2^{\frac{3}{2}}} + \frac{1}{4}x^2 + \frac{1}{2}x + \frac{1}{5}\ln|x-1|$$

Vérifiant avec la commande de wxMaxima's `integrate` :

```
(%i9) integrate(x^4/(2*x^3-2*x^2+3*x-3),x);
```

(%o9) $-\frac{9\log\left(2\,x^2+3\right)}{40} - \frac{9\,\mathrm{atan}\left(\frac{2\,x}{\sqrt{6}}\right)}{10\,\sqrt{6}} + \frac{\log(x-1)}{5} + \frac{x^2+2\,x}{4}$

wxMaxima valide notre réponse moyennant le fait que l'écriture du terme dans la fonction arc tangente inverse n'est pas rationnalisée. Notez également que les barres de valeur absolue ne sont pas présentes dans les fonctions `log`.

1.4 Intégrales impropres

Les *intégrales impropres* sont des intégrales définies avec une limite en $\pm\infty$ ou une discontinuité dans l'intégrande. Formellement, les intégrales impropres sont calculées comme des limites. Par exemple :

Si la borne supérieure est infinie : $\displaystyle\int_a^{\infty} f(x)\ \mathrm{d}x = \lim_{t\to\infty}\int_a^{t} f(x)\ \mathrm{d}x$

Si $f(x)$ est discontinue en b : $\displaystyle\int_a^{b} f(x)\ \mathrm{d}x = \lim_{t\to b^-}\int_a^{t} f(x)\ \mathrm{d}x$

Ces formules se généralisent naturellement aux bornes inférieures.

En pratique, nous pouvons souvent nous contenter de calculer l'intégrale indéfinie et d'évaluer le résultat aux bornes d'intégration. Cependant, nous devons toujours veiller à couper tout intervalle d'intégration à la valeur d'une discontinuité de $f(x)$.

Exemple 1.4.1. Calculer $\displaystyle\int_1^{\infty} \frac{1}{x^2}\ \mathrm{d}x$.

D'abord nous essayons l'approche formelle en prenant la limite d'une fonction d'aire $\displaystyle\int_1^{t} \frac{1}{x^2}\ \mathrm{d}x$:

```
(%i1) f(x):=1/x^2$
      limit(integrate(f(x),x,1,t),t,inf);

Is  "t-1"  positive, negative, or zero? positive

(%o2) 1
```

wxMaxima demande une précision sur t pour déterminer l'intervalle d'intégration.

Recommençons le calcul de manière moins formelle :

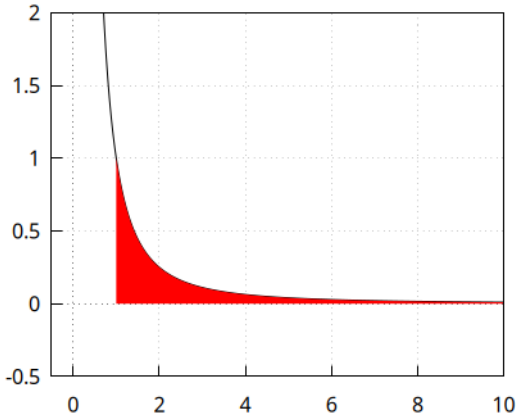
```
(%i3) integrate(f(x),x)$
      F(x):=''%;
(%o4) F(x):=-1/x
(%i5) F(inf)-F(1);

(%o5)   1 - 1/inf
```

De manière évidente, $\frac{1}{\infty} = 0$, et la réponse est bien 1.

Il est intéressant de réasliser un graphique hachuré pour illustrer cette intégrale : une aire s'étendant à l'infini peut encore être finie !

```
(%i6) wxdraw2d(
      grid=true,
      xaxis=true,
      yaxis=true,
```

```
xrange=[-.5,10],
yrange=[-.5,2],
color=black,
 explicit(f(x),x,-.5,10),
filled_func=true,
 filled_func=f(x),
 explicit(0,x,1,10),
filled_func=false
);
```



Exemple 1.4.2. Calculer $\displaystyle\int_0^1 \frac{1}{\sqrt{x}}\,\mathrm{d}x$.

Dans ce cas, l'intégrande devient infinie à la borne inférieure de l'intégrale. Formellement, nous devons calculer $\displaystyle\lim_{t\to 0^+}\int_t^1 \frac{1}{\sqrt{x}}\,\mathrm{d}x$:

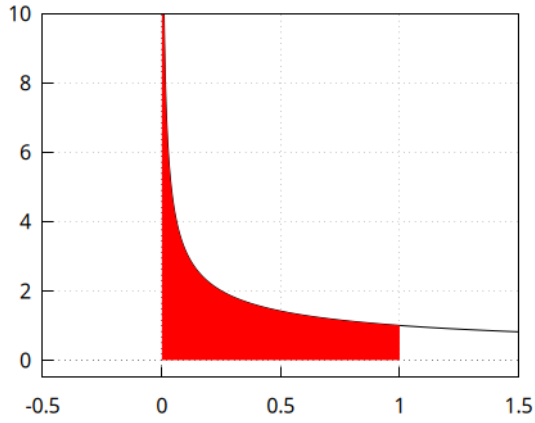
```
(%i7) f(x):=1/sqrt(x)$
      limit(integrate(f(x),x,t,1),t,0);

"Is  "t-1"  positive, negative, or zero?"negative;
"Is  "t"  positive, negative, or zero?"positive;

(%o8) 2
```

Recommençons ce calcul de manière moins formalisée :

```
(%i9) integrate(f(x),x)$
      F(x):=''(%);
(%o10) F(x):=2*sqrt(x)
(%i11) F(1)-F(0);
(%o11) 2
```

De nouveau, il est intéressant de tracer un graphique. Cette fois, nous avons une aire hachurée qui semble infiniment haute, mais qui est finie. Nous rencontrons une erreur lorsque nous voulons hachurer à partir de l'asymptote verticale, donc nous commençons à hachurer à partir de la valeur $x = .001$ au lieu de 0.

```
(%i12) wxdraw2d(
       grid=true,
       xaxis=true,
       yaxis=true,
       xrange=[-.5,1.5],
       yrange=[-.5,10],
       color=black,
        explicit(f(x),x,0.001,1.5),
       filled_func=true,
        filled_func=f(x),
        explicit(0,x,0.001,1),
       filled_func=false
       );
```



Exemple 1.4.3. Calculer $\displaystyle\int_1^4 \frac{1}{(x-3)^2}\,\mathrm{d}x$.

Cette fois, l'intervalle d'intégration contient une discontinuité. Pour calculer algébriquement l'intégrale, nous devons la scinder en une somme de deux intégrales au point de discontinuité, puis exprimer chaque intégrale à l'aide d'une limite :

$$\int_1^4 \frac{1}{(x-3)^2}\,\mathrm{d}x = \lim_{t\to 3^-}\int_1^t \frac{1}{(x-3)^2}\,\mathrm{d}x + \lim_{t\to 3^+}\int_t^4 \frac{1}{(x-3)^2}\,\mathrm{d}x$$

wxMaxima signale que les fonctions d'aire sont divergentes avant que nous puissions calculer les limites :

```
(%i13) f(x):=1/(x-3)^2$
```

```
(%i14) integrate(f(x),x,1,t);

"Is  "t-1"  positive, negative, or zero?"positive;
"Is  "t-2"  positive, negative, or zero?"positive;

defint: integral is divergent.
 -- an error. To debug this try: debugmode(true);
```

Cette réponse est insatisfaisante, aussi nous abordons le problème d'une autre manière, en utilisant une primitive de $f(x)$:

```
(%i15) A:integrate(f(x),x);
(%o15) -1/(x-3)
```

Nous calculons maintenant la première intégrale en utilisant la limite de la différence des valeurs de la primitive $\lim_{x\to 3^-}[A(x)-A(1)]$:

```
(%i16) limit(A,x,3,minus)-subst(1,x,A);
(%o16) infinity-1/2
```

Nous voyons que $\lim_{t\to 3^-}\int_1^t \frac{1}{(x-3)^2}\,\mathrm{d}x = +\infty$. La deuxième intégrale a aussi pour limite $+\infty$ (le calcul est laisséd comme exercice à réaliser).

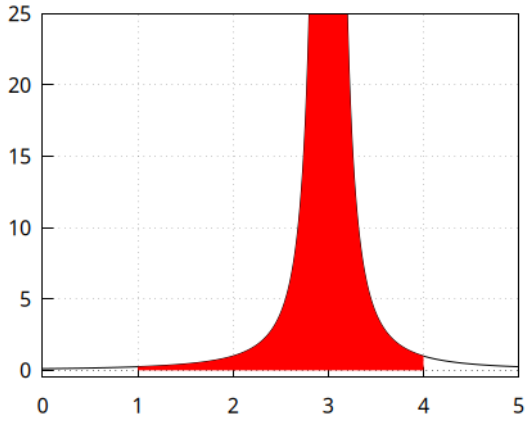
Un point intéressant à propos de ce problème est que nous pouvons obtenir une réponse erronée si nous ne tenons pas compte de la discontinuité :

```
(%i18) integrate(f(x),x)$
       F(x):=''(%);
(%o18) F(x):=-1/(x-3)
(%i19) F(4)-F(1);
(%o19) -3/2
```

Il s'agit d'un rappel important nous indiquant de toujours vérifier les points problématiques appartenant à l'intervalle étudié !

Nous terminons avec un rapide graphique pour être complet :

```
(%i17) wxdraw2d(
       grid=true,
       xaxis=true,
       yaxis=true,
       xrange=[0,5],
       yrange=[-.5,25],
       color=black,
        explicit(f(x),x,0,5),
       filled_func=true,
        filled_func=f(x),
        explicit(0,x,1,4),
       filled_func=false
       );
```


25
20
15
10
5
0
0
1
2
3
4
5

1.5 Exercices du Chapitre 1

1. Utiliser `integrate` afin de trouver une approximation numérique de l'intégrale définie $\displaystyle\int_4^7 \frac{1}{x^2 \cdot \sqrt{x-3}}\,\mathrm{d}x$, puis réaliser un graphique ombré à une échelle adaptée afin de mettre en évidence l'aire que représente cette intégrale.
2. Recommencez le premier exerice en définissant une fonction d'aire $A(x)$ et en évaluant la différence $A(7) - A(4)$. Vous êtes libre de choisir n'importe quel point de départ, tant que cela ne génère pas une intégrale qui soit divergente.
3. Recommencez le premier exercice en calculant une primitive $F(x)$ de $f(x) = \dfrac{1}{x^2 \cdot \sqrt{x-3}}$ et en évaluant $F(x)|_4^7$.
4. Calculer $\displaystyle\int x^2 \cdot \sqrt{1-x^3}\,\mathrm{d}x$ à l'aide d'un changement de variable en u selon le modèle de l'exemple 1.2.1.
5. A l'aide d'un changement de variable calculez $\displaystyle\int_0^{\frac{\pi}{6}} \sin(3x)\cos^3(3x)\,\mathrm{d}x$ comme dans l'exemple 1.2.2. Terminez le calcul en exprimant les bornes de l'intégrale en fonction de u plutôt que d'exprimer la primitive trouvée en fonction de x.
6. Utilisez `trigreduce` pour appliquer la formule de l'angle double à l'intégrande de $\displaystyle\int \cos^4 x\,\mathrm{d}x$. Calculez ensuite séparément chacune des intégrales résultantes en trouvant directement une primitive. Vérifiez le résultat final à l'aide de la commande `diff` suivie de `trigexpand` et de `trigsimp`.
7. Calculer l'intégrale $\displaystyle\int_{-3}^{3} \sqrt{9-x^2}\,\mathrm{d}x$ en effectuant une substitution trigonométrique dans le style de l'exemple 1.2.3. Tracer un graphe avec une zone ombrée et un rapport isométrique en utilisant `dimensions`. Expliquer comment cette intégrale peut être calculée à l'aide d'une formule géométrique simple.
8. Définir une fonction `CTS(x,a,b,c)` pour compléter le carré, étant donnés les coefficients d'un polynôme du second degré de la forme $ax^2 + bx + c$. Utiliser cette fonction pour compléter le carré de $2x^2 - 3x + 7$. Vérifier le résultat à l'aide de `expand`.
9. Utiliser `CTS(x,a,b,c)` pour compléter le carré au dénominateur de $\displaystyle\int \frac{\mathrm{d}x}{9x^2+12x+10}$. Effectuer un changement de variable en u pour obtenir la forme $\displaystyle\int \frac{1}{u^2+1}\,\mathrm{d}u$ (avec éventuellement des constantes devant). Finalement, calculer l'intégrale et transformer le résultat en x. Vérifier à l'aide de `diff`.
10. Terminez l'intégrale de l'Exemple 1.3.3 :

$$\int \frac{5x - \dfrac{2}{3}}{3x^2+2x+2}\,\mathrm{d}x - \int \frac{5}{3(x+1)}\,\mathrm{d}x$$

La deuxième intégrale peut déjà être calculée par la recherche directe d'une primitive. La première intégrale peut être transformée en deux intégrales :

$$\int \frac{\mathrm{d}u}{u} + \int \frac{\text{constantes}}{3x^2+2x+2}\,\mathrm{d}x$$

Calculez le premier morceau en observant la forme de l'expression, puis utilisez `CTS(x,a,b,c)` pour transformer le dénominateur dans le second morceau. Effectuez le changement de variable u approprié afin d'obtenir une intégrale que l'on peut calculer directement avec une primitive évidente. Vérifiez enfin votre résultat à l'aide de `diff`.

11. Intégrer $\int x \cdot \sin^2 x \, \mathrm{d}x$ à l'aide d'une intégration par parties comme dans l'exemple 1.3.1.
12. Lorsque nous appliquons une intégration par parties à une intégrale définie, nous obtenons $\int_a^b u \, \mathrm{d}v = uv|_a^b - \int_a^b v \, \mathrm{d}u$. Appliquer cette formule pour calculer $\int_0^2 x \cdot e^x \, \mathrm{d}x$.
13. Calculez $\int e^{-\alpha t} \cos \beta t \, \mathrm{d}t$ en utilisant une intégration par parties. Vous obtenez alors une équation avec $\int e^{-\alpha t} \cos \beta t \, \mathrm{d}t$ comme membre gauche et les résultats de la première application de l'intégration par parties au membre droit. Répétez en appliquant une nouvelle fois une une intégration par parties. Après ces deux intégrations par parties, vous devriez reconnaître un terme contenant l'intégrale originale situé dans le membre droit. Regroupez les deux termes contenant l'intégrale originale dans le membre de gauche, et résolvez pour trouver la valeur de cette intégrale. Vérifiez votre réponse en utilisant `diff`. Tout au long de vos calculs, utilisez `%alpha` et `%beta`. Vous devrez utiliser `declare` pour que `diff` reconnaisse α et β comme des constantes.
14. Créez une fonction `DOS(x,a,b)` pour factoriser $ax^2 - b$ en deux facteurs linéaires, même si a et b ne sont pas des carrés parfaits. Appliquez `DOS(x,a,b)` au dénominateur de $\int \dfrac{\mathrm{d}x}{2x^2 - 5}$. Effectuez ensuite une décomposition en éléments simples dans le style de l'Exemple 1.3.3 et terminez le calcul de l'intégrale par la recherche directe d'une primitive.
15. Calculez l'intégrale de l'exercice précédent en utilisant une substitution trigonométrique dans le style des Exemples 1.2.3 et 1.2.4. Vérifiez que votre réponse est identique à la précédente.
16. Calculez l'intégrale $\int \dfrac{1}{x \cdot \sqrt{1-x}} \, \mathrm{d}x$ en commençant par la substitution $u^2 = 1 - x$. Une fois l'intégrale transformée en fonction de u, utilisez `partfrac` pour effectuer une décomposition en éléments simples, puis terminez le calcul des intégrales en u par une recherche directe de primitive. N'oubliez pas d'exprimer votre réponse finale en x.
17. Calculez la deuxième intégrale impropre de l'exemple 1.4.3 : $\lim\limits_{t \to 3^+} \int_t^4 \dfrac{1}{(x-3)^2} \, \mathrm{d}x$
18. Calculez $\int_0^\infty \dfrac{1}{x^2 + a^2} \, \mathrm{d}x$ en effectuant une substitution trigonométrique appropriée et en transformant les bornes d'intégration en fonction de la variable de substitution.

Chapitre 2

Techniques pour intégrer numériquement

Commandes essentielles de ce Chapitre

`integrate`	`sum`	`rhs`
`float`	`for-do`	`kill(all)`
`rectangle`	`polygon`	`ratprint`
`makelist`	`sublis`	`quad_qag`
`wxdraw2d`	`solve`	`random`

2.1 Somme des points milieux

Si la formule analytique d'une fonction $f(x)$ est connue, on peut approcher $\int_a^b f(x)\,\mathrm{d}x$ en utilisant une somme de Riemann par la méthode du point milieu. Autrement dit, on découpe l'intervalle $[a, b]$ en n sous-intervalles aux points de coupure $a = x_0, x_1, x_2, \ldots, x_{n-1}, x_n = b$, puis on utilise le point milieu de chaque sous-intervalle pour calculer la hauteur de chaque rectangle pour cette approximation.

On choisit des sous-intervalles de même largeur $\Delta x = \dfrac{b-a}{n}$, et la hauteur du $i^{\text{ème}}$ rectangle est donnée par $f\left(\dfrac{x_{i-1}+x_i}{2}\right)$. En additionnant les aires de ces rectangles, on obtient l'approximation par la méthode du point milieu :

$$f\left(\frac{x_0+x_1}{2}\right)\Delta x + f\left(\frac{x_1+x_2}{2}\right)\Delta x + \cdots + f\left(\frac{x_{n-1}+x_n}{2}\right)\Delta x = \sum_{i=1}^{n} f\left(\frac{x_{i-1}+x_i}{2}\right)\Delta x$$

L'approximation par la méthode du point milieu devient plus précise au fur et à mesure que n augmente et que les rectangles deviennent plus fins.

Exemple 2.1.1. Montrez que wxMaxima ne peut pas calculer analytiquement $\int_0^3 e^{\sin x}\,\mathrm{d}x$. Calculez la somme des points milieux pour $n = 20$ qui approxime l'intégrale, et tracez un graphique illustrant les rectangles correspondant à cette somme des points milieux.

Nous commençons par définir $f(x)$ et tentons d'utiliser `integrate`. wxMaxima renvoie simplement l'intégrale, indiquant qu'il ne peut pas trouver de solution analytique.

```
(%i1) f(x):=%e^(sin(x))$
      integrate(f(x),x,0,3);
(%o2) integrate(%e^sin(x),x,0,3)
```

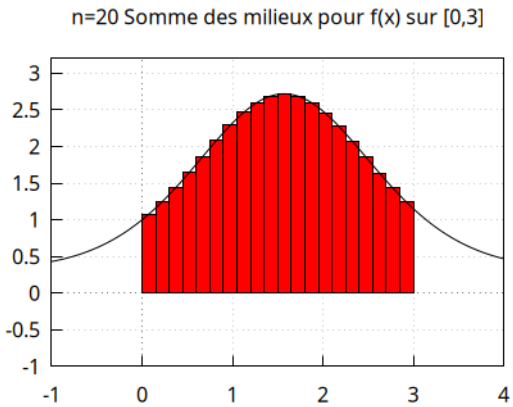
Maintenant, nous définissons Δx et x_i puis nous calculons l'approximation par les points milieux :

```
(%i3) delx:(3-0)/20;
      x(i):=''delx*i;
(%o3) 3/20
(%o4) x(i):=3/20*i

(%i5) float(sum(f((x(i)+x(i-1))/2)*delx,i,1,20));
(%o5) 6.058676126838001
```

Nous utilisons `rectangle` à l'intérieur de `makelist` pour générer une liste de rectangles compatible avec `wxdraw2d`. Notez que `rectangle` trace un rectangle à partir d'une liste de seulement deux sommets situés aux coins opposés.

```
(%i6) RECTANGLES:makelist(rectangle([x(i-1),0],[x(i),f((x(i)+x(i-1))/2)])
```

```
      ,i,1,20)$
(%i7) load(draw)$
      wxdraw2d(
      grid=true,
      xaxis=true,
      yaxis=true,
      yrange=[-1,3.2],
      title="n=20 Somme des milieux pour f(x) sur [0,3]",
      color=black,
      fill_color=red,
      border=true,
       RECTANGLES,
       explicit(f(x),x,-1,4)
      );
```



Exemple 2.1.2. Écrire une boucle `for-do` pour calculer les approximations par le point milieu de $\int_0^3 e^{\sin x}\,\mathrm{d}x$ en utilisant $n = 20, 40, 60, \ldots$ jusqu'à ce que l'approximation se stabilise à au moins trois décimales près.

En premier, nous adaptons la somme pour $n = 20$ de manière à travailler de manière plus générale :

```
(%i8) f(x):=%e^(sin(x))$
      delx(n):=(3-0)/n$
      x(i,n):=''delx(n)*i$
(%i9) SUM(n):=sum((f((x(i-1,n)+x(i,n))/2)*delx(n)),i,1,n)$
```

Nous écrivons maintenant la boucle for-do et l'exécutons :

```
(%i10) (print("n....MIDPOINT SUM"),
        for k:1 thru 10 do
        (n:(20*k),
         S:(float(SUM(n))),
```

```
        print(n,"","","",S))
      );

 n....MIDPOINT SUM
20    6.058676126838001
40    6.057171126434241
60    6.056892460067756
80    6.05679492965105
100   6.056749787496514
120   6.056725265963955
140   6.056710480298045
160   6.05670088385024
180   6.056694304565428
200   6.056689598445805
```

Il semble que la valeur de l'approximation pour cette intégrale se stabilise à 6,056, donc 10 itérations sont suffisantes.

2.2 Sommes des trapèzes

2.2.1 Pour une fonction définie analytiquement

Lorsque l'expression analytique de $f(x)$ est connue, nous pouvons approximer $\int_a^b f(x)\ \mathrm{d}x$ en utilisant des trapèzes à base étroite, chacun ayant une hauteur de $f(x_{i-1})$ sur le côté gauche et $f(x_i)$ sur le côté droit. Une fois de plus, nous divisons l'intervalle $[a, b]$ en n sous-intervalles égaux de largeur $\Delta x = \frac{b-a}{n}$, et l'aire de chaque trapèze est donnée par $\frac{(f(x_{i-1})+f(x_i))}{2} \cdot \Delta x$. L'approximation de l'aire est alors donnée par :

$$\frac{(f(x_0)+f(x_1))}{2} \cdot \Delta x + \cdots + \frac{(f(x_{n-1})+f(x_n))}{2} \cdot \Delta x = \sum_{i=1}^{n} \frac{(f(x_{i-1})+f(x_i))}{2} \cdot \Delta x$$

Exemple 2.2.1. Calculer l'approximation par des trapèzes pour $n = 20$ de $\int_0^3 e^{\sin x}\ \mathrm{d}x$, et comparer le résultat aux approximations par le point médian effectuées dans l'exemple précédent. Réaliser un graphique pour illustrer cette approximation.

Nous commençons par définir Δx, x_i et l'aire d'un seul trapèze, puis nous calculons la somme :

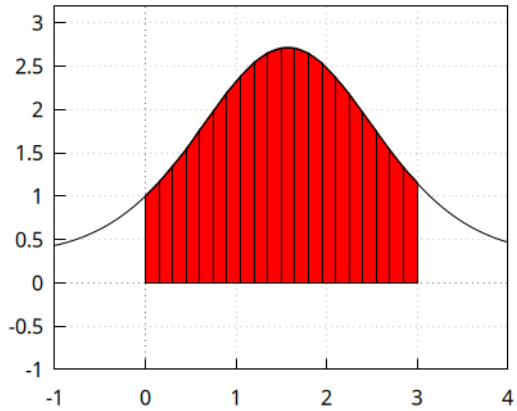
```
(%i1) f(x):=%e^(sin(x))$
      delx:(3-0)/20$
      x(i):=delx*i$
      AREA(i):=((f(x(i-1))+f(x(i)))/2)*delx;
(%o4) AREA(i):=(f(x(i-1))+f(x(i)))/2*delx
(%i5) float(sum(AREA(i),i,1,20));
(%o5) 6.052656613881449
```

Lorsque nous comparons cette approximation à la somme des points médians pour $n = 20$, nous constatons que l'approximation par les trapèzes est en réalité moins précise que l'approximation par le point médian dans ce cas. Les trapèzes *sous-estiment* systématiquement l'aire car $f(x)$ est concave vers le bas sur presque tout l'intervalle considéré.

Une fois de plus, nous utilisons `makelist` pour tracer l'approximation trapézoïdale. Cette fois, nous devons utiliser `polygon`, qui attend une liste de sommets ordonnés autour de chaque trapèze :

```
(%i6) TRAPEZOIDS:makelist(polygon([ [x(i-1),0], [x(i),0],
           [x(i),f(x(i))], [x(i-1),f(x(i-1))] ]),i,1,20)$
(%i7) wxdraw2d(
      grid=true,
      xaxis=true,
      yaxis=true,
      yrange=[-1,3.2],
      color=black,
      border=true,
```

```
fill_color=red,
 TRAPEZOIDS,
 explicit(f(x),x,-1,4)
);
```



2.2.2 For a Discrete Data Set

Si la formule analytique de $f(x)$ n'est *pas* connue (comme dans le cas d'une liste de points de données), on peut tout de même effectuer une approximation par la méthode des trapèzes en reliant simplement les points donnés les uns aux autres.

Exemple 2.2.2. Dans une expérience de physique, la vitesse d'un objet est mesurée toutes les 0,01 s afin d'obtenir l'ensemble de données suivant :

```
t(s).....v(m/s)

0.01      2.3
0.02      1.8
0.03      1.8
0.04      1.7
0.05      1.5
0.06      1.5
0.07      1.3
0.08      1.1
0.09      0.8
0.10      0.5
```

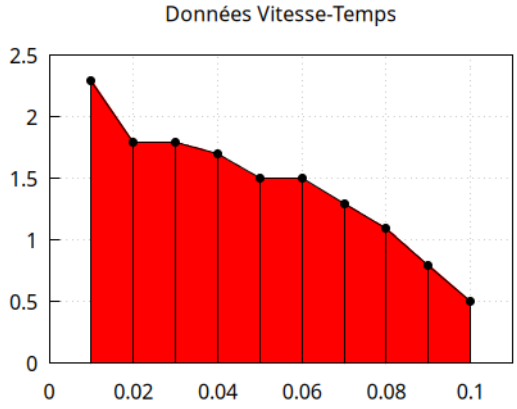
Calculer le déplacement total de l'objet $\Delta x = \int_{0,01}^{0,10} v(t)\ \mathrm{d}t$, en utilisant une approximation par la méthode des trapèzes, puis tracer le graphe de $v(t)$ afin d'illustrer cette approximation.

Dans cet exemple, l'ensemble de données est régulièrement espacé sur l'axe des t, ce qui nous permet de définir un $\Delta t = 0{,}01$ fixe. Nous saisissons les données au format de liste, en commençant par les listes des coordonnées x et des coordonnées y, utilisons `makelist` pour créer des paires ordonnées et calculons la somme des aires des trapèzes en faisant appel aux éléments de liste nécessaires :

```
(%i1) X:[.01,.02,.03,.04,.05,.06,.07,.08,.09,.10]$
      Y:[2.3,1.8,1.8,1.7,1.5,1.5,1.3,1.1,.8,.5]$
      POINTS:makelist([X[i],Y[i]],i,1,10)$
      delt:0.01$
(%i5) TRAPEZOID(i):=0.5*(Y[i]+Y[i+1])*delt$
      sum(TRAPEZOID(i),i,1,9);
(%o6) 0.129
```

Nous obtenons un déplacement approximatif de $\Delta x \approx 0{,}129\,\text{m}$. Nous procédons ensuite au tracé en définissant une liste de trapèzes et en faisant appel aux éléments de notre liste :

```
(%i7) TRAPEZOIDS:makelist(polygon([[X[i],0],
       [X[i+1],0],[X[i+1],Y[i+1]],[X[i],Y[i]]]),i,1,9)$
(%i8) wxdraw2d(
      grid=true,
      xaxis=true,
      yaxis=true,
      xrange=[0,.11],
      yrange=[0,2.5],
      title="Données Vitesse-Temps",
      color=black,
      border=true,
      fill_color=red,
       TRAPEZOIDS,
      point_type=7,
       points(POINTS)
      );
```



Exemple 2.2.3. La force exercée par un ressort est mesurée en fonction de l'allongement afin d'obtenir l'ensemble de données suivant :

```
x(m).....F(N)
0.10      10
0.15      20
0.19      30
0.22      40
0.25      50
0.28      60
0.30      70
0.32      80
0.33      90
0.34     100
```

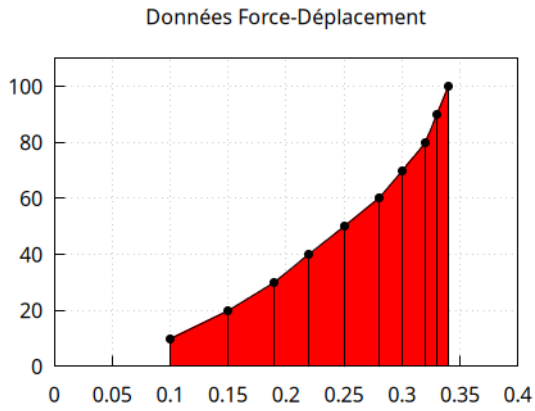
Utiliser une approximation par la méthode des trapèzes pour calculer le travail effectué sur le ressort : $W = \int_{0,10}^{0,34} F(x)\ \mathrm{d}x$. Tracer le graphe de $F(x)$ pour illustrer l'approximation par la méthode des trapèzes.

Cette fois, l'espacement est irrégulier sur l'axe des x, de sorte que nous ne pouvons pas définir une largeur de trapèze fixe. Nous ajustons les aires des trapèzes en utilisant les différences entre les valeurs adjacentes de x pour définir la largeur de chaque trapèze.

```
(%i9) kill(all)$
      X:[.1,.15,.19,.22,.25,.28,.30,.32,.33,.34]$
      Y:[10,20,30,40,50,60,70,80,90,100]$
      POINTS:makelist([X[i],Y[i]],i,1,10)$
(%i4) TRAPEZOID(i):=0.5*(Y[i]+Y[i+1])*(X[i+1]-X[i])$
      sum(TRAPEZOID(i),i,1,9);
(%o5) 10.4
```

Nous constatons que le travail total exercé sur le ressort est $W \approx 10{,}4$ J. Nous traçons l'approximation comme précédemment :

```
(%i6) TRAPEZOIDS:makelist(polygon([ [X[i],0],[X[i+1],0],
      [X[i+1],Y[i+1]],[X[i],Y[i]] ]),i,1,9)$
(%i7) wxdraw2d(
      grid=true,
      xaxis=true,
      yaxis=true,
      xrange=[0,.4],
      yrange=[0,110],
      title="Données Force-Déplacement",
      color=black,
      border=true,
      fill_color=red,
       TRAPEZOIDS,
      point_type=7,
       points(POINTS)
      );
```


Données Force-Déplacement
100
80
60
40
20
0
0
0.05
0.1
0.15
0.2
0.25
0.3
0.35
0.4

2.3 La méthode de Simpson

2.3.1 Pour une fonction définie analytiquement

Les approximations par les méthodes des points médian et des trapèzes sont toutes deux des exemples d'approximations par *polynômes d'interpolation*. L'approximation par le point médian utilise des polynômes de degré 0 (fonctions constantes) pour estimer l'aire sur chaque sous-intervalle, et l'approximation par les trapèzes utilise des polynômes de degré 1 (droites) pour estimer l'aire sur chaque sous-intervalle. La méthode de Simpson étend simplement cette idée aux polynômes de degré 2 ; c'est-à-dire que nous approximons la fonction par une suite de *paraboles* sur chaque paire de sous-intervalles.

Chaque parabole de la méthode de Simpson passe par trois points consécutifs de la fonction que l'on souhaite intégrer. Il est toujours possible de calculer la formule d'une parabole passant par trois points, car la formule d'une parabole $y = ax^2 + bx + c$ possède trois coefficients indéterminés. Les points définissent donc un système de trois équations à trois inconnues.

Exemple 2.3.1. Trouver une expression pour la parabole passant par les points de coordonnées $(-1, 1)$, $(0, 3)$ et $(2, -1)$, puis tracer cette parabole ainsi que les points donnés.

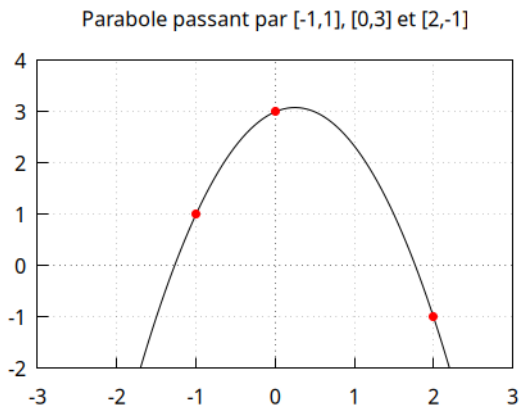
Nous commençons par définir une équation générale pour la parabole, $y = ax^2 + bx + c$, puis nous utilisons les valeurs des points pour définir un système d'équations. Après avoir résolu le système obtenu, nous substituons a, b, c dans l'équation générale :

```
(%i1) EQN:y=a*x^2+b*x+c$
      EQN1:sublis([x=-1,y=1],EQN);
      EQN2:sublis([x=0,y=3],EQN);
      EQN3:sublis([x=2,y=-1],EQN);

(%o2) 1=c-b+a
(%o3) 3=c
(%o4) -1=c+2*b+4*a

(%i5) solve([EQN1,EQN2,EQN3],[a,b,c]);
(%o5) [[a=-4/3,b=2/3,c=3]]
(%i6) sublis([a=-4/3,b=2/3,c=3],EQN);

(%o6) y=-(4*x^2)/3+(2*x)/3+3
```

Pour terminer, nous représentons graphiquement la parabole passant par les trois points de coordonnées $(-1, 1)$, $(0, 3)$ et $(2, -1)$:

```
(%i7) PARABOLA:rhs(%);
(%o7) -(4*x^2)/3+(2*x)/3+3
(%i8) wxdraw2d(
       grid=true,
       xaxis=true,
       yaxis=true,
       xrange=[-3,3],
```

```
yrange=[-2,4],
title="Parabole passant par [-1,1], [0,3] et [2,-1]",
color=black,
 explicit(PARABOLA,x,-3,3),
color=red,
point_type=7,
 points([ [-1,1],[0,3],[2,-1] ])
);
```



Nous appliquons la méthode de Simpson en approximant $f(x)$ par une parabole distincte pour chaque paire de sous-intervalles consécutifs. L'aire sous chaque parabole se calcule facilement, puis nous additionnons toutes les aires afin d'obtenir l'approximation de $\int_a^b f(x)\,\mathrm{d}x$. Si le nombre de sous-intervalles, n, est pair et que les sous-intervalles ont une largeur égale, le résultat se simplifie en la *formule de Simpson*, que l'on peut trouver dans tout manuel de calcul intégral.

Exemple 2.3.2. Trouver une approximation par la méthode de Simpson de $\int_0^3 e^{\sin x}\,\mathrm{d}x$ en utilisant $n = 6$. Tracer un graphique montrant $e^{\sin x}$ ainsi que la partie pertinente de chaque parabole d'interpolation.

La méthode de Simpson consiste à construire une parabole sur chaque *paire* consécutive de sous-intervalles ; il nous faut donc seulement trois paraboles pour cet exemple :

`PARABOLA1` relie $(x_0, f(x_0))$, $(x_1, f(x_1))$ et $(x_2, f(x_2))$
`PARABOLA2` relie $(x_2, f(x_2))$, $(x_3, f(x_3))$ et $(x_4, f(x_4))$
`PARABOLA3` relie $(x_4, f(x_4))$, $(x_5, f(x_5))$ et $(x_6, f(x_6))$

Pour des grandes valeurs de n, on peut entièrement automatiser le processus de recherche des paraboles. Nous adoptons ici une approche hybride en automatisant uniquement le calcul de `EQN(i)` obtenu en substituant $(x_i, f(x_i))$ dans l'équation générale d'une parabole :

```
(%i9) f(x):=%e^(sin(x))$
(%i10) GENEQN:Y=a*X^2+b*X+c$
(%i11) delx:0.5$
(%i12) x(i):=i*delx$
(%i13) EQN(i):=(sublis([X=x(i),Y=f(x(i))],GENEQN))$
```

Nous avons maintenant trois systèmes d'équations à résoudre et trois paraboles à construire. Notons que la substitution de a, b et c dans `GENEQN` nécessite un `[1]` supplémentaire pour extraire les coefficients entre crochets issus de la sortie de `solve` :

```
(%i14) COEFFS1:float(solve([EQN(0),EQN(1),EQN(2)],[a,b,c]));
(%o14) [[a=0.17896846366337,b=1.140808361052482,c=1.0]]
(%i15) PARABOLA1:rhs(sublis(COEFFS1[1],GENEQN));
(%o15) 0.17896846366337*X^2+1.140808361052482*X+1.0

(%i16) COEFFS2:float(solve([EQN(2),EQN(3),EQN(4)],[a,b,c]));
(%o16) [[a=-1.241214965266928,b=3.886445799099931,c=-0.32545400911715]]
(%i17) PARABOLA2:rhs(sublis(COEFFS2[1],GENEQN));
(%o17) -1.241214965266928*X^2+3.886445799099931*X-0.32545400911715

(%i18) COEFFS3:float(solve([EQN(4),EQN(5),EQN(6)],[a,b,c]));
(%o18) [[a=-0.0090668352651675,b=-1.285680715174632,c=5.090206499424927]]
(%i19) PARABOLA3:rhs(sublis(COEFFS3[1],GENEQN));
(%o19) -0.0090668352651675*X^2-1.285680715174632*X+5.090206499424927
```

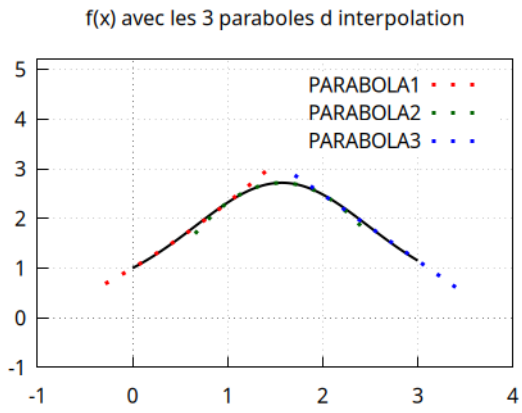
Maintenant que nous disposons des équations des trois paraboles, nous pouvons intégrer chacune d'elles sur les sous-intervalles pertinents (x_0, x_2), (x_2, x_4) et (x_4, x_6) :

```
(%i20)  float(integrate(PARABOLA1,X,x(0),x(2))
              +integrate(PARABOLA2,X,x(2),x(4))
              +integrate(PARABOLA3,X,x(4),x(6))
             );
(%o20) 6.056688193799587
```

Enfin, nous traçons $f(x)$ ainsi que les paraboles d'interpolation. Chaque parabole est tracée légèrement au-delà de l'intervalle d'intégration associé, afin de mieux visualiser ce qui se passe :

```
(%i21) wxdraw2d(
       grid=true,
       xaxis=true,
       yaxis=true,
       xrange=[-1,4],
       yrange=[-1,5.2],
       title="f(x) avec les 3 paraboles d interpolation",
       color=black,
       line_width=2,
        explicit(f(x),x,0,3),
       line_type=dots,
       color=red,
       line_width=3,
```

```
key="PARABOLA1",
 explicit(PARABOLA1,X,x(0)-.4,x(2)+.4),
color=dark_green,
key="PARABOLA2",
 explicit(PARABOLA2,X,x(2)-.4,x(4)+.4),
color=blue,
key="PARABOLA3",
 explicit(PARABOLA3,X,x(4)-.4,x(6)+.4)
);
```



2.3.2 *Pour un ensemble de données discrètes

La méthode de Simpson peut encore être utilisée pour approcher une intégrale si n est impair ou si l'espacement des sous-intervalles varie. On peut donc appliquer la méthode à des ensembles de données réalistes. Il faut être particulièrement vigilant lorsque n est impair, car l'intervalle d'intégration ne peut pas être découpé proprement en paires de sous-intervalles – dans ce cas, on peut simplement utiliser l'approximation par les trapèzes sur le dernier sous-intervalle.

Exemple 2.3.3. Pour le même ensemble de données qu'à l'exemple 2.2.3, calculer le travail total exercé sur le ressort $W = \int_{.10}^{.34} F(x)\,\mathrm{d}x$ en utilisant des paraboles d'interpolation sur les 8 premiers sous-intervalles et un trapèze sur le dernier intervalle. Réaliser un graphique illustrant l'approximation.

```
x(m).....F(N)
0.10      10
0.15      20
0.19      30
0.22      40
0.25      50
0.28      60
0.30      70
```

```
0.32     80
0.33     90
0.34    100
```

Cette fois, nous devons travailler avec une liste de points de données spécifiques plutôt qu'avec une formule de fonction à intégrer. De plus, nous cherchons à automatiser le processus autant que possible en utilisant `makelist`. Nous masquerons également de nombreux résultats intermédiaires par souci de concision. Nous commençons par définir et résoudre le système d'équations pour chaque parabole d'interpolation. Les indices `2*k+1`, `2*k+2` et `2*k+3` sont utilisés pour appeler les points $\{(x_1, y_1), (x_2, y_2), (x_3, y_3)\}$, $\{(x_3, y_3), (x_4, y_4), (x_5, y_5)\}$, $\{(x_5, y_5), (x_6, y_6), (x_7, y_7)\}$, $\{(x_7, y_7), (x_8, y_8), (x_9, y_9)\}$ pour les quatre paraboles :

```
(%i22) kill(all)$
(%i1) x:[.1,.15,.19,.22,.25,.28,.30,.32,.33,.34]$
      y:[10,20,30,40,50,60,70,80,90,100]$
      POINTS:makelist([x[i],y[i]],i,1,10)$
(%i4) GENEQN:Y=a*X^2+b*X+c$
(%i5) EQN(i):=(sublis([X=x[i],Y=y[i]],GENEQN))$
(%i6) ratprint:false$
(%i7)  SOLUTIONS:makelist(float(solve([EQN(2*k+1),EQN(2*k+2),EQN(2*k+3)],
                                [a,b,c])),k,0,3);
(%o7) [[[a=555.5555555555555,b=61.11111111111111,c=-1.666666666666667]],
      [[a=0.0,b=333.3333333333333,c=-33.33333333333334]],
      [[a=3333.333333333334,b=-1433.333333333333,c=200.0]],
      [[a=16666.66666666667,b=-9833.333333333334,c=1520.0]]]
```

Ensuite, nous substituons chacune de ces solutions pour a, b, c dans l'équation générale $y = ax^2 + bx + c$ afin d'obtenir une équation pour chaque parabole d'interpolation. La partie la plus délicate de cette étape est de bien comprendre le format de liste de `SOLUTIONS` – il s'agit d'une liste de solutions, mais chaque solution est représentée comme une liste contenant un seul élément (les solutions séparées par des virgules). Nous appelons chaque élément avec `SOLUTIONS[l+1]`, puis nous extrayons chaque solution en ajoutant un `[1]` supplémentaire. Une liste d'intégrales est ensuite calculée, chacune sur le sous-intervalle approprié. Enfin, les intégrales sont sommées :

```
(%i8) intPARABOLAS:makelist(rhs(sublis((SOLUTIONS[l+1])[1],GENEQN)),l,0,3)$
(%i9) INTEGRALS:makelist(integrate(intPARABOLAS[m+1],X,x[2*m+1],
                    x[2*m+3]),m,0,3)$
(%i10) SIMPSONPART:float(sum(INTEGRALS[i],i,1,4));
(%o10) 9.388055555555557
```

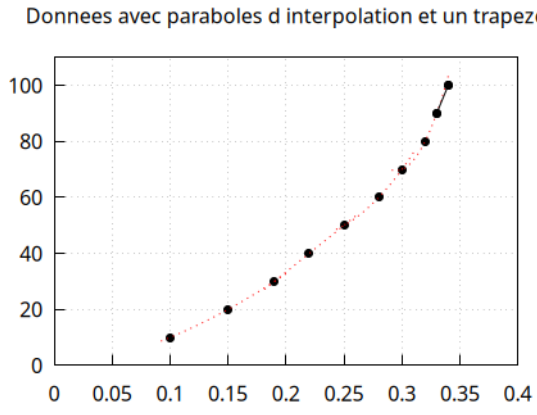
Maintenant, nous ajoutons le trapèze défini sur (x_9, x_{10}) et nous calculons l'aire totale :

```
(%i11) TRAPEZOIDAREA:0.5*(y[10]+y[9])*(x[10]-x[9]);
(%o11) 0.95

(%i12) SIMPSONPART+TRAPEZOIDAREA;
(%o12) 10.33805555555556
```

On constate que l'intégrale du travail donne $W \approx 10.34$, ce qui est en accord avec l'approximation par les trapèzes ($W \approx 10.4$) effectuée à l'exemple 2.2.3.

Nous terminons en construisant un graphique de l'approximation par la méthode de Simpson ainsi que le trapèze sur le dernier sous-intervalle. Nous utilisons `makelist` pour pourvoir tracer nos paraboles avec la commande `wxdraw2d`, en étendant légèrement le domaine au-delà de chaque intervalle d'interpolation afin de rendre les paraboles plus visibles.

```
(%i13) graphPARABOLAS:makelist(explicit(rhs(sublis((SOLUTIONS[1+1])[1],
       GENEQN)),X,x[2*1+1]-.01,x[2*1+3]+.01),1,0,3)$
(%i14) wxdraw2d(
       grid=true,
       xaxis=true,
       yaxis=true,
       xrange=[0,.4],
       yrange=[0,110],
       title="Donnees avec paraboles d interpolation et un trapeze",
       color=black,
       point_type=7,
       points(POINTS),
       color=red,
       line_type=short_dashes,
        graphPARABOLAS,
       points_joined=true,
       color=black,
       line_type=solid,
        points([[x[9],y[9]],[x[10],y[10]]])
       );
```



2.4 Méthodes de quadrature intégrées dans wxMaxima

Le terme *quadrature* est utilisé pour décrire une variété de méthodes d'intégration numérique. Nous ne souhaitons pas approfondir davantage l'intégration numérique, et nous n'utiliserons qu'une seule commande d'intégration numérique, `quad_qag`. La syntaxe de `quad_qag` est la même que celle de `integrate`, sauf que nous devons entrer un cinquième argument : un entier compris entre 1 et 6 indiquant la méthode choisie pour la quadrature. La sortie de `quad_qag` affiche quatre nombres : le premier est l'approximation numérique de l'intégrale, le deuxième est une approximation de l'erreur (nous ignorons les autres).

Exemple 2.4.1. Calculez l'intégrale $\int_0^3 e^{\sin x}\,\mathrm{d}x$ en utilisant la procédure intégrée pour la quadrature de wxMaxima. Comparez les résultats des 6 méthodes de quadrature disponibles pour `quad_qag`.

```
(%i1) f(x):=%e^(sin(x));
(%o1) f(x):=%e^sin(x)

(%i2) quad_qag(f(x),x,0,3,1);
(%o2) [6.05666953555315,5.7821357200595372*10^-12,45,0]
(%i3) quad_qag(f(x),x,0,3,2);
(%o3) [6.056669535553152,9.9580925827612082*10^-11,21,0]
(%i4) quad_qag(f(x),x,0,3,3);
(%o4) [6.05666953555315,6.7242539709168614*10^-14,31,0]
(%i5) quad_qag(f(x),x,0,3,4);
(%o5) [6.056669535553152,6.724253970916864*10^-14,41,0]
(%i6) quad_qag(f(x),x,0,3,5);
(%o6) [6.056669535553151,6.7242539709168627*10^-14,51,0]
(%i7) quad_qag(f(x),x,0,3,6);
(%o7) [6.056669535553151,6.7242539709168627*10^-14,61,0]
```

On voit que $\int_0^3 e^{\sin x}\,\mathrm{d}x \approx 6{,}057$, ce qui est cohérent avec nos approximations précédentes. Toutes les méthodes de `quad_qag` donnent la même valeur du résultat sur de nombreuses décimales.

2.5 Une méthode de Monte Carlo

Le terme *Monte Carlo* recouvre une variété de méthodes numériques qui exploitent des données aléatoires. On peut par exemple générer une méthode de Monte Carlo simple pour calculer $\int_a^b f(x)\ \mathrm{d}x$ en égalant deux méthodes de calcul de la *valeur moyenne* d'une fonction :

1. $f_{avg} = \dfrac{1}{b-a}\displaystyle\int_a^b f(x)\ \mathrm{d}x$
2. $f_{avg} \approx \dfrac{1}{N}\displaystyle\sum_{i=1}^{N} f(x_i)$ où les N valeurs de x_i sont générées aléatoirement sur $[a, b]$.

Nous égalons les deux calculs de f_{avg} pour obtenir :

$$\int_a^b f(x)\ \mathrm{d}x \approx \frac{(b-a)}{N} \cdot \sum_{i=1}^{N} f(x_i)$$

Dans wxMaxima, on peut générer des nombres décimaux aléatoires sur $[0, c]$ en entrant `random(c)`, en veillant à exprimer c avec une décimale (si c est exprimé comme un entier, `random` ne produit que des entiers).

Exemple 2.5.1. Calculez $\int_0^3 e^{\sin x}\ \mathrm{d}x$ en utilisant une méthode de Monte Carlo avec $N = 1000$. Répétez le calcul plusieurs fois pour vous faire une idée de l'incertitude liée à cette méthode d'approximation.

```
(%i1) f(x):=%e^(sin(x))$

(%i2) ((3-0)/1000)*sum(f(random(3.0)),i,1,1000);
(%o2) 6.095597917343273

(%i3) ((3-0)/1000)*sum(f(random(3.0)),i,1,1000);
(%o3) 6.105455491341729

(%i4) ((3-0)/1000)*sum(f(random(3.0)),i,1,1000);
(%o4) 6.014387493961544
```

En principe, nous pourrions effectuer une analyse statistique sur plusieurs itérations de cette méthode et utiliser l'écart type pour quantifier l'incertitude. Il est peut-être préférable de solliciter davantage notre ordinateur en utilisant $N = 1\,000\,000$:

```
(%i5) ((3-0)/1000000)*sum(f(random(3.0)),i,1,1000000);
(%o5) 6.057168102573812
```

Le résultat est très proche de la réponse que nous avons obtenue en utilisant `quad_qag`. Signalons cependant que mon ordinateur a mis près de deux minutes pour calculer cette approximation !

Exemple 2.5.2. Utilisez une méthode de Monte Carlo avec $N = 1000$ pour calculer $\int_2^3 e^{-x^2}\,\mathrm{d}x$. Comparez avec la réponse obtenue en utilisant `quad_qag`.

Afin d'utiliser `random` pour générer des nombres aléatoires sur $[2, 3]$, on part de $x = 2$ et on ajoute des nombres aléatoires compris entre 0 et 1,0 :

```
(%i6) f(x):=%e^(-x^2)$
(%i7) ((3-2)/1000)*sum(f(2+random(1.0)),i,1,1000);
(%o7) 0.0041199267231285
(%i8) quad_qag(%e^(-x^2),x,2,3,1);
(%o8) [0.0041259574970996,4.5908457058908545*10^-16,15,0]
```

La méthode de Monte Carlo a donné une précision de quelques chiffres significatifs en utilisant seulement un échantillon de 1000 points aléatoires.

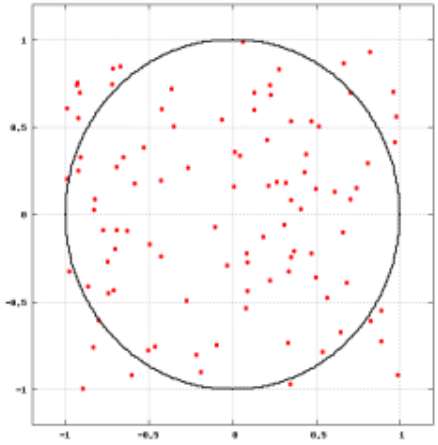
2.6 Exercices du Chapitre 2

1. Calculez $\int_1^e \cos(\ln x)\,\mathrm{d}x$ en utilisant une approximation du point médian avec $n = 100$. Construisez un graphique illustrant cette approximation dans le style de l'exemple 2.1.1. Vérifiez votre réponse à l'aide de `integrate`.
2. Recommencer l'exercice précédent en utilisant une approximation par les trapèzes dans le style de l'exemple 2.2.1.
3. Pour $f(x) = \cosh x$ et $g(x) = \cos x$, calculez les approximations par la méthode des trapèzes avec $n = 3$ sur $[0, 1]$, incluant des graphiques illustrant les trapèzes. Quelle approximation est une sous-estimation de la valeur de l'intégrale ? Quelle est une sur-estimation ? Comment les erreurs sont-elles liées à la concavité de f et g ?
4. Des mesures de tension sont effectuées pendant une période dans un circuit alternatif :

   ```
   t(s).....v(t)(V)
   0/1200      4.2
   1/1200     67.5
   2/1200    114.0
   3/1200    112.7
   4/1200     68.1
   5/1200     -0.8
   6/1200    -69.4
   7/1200   -110.7
   8/1200    114.1
   9/1200    -68.7
   10/1200    -4.6
   ```

   Calculez $\int_0^{\frac{10}{1200}} v^2(t)\,\mathrm{d}t$ en utilisant une approximation par trapèzes. Incluez un graphique pour illustrer l'approximation.
5. Répétez l'exercice précédent en utilisant une approximation par la méthode de Simpson. Incluez un graphique montrant les paraboles d'interpolation dans le style de l'exemple 2.3.3.
6. Calculez $\int_0^{\frac{\pi}{2}} \cos(\sin x)\,\mathrm{d}x$ en utilisant `quad_qag`. Produisez un graphique coloré illustrant l'aire représentée par l'intégrale.
7. Recommencez le calcul de l'intégrale de l'exercice précédent en utilisant la méthode de Simpson avec $n = 10$, dans le style de l'exemple 2.3.2. Réalisez un graphique pour illustrer cette approximation.
8. Recalculez l'intégrale de l'exercice précédent en utilisant une méthode de Monte Carlo avec 1000 points aléatoires.

9. * Nous pouvons utiliser une méthode de Monte Carlo pour approximer π. Pour visualiser la méthode, commencez par utiliser `makelist` pour générer 100 points aléatoires dans un carré de côté 2 centré à l'origine. Tracez ces points sur le même graphique que le cercle unité. Votre graphique être similaire à celui-ci :



L'aire du cercle est donnée par $A_{\text{cercle}} = \pi \cdot r^2 = \pi$ et l'aire du carré est $A_{\text{carré}} = 4$, donc on s'attend à ce que la proportion de points tombant à l'intérieur du cercle soit $\frac{A_{\text{cercle}}}{A_{\text{carré}}} = \frac{\pi}{4}$. Utilisez alors votre graphique pour trouver une approximation de π.

10. ** Pour appliquer la méthode de Monte Carlo à l'exemple précédent, nous voulons automatiser le processus de comptage des points qui se trouvent à l'intérieur du cercle unité. Cela peut être accompli en utilisant une boucle `for-do` pour générer un point à la fois et une instruction `if-then` pour tester si chaque point est à l'intérieur du cercle ou non. Avant l'exécution de la boucle, nous devons définir le point de départ d'un compteur `n:0`. Si un point est à l'intérieur du cercle, on ajoute 1 au compteur en écrivant `n:n+1`. Si nous incluons les points tombant exactement sur le bord du cercle, l'instruction `if-then` appropriée est :
`if (x^2+y^2<1 or x^2+y^2=1) then n:n+1`.
Enfin, nous utilisons une autre instruction `if-then` demandant à wxMaxima d'afficher la valeur de `n` après la dernière itération de la boucle.

Construisez la boucle `do` appropriée, en commençant par un petit nombre d'itérations jusqu'à ce que tout fonctionne correctement. Pendant vos expérimentations avec le programme, il est conseillé de demander à wxMaxima d'afficher `n` à chaque itération afin de faciliter la détection de tout problème dans le calcul. Une fois que vous êtes sûr que cela fonctionne, utilisez la boucle pour générer 10 000 points aléatoires. Utilisez la proportion de points à l'intérieur du cercle pour approximer π trois fois séparément afin d'avoir une idée de l'incertitude sur le résultat.

Chapitre 3

Applications géométriques de l'intégration

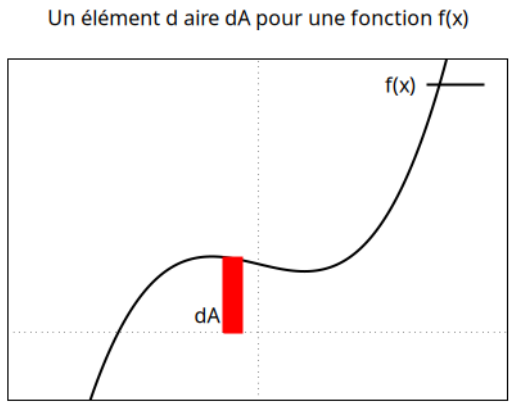
Commandes essentielles de ce chapitre

`rectangle`	`solve`	`kill(all)`
`integrate`	`find_root`	`diff`
`ratprint:false`	`wxdraw3d`	`quad_qag`
`float`	`parametric_surface`	
`filled_func`	`makelist`	

3.1 Intégrales et aires

3.1.1 Aire et intégration physique

Rappelons nous que $\int_a^b f(x)\,\mathrm{d}x$ donne l'aire délimitée entre $f(x)$ et l'axe des x. Il est utile de visualiser un *élément* d'aire en x qui est donc représenté par un mince rectangle de largeur $\mathrm{d}x$. Nous désignons cette fine tranche par $\mathrm{d}A$.



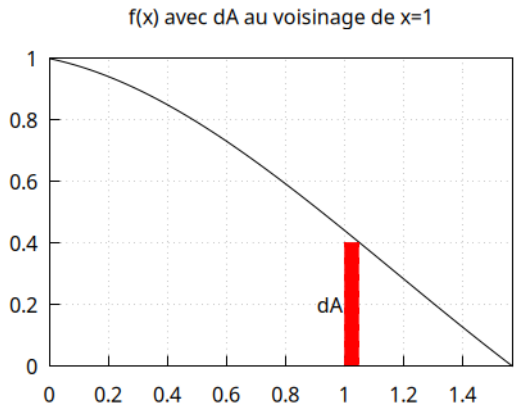
Une fois que $\mathrm{d}A$ est exprimé entièrement en fonction d'une seule variable (ici, $\mathrm{d}A = f(x)\cdot\mathrm{d}x$), on utilise l'intégration pour sommer tous les éléments d'aire. On peut écrire $A = \int \mathrm{d}A = \int_a^b f(x)\,\mathrm{d}x$, et on considère l'intégrale comme un *outil de sommation* permettant d'additionner les $\mathrm{d}A$. Cette approche conceptuelle est souvent appelée « intégration physique », et elle est très puissante lorsqu'on applique l'intégration dans un contexte géométrique.

Exemple 3.1.1. Tracer $f(x) = e^{-0,2\cdot x}\cdot\cos x$ sur $\left[0, \frac{\pi}{2}\right]$ ainsi qu'un élément d'aire en $x = 1$. Définir l'intégrale $A = \int \mathrm{d}A$ et exprimer $\mathrm{d}A$ en fonction de x afin d'obtenir une intégrale définie. Enfin, utiliser wxMaxima pour calculer l'aire et produire un graphique comportant l'aire colorée.

On commence par représenter une esquisse de $f(x)$ et de $\mathrm{d}A$:

```
(%i1) f(x):=%e^(-0.2*x)*cos(x)$
      wxdraw2d(
      grid=true,
      xaxis=true,
      yaxis=true,
      title="f(x) avec dA au voisinage de x=1",
      color=black,
       explicit(f(x),x,0,%pi/2),
      border=false,
```

```
color=red,
 rectangle([1,0],[1.05,f(1.05)]),
color=black,
 label(["dA",0.95,0.2])
);
```



dA est simplement le produit de la hauteur et de la largeur de l'élément d'aire : $f(x) \cdot \mathrm{d}x$.
On définit alors l'intégrale : $A = \int \mathrm{d}A = \int f(x)\ \mathrm{d}x = \int_0^{\frac{\pi}{2}} e^{-0,2 \cdot x} \cdot \cos x\ \mathrm{d}x$

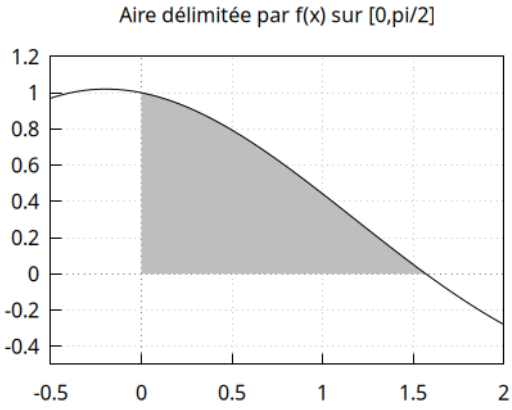
Le calcul de l'intégrale est simple à effectuer avec `integrate`. On commence par `ratprint:false` pour supprimer les avertissements `ratprint`, et on obtient une approximation décimale à l'aide de `float` :

```
(%i2) ratprint:false$
(%i3) integrate(f(x),x,0,%pi/2);
(%o3) (25*%e^(-%pi/10))/26+5/26
(%i4) float(%);
(%o4) 0.89461797216216
```

Nous terminons en réalisant un graphique comportant l'aire colorée :

```
(%i5) wxdraw2d(
      grid=true,
      xrange=[-0.5,2],
      yrange=[-0.5,1.2],
      xaxis=true,
      yaxis=true,
      title="Aire délimitée par f(x) sur [0,pi/2]",
      fill_color=grey,
      filled_func=true,
       filled_func=f(x),
       explicit(0,x,0,%pi/2),
      filled_func=false,
      color=black,
```

```
 explicit(f(x),x,-0.5,2)
);
```



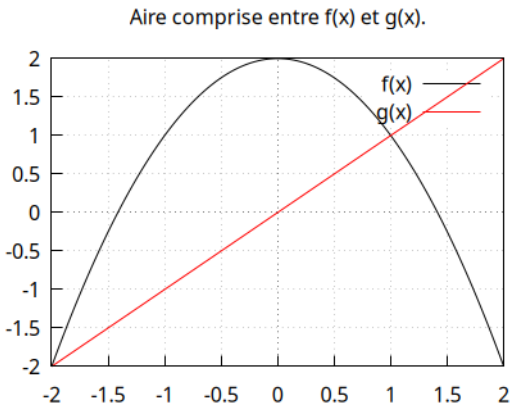
3.1.2 Aire comprise entre deux fonctions

Pour calculer l'aire comprise *entre* deux fonctions, il suffit de calculer un élément d'aire et d'exprimer $\mathrm{d}A$ à l'aide des fonctions données. On utilise ensuite l'intégrale $A = \int \mathrm{d}A$ pour sommer tous les éléments d'aire sur l'intervalle approprié.

Exemple 3.1.2. Calculer l'aire comprise entre $f(x) = 2 - x^2$ et $g(x) = x$.

Nous commençons par une esquisse de la situation :

```
(%i1) f(x):=2-x^2$
      g(x):=x$
(%i3) wxdraw2d(
      grid=true,
      xaxis=true,
      yaxis=true,
      xrange=[-2,2],
      yrange=[-2,2],
      title="Aire comprise entre f(x) et g(x).",
      color=black,
      key="f(x)",
       explicit(f(x),x,-2,2),
      color=red,
      key="g(x)",
      explicit(g(x),x,-2,2)
      );
```

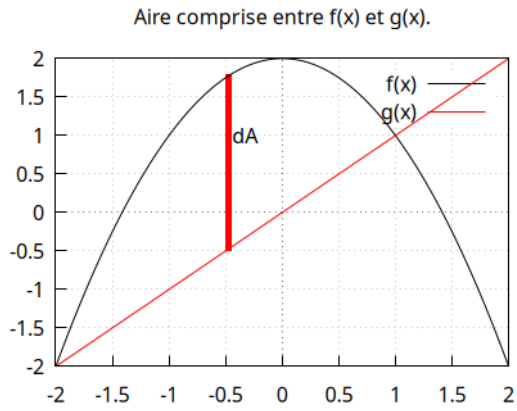


On détermine ensuite l'intervalle d'intégration approprié : l'aire délimitée se situe de manière évidente entre les deux points d'intersection des courbes.

```
(%i4) solve(f(x)=g(x),x);
(%o4) [x=1,x=-2]
```

Enfin, on peut visualiser un élément d'aire sur l'intervalle d'intégration et écrire la formule donnant sa valeur. Ici, on utilise un mince rectangle pour représenter un élément d'aire en $x = -0{,}5$:

```
(%i5) wxdraw2d(
      grid=true,
      xaxis=true,
      yaxis=true,
      xrange=[-2,2],
      yrange=[-2,2],
      title="Aire comprise entre f(x) et g(x).",
      color=red,
      border=false,
       rectangle([-0.5,g(-0.5)],[-0.45,f(-0.45)]),
      color=black,
       label(["dA",-0.33,1]),
      key="f(x)",
       explicit(f(x),x,-2,2),
      color=red,
      key="g(x)",
       explicit(g(x),x,-2,2)
      );
```

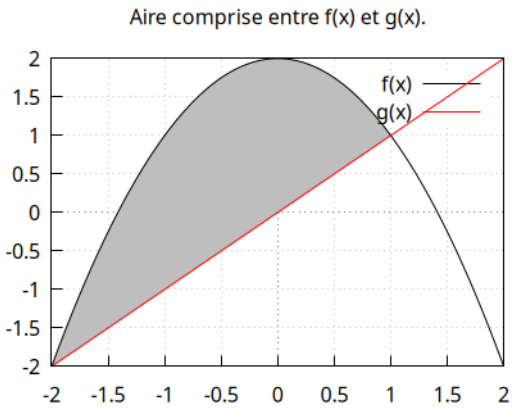


On constate que l'élément d'aire a une hauteur de $f(x) - g(x)$, ce qui donne $\mathrm{d}A = [f(x) - g(x)] \cdot \mathrm{d}x$. On somme les éléments d'aire en calculant l'intégrale $A = \int \mathrm{d}A = \int_{-2}^{1} [f(x) - g(x)]\ \mathrm{d}x$:

```
(%i6) float(integrate((f(x)-g(x)),x,-2,1));
(%o6) 4.5
```

Nous obtenons $A = 4.5$ unités d'aire. Nous pouvons aussi réaliser un graphique mettant en évidence l'aire que nous avons calculée :

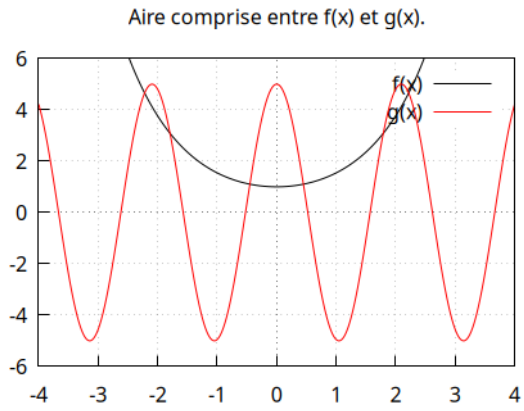
```
(%i7) wxdraw2d(
      grid=true,
      xaxis=true,
      yaxis=true,
      xrange=[-2,2],
      yrange=[-2,2],
      title="Aire comprise entre f(x) et g(x).",
      fill_color=grey,
       filled_func=true,
        filled_func=f(x),
        explicit(g(x),x,-2,1),
       filled_func=false,
     color=black,
     key="f(x)",
      explicit(f(x),x,-2,2),
     color=red,
     key="g(x)",
      explicit(g(x),x,-2,2)
     );
```



Exemple 3.1.3. Calculer l'aire entre $f(x) = \cosh x$ et $g(x) = 5\cos 3x$.

Nous commençons par réaliser une esquisse de la situation :

```
(%i8) f(x):=cosh(x)$
      g(x):=5*cos(3*x)$
      wxdraw2d(
      grid=true,
      xaxis=true,
      yaxis=true,
      xrange=[-4,4],
      yrange=[-6,6],
      title="Aire comprise entre f(x) et g(x).",
      color=black,
      key="f(x)",
       explicit(f(x),x,-4,4),
      color=red,
      key="g(x)",
      explicit(g(x),x,-4,4)
      );
```



On constate que les deux fonctions sont paires, il suffit donc de calculer l'aire à droite de $x = 0$ et de doubler le résultat. Pour vérifier que les fonctions sont paires :

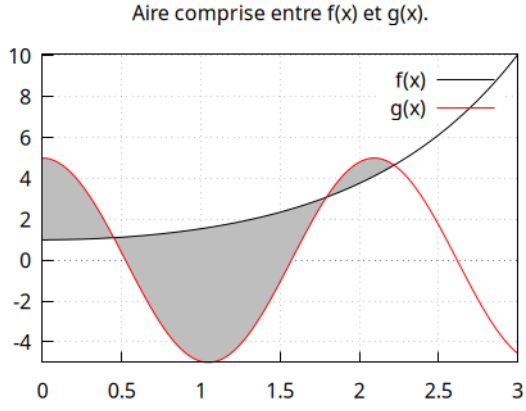
```
(%i9) f(-x);
      g(-x);
(%o9) cosh(x)
(%o10) 5*cos(3*x)
```

Il faut garder à l'esprit que l'on ne calcule plus une aire *algébrique* mais une aire *géométrique*, qui est toujours positive. Ainsi, lorsque $f(x)$ est au-dessus de $g(x)$, l'élément d'aire est $[f(x) - g(x)] \cdot \mathrm{d}x$, mais lorsque $g(x)$ est au-dessus de $f(x)$, l'élément d'aire devient $[g(x) - f(x)] \cdot \mathrm{d}x$. On doit donc découper l'intervalle d'intégration en trois intervalles selon les abscisses des points d'intersection de $f(x)$ et $g(x)$, puis additionner les aires et multiplier par 2 :

```
(%i11) x1:find_root(f(x)-g(x),0,1);
       x2:find_root(f(x)-g(x),1.5,2);
       x3:find_root(f(x)-g(x),2,2.5);
(%o11) 0.44947432956866
(%o12) 1.792473165779467
(%o13) 2.219127938640189
(%i14) ratprint:false$
       A1:float(integrate(g(x)-f(x),x,0,x1));
       A2:float(integrate(f(x)-g(x),x,x1,x2));
       A3:float(integrate(g(x)-f(x),x,x2,x3));
(%o14) 1.160865665363456
(%o15) 5.39123022939955
(%o16) 0.29427502144913
(%i17) A1+A2+A3;
(%o17) 6.846370916212132

(%i18) %*2;
(%o18) 13.69274183242426
```

Nous obtenons une aire géométrique totale de $A \approx 13.7$. Nous terminons avec une représentation graphique des courbes et de l'aire sur la partie correspondant aux abscisses positives :

```
(%i19) wxdraw2d(
       grid=true,
       xaxis=true,
       yaxis=true,
       xrange=[0,3],
       title="Aire comprise entre f(x) et g(x).",
       fill_color=grey,
        filled_func=true,
         filled_func=g(x),
         explicit(f(x),x,0,x1),
        filled_func=false,
        filled_func=true,
         filled_func=f(x),
         explicit(g(x),x,x1,x2),
        filled_func=false,
        filled_func=true,
         filled_func=g(x),
         explicit(f(x),x,x2,x3),
         filled_func=false,
        color=black,
        key="f(x)",
         explicit(f(x),x,0,3),
        color=red,
        key="g(x)",
         explicit(g(x),x,0,3)
        );
```



3.2 Solides de révolution

Un **solide de révolution** est un objet tridimensionnel créé en faisant tourner le graphe d'une fonction $f(x)$ autour d'un axe (généralement l'axe des x ou des y). Les solides de révolution possèdent une *symétrie cylindrique*, ce qui rend leur volume relativement facile à calculer. Nous utilisons une approche par une intégration physique en calculant un petit élément de volume $\mathrm{d}V$ et en sommant ces éléments à l'aide d'une intégrale : $V = \int \mathrm{d}V$.

La symétrie cylindrique d'un solide de révolution nous offre deux principales options pour choisir $\mathrm{d}V$: la méthode des « disques/rondelles » et la méthode des « coquilles cylindriques ».

Remarque : dans les exemples suivants, nous utilisons des ensembles d'équations *paramétriques* pour tracer des solides de révolution avec `wxdraw3d`. Les graphiques 3D constituent un outil utile pour la visualisation, mais les mathématiques nécessaires pour les tracer correspondent davantage aux étudiants ayant suivi un cours d'analyse multivariable. Nous incluons cependnat tout le code par souci d'exhaustivité.

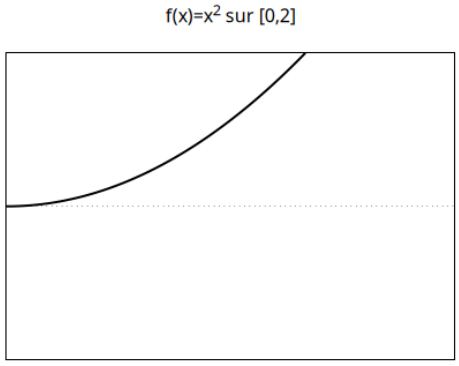
3.2.1 Disques et Rondelles

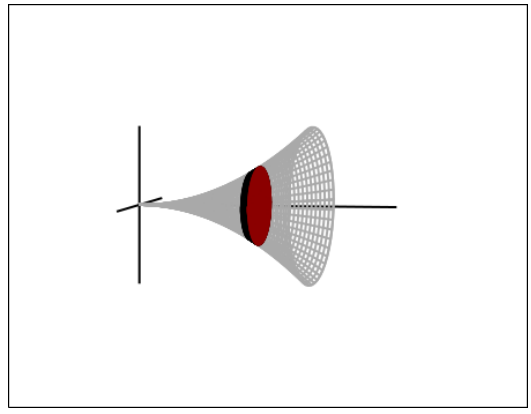
La méthode des « disques/rondelles » utilise des éléments de volume constitués de fines tranches cylindriques. Les tranches minces sont des disques lorsque l'objet est plein, et des « rondelles » lorsque l'objet présente un trou le long de l'axe de symétrie.

Exemple 3.2.1. Tracer $f(x) = x^2$ et le solide de révolution obtenu en faisant tourner $f(x)$ sur $[0, 2]$ autour de l'axe des x. Inclure un disque $\mathrm{d}V$ dans le graphique. Enfin, exprimer $\mathrm{d}V$ en fonction d'une seule variable et utiliser une intégrale pour sommer les éléments de volume.

```
(%i1) wxdraw2d(
      dimensions=[600,600],
      xrange=[0,3],
      yrange=[-4,4],
      xaxis=true,
      yaxis=true,
      xtics=false,
      ytics=false,
      line_width=2,
      title="f(x)=x^2 sur [0,2]",
      color=black,
      explicit(x^2,x,0,2)
      );
(%i2) wxdraw3d(
      axis_3d=false,
      dimensions=[600,600],
      view=[85,10],
      xrange=[0,3],
      yrange=[-4,4],
      zrange=[-4,4],
      xtics=false,
```

```
ytics=false,
ztics=false,
color=black,
nticks=600,
line_width=2,
 parametric(t,0,0,t,0,3),
 parametric(0,t,0,t,-4,4),
 parametric(0,0,t,t,-4,4),
color=dark_grey,
 parametric_surface(r,r^2*cos(t),r^2*sin(t),r,0,2,t,0,2*%pi),
color=black,
 parametric_surface(r,r^2*cos(t),r^2*sin(t),r,1.3,1.4,t,0,2*%pi),
color=dark_red,
 parametric_surface(1.4,u^2*cos(t),u^2*sin(t),u,0,1.4,t,0,2*%pi)
);
```





Bien que l'axe z pointe techniquement vers le haut dans cette figure, la symétrie du graphique nous permet de considérer l'axe vertical comme étant y. La surface courbe est « balayée » lorsque l'on fait tourner $y = x^2$ autour de l'axe x.

$\mathrm{d}V$ est représenté en noir ; on dit qu'il possède une épaisseur (horizontale) $\mathrm{d}x$ et un rayon x^2 pour la position particulière x du disque (x^2 est la distance entre l'axe x et un point sur la courbe d'origine). Le volume du disque est le produit de l'aire de la base par l'épaisseur, donc $\mathrm{d}V = \pi \cdot (x^2)^2 \cdot \mathrm{d}x$. Nous pouvons maintenant établir l'intégrale de volume avec les bornes d'intégration correspondant à x :

$$V = \int \mathrm{d}V = \int_0^2 \pi \cdot x^4 \, \mathrm{d}x$$

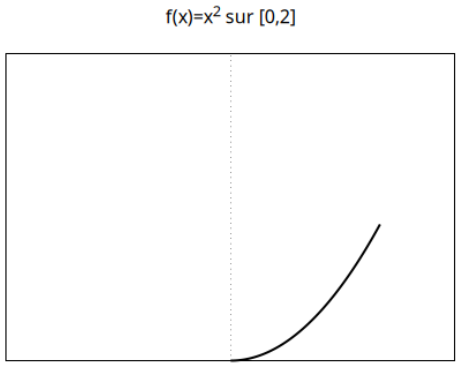
Cette intégrale n'est pas trop difficile à calculer à la main, mais nous utilisons wxMaxima pour nous entraîner.

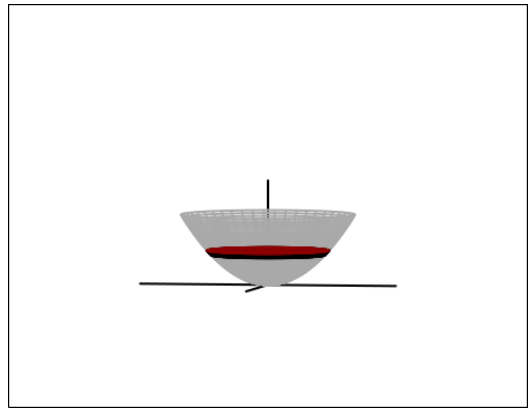
```
(%i3) integrate(%pi*x^4,x,0,2);
(%o3) (32*%pi)/5
(%i4) float(%);
(%o4) 20.10619298297468
```

Le volume est donc égal à $V = \frac{32\pi}{5} \approx 20.1$ unités de volume.

Exemple 3.2.2. Tracer $f(x) = x^2$ et le solide de révolution obtenu en faisant tourner $f(x)$ sur $[0, 2]$ autour de l'axe y. Dessiner un disque dV dans le graphique. Enfin, exprimer dV en fonction d'une seule variable et utiliser une intégrale pour sommer les éléments de volume.

```
(%i5) wxdraw2d(
       dimensions=[600,600],
       xrange=[-3,3],
       yrange=[0,9],
       xaxis=true,
       yaxis=true,
       xtics=false,
       ytics=false,
       line_width=2,
       title="f(x)=x^2 sur [0,2]",
       color=black,
       explicit(x^2,x,0,2)
       );

(%i6) wxdraw3d(
      axis_3d=false,
      dimensions=[600,600],
      view=[85,10],
      xrange=[-3,3],
      yrange=[-3,3],
      zrange=[0,9],
      xtics=false,
      ytics=false,
      ztics=false,
      color=black,
      nticks=600,
      line_width=2,
       parametric(t,0,0,t,-3,3),
       parametric(0,t,0,t,-3,3),
       parametric(0,0,t,t,0,6),
      color=dark_grey,
       parametric_surface(r*cos(t),r*sin(t),r^2,r,0,2,t,0,2*%pi),
      color=black,
       parametric_surface(r*cos(t),r*sin(t),r^2,r,1.3,1.4,t,0,2*%pi),
      color=dark_red,
       parametric_surface(r*cos(t),r*sin(t),1.4^2,r,0,1.4,t,0,2*%pi)
      );
```





Là encore, il est plus facile de produire un graphique avec $f(x)$ en rotation autour de l'axe z, mais nous pouvons le considérer comme l'axe y : nous avons produit le graphique uniquement pour guider notre raisonnement. Cette fois, l'épaisseur du disque est $\mathrm{d}y$. Le disque est situé à une position verticale de y, avec un rayon donné par $x = \sqrt{y}$, donc le volume du disque est $\mathrm{d}V = \pi \cdot (\sqrt{y})^2 \cdot \mathrm{d}y$. Nous calculons l'intégrale de volume avec les bornes d'intégration correspondant à y (lorsque x varie de 0 à 2, $y = x^2$ varie de 0 à 4) :

$$V = \int \mathrm{d}V = \int_0^4 \pi \cdot y \ \mathrm{d}y$$

Là encore, l'intégrale est simple à calculer à la main, mais nous choisissons d'utiliser wxMaxima :

```
(%i7) integrate(y,y,0,4);
(%o7) 8
```

La valeur du volume est donc de $V = 8$ unités de volume.

Exemple 3.2.3. Tracer la région délimitée entre $f(x) = 0{,}3 \cdot x$ et $g(x) = \sin x$, puis représenter le solide de révolution obtenu en faisant tourner cette région autour de l'axe x, en incluant un élément de volume « rondelle » $\mathrm{d}V$. Enfin, établir et calculer l'intégrale de volume pour le solide.

Le tracé de l'aire délimitée entre $f(x)$ et $g(x)$ nécessite de trouver les intersections en résolvant l'équation $0{,}3 \cdot x = \sin x$ avec `find_root`. Nous utilisons la symétrie pour en déduire la seconde intersection à partir de la première. Nous traçons l'aire en utilisant `filled_func` pour la partie colorée :

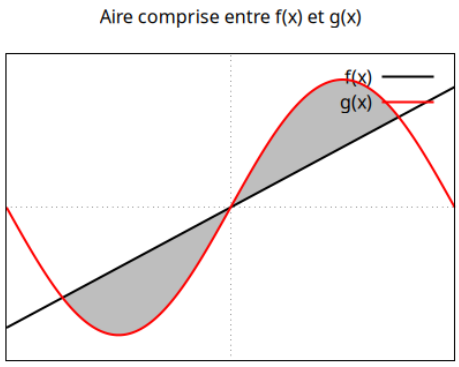
```
(%i8) kill(all)$

(%i1) f(x):=0.3*x$
      g(x):=sin(x)$
(%i3) find_root(f(x)-g(x),x,1,4);
(%o3) 2.356441149856161

(%i4) wxdraw2d(
      xaxis=true,
```

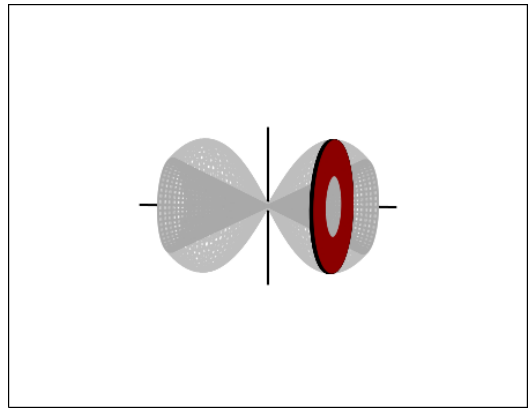
```
 yaxis=true,
 xtics=false,
 ytics=false,
 dimensions=[600,600],
 xrange=[-%pi,%pi],
 yrange=[-1.2,1.2],
 title="Aire comprise entre f(x) et g(x)",
 fill_color=grey,
  filled_func=true,
   filled_func=f(x),
   explicit(g(x),x,-2.356,0),
  filled_func=false,
  filled_func=true,
   filled_func=g(x),
   explicit(f(x),x,0,2.356),
  filled_func=false,
 line_width=2,
 color=black,
 key="f(x)",
  explicit(f(x),x,-%pi,%pi),
 color=red,
 key="g(x)",
  explicit(g(x),x,-%pi,%pi)
 );
```

Là encore, le tracé du solide de révolution est plus compréhensible pour les étudiants en analyse multivariable, mais nous incluons le code à titre de référence :

```
(%i5)wxdraw3d(
     axis_3d=false,
     dimensions=[600,600],
     view=[85,10],
     xrange=[-%pi,%pi],
     yrange=[-1.2,1.2],
     zrange=[-1.2,1.2],
     xtics=false,
     ytics=false,
     ztics=false,
     color=black,
     nticks=200,
     line_width=2,
      parametric(t,0,0,t,-%pi,%pi),
      parametric(0,t,0,t,-1.2,1.2),
      parametric(0,0,t,t,-1.2,1.2),
     color=grey,
     transparent=true,
      parametric_surface(r, g(r)*sin(t),g(r)*cos(t),r,-2.356,2.356,t,
      0,2*%pi),
     transparent=false,
     color=dark_grey,
```

```
 parametric_surface(r, f(r)*sin(t), f(r)*cos(t),r,-2.356, 2.356,t,
 0,2*%pi),
color=black,
 parametric_surface(r, g(r)*sin(t), g(r)*cos(t),r,1.5,1.6,t,0,2*%pi),
color=dark_red,
 parametric_surface(1.6, u*sin(t),u*cos(t),u,f(1.6),g(1.6),t,0,2*%pi)
);
```





Pour le calcul du volume, nous pouvons simplement trouver le volume de la moitié droite et le multiplier par 2. Le volume de la « rondelle » est toujours donné par le produit de l'aire et de l'épaisseur, et l'aire de la base est simplement égale à l'aire du cercle extérieur moins l'aire du cercle intérieur : $A = \pi \cdot r^2_{\text{extérieur}} - \pi \cdot r^2_{\text{intérieur}}$. Ainsi $\mathrm{d}V = \left(\pi \cdot (\sin x)^2 - \pi \cdot (0{,}3 \cdot x)^2\right) \cdot \mathrm{d}x$. Enfin, nous calculons l'intégrale de volume :

$$V = \int \mathrm{d}V = 2 \cdot \int_0^{2{,}356} \pi \cdot (\sin x)^2 - \pi \cdot (0{,}3 \cdot x)^2 \ \mathrm{d}x$$

Calculons l'intégrale à l'aide de wxMaxima :

```
(%i6) ratprint:false$
(%i7) float(2*integrate(%pi*(sin(x))^2-%pi*(0.3*x)^2, x, 0, 2.356));
(%o7) 6.507331412313022
```

Nous obtenons comme valeur $V \approx 6.5$ unités de volume.

3.2.2 Coques cylindriques

Un solide de révolution peut également être décomposé en éléments de volume sous forme de coques cylindriques imbriquées. Le volume d'une coque cylindrique de hauteur h, de rayon r et d'épaisseur dr est obtenu en « déroulant » la coque en une plaque rectangulaire. Lorsqu'une coque est déroulée, la longueur est $2\pi r$ (la circonférence de la coque d'origine). Ainsi, on utilise comme élément de volume $\mathrm{d}V = 2\pi r \cdot h \cdot \mathrm{d}r$ chaque fois que l'on décompose un solide en coques cylindriques.

Exemple 3.2.4. Tracer l'aire délimitée par $f(x) = \cosh x$ sur $[-2, 2]$, puis représenter le solide obtenu en faisant tourner cette aire autour de l'axe y. Inclure une représentation d'un élément de volume sous forme de fine coque cylindrique. Enfin, établir et calculer l'intégrale de volume.

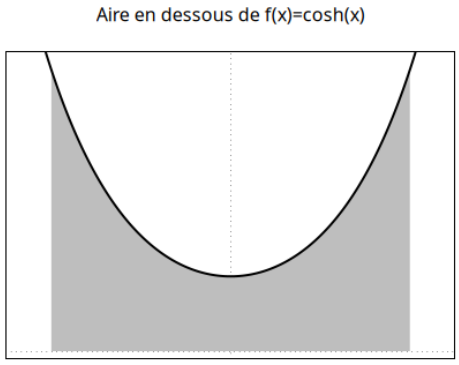
Nous commençons par définir $f(x)$ et nous représentons l'aire concernée en la colorant :

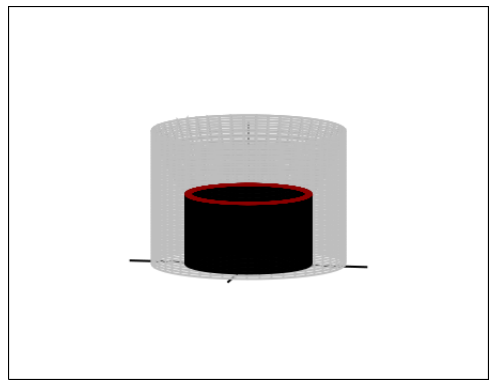
```
(%i8) kill(all)$
(%i1) f(x):=cosh(x)$
      wxdraw2d(
      xaxis=true,
      yaxis=true,
      xtics=false,
      ytics=false,
      dimensions=[600,600],
      xrange=[-2.5,2.5],
      yrange=[-0.1,4],
      title="Aire en dessous de f(x)=cosh(x)",
      fill_color=grey,
       filled_func=true,
        filled_func=f(x),
        explicit(0,x,-2.0,2.0),
       filled_func=false,
      line_width=2,
      color=black,
       explicit(f(x),x,-2.5,2.5)
      );
```

Comme précédemment, le code pour la réprésentation en 3d est donné à titre de référence :

```
(%i3) wxdraw3d(
      axis_3d=false,
      dimensions=[600,600],
      view=[75,10],
      xrange=[-2.5,2.5],
      yrange=[-2.5,2.5],
      zrange=[0,4],
      xtics=false,
      ytics=false,
      ztics=false,
      color=black,
```

```
nticks=600,
line_width=2,
 parametric(t,0,0,t,-2.5,2.5),
 parametric(0,t,0,t,-2.5,2.5),
 parametric(0,0,t,t,-0.1,4),
color=dark_grey,
 parametric_surface(r*cos(t),r*sin(t),cosh(r),r,0,2,t,0,2*%pi),
color=grey,
 parametric_surface(2*cos(t),2*sin(t),u,t,0,2*%pi,u,0,cosh(2)),
color=black,
 parametric_surface(1.3*cos(t),1.3*sin(t),u,u,0,cosh(1.3),t,0,2*%pi),
color=dark_red,
 parametric_surface(r*cos(t),r*sin(t),cosh(1.3),r,1.2,1.3,t,0,2*%pi)
);
```





Le rayon de la coque cylindrique est x, la hauteur est $\cosh x$ et l'épaisseur est $\mathrm{d}x$. Lorsque l'on déroule la coque, on obtient une plaque de longueur $2\pi x$, de hauteur $\cosh x$ et d'épaisseur $\mathrm{d}x$, donc $\mathrm{d}V = 2\pi x \cdot \cosh x \ \mathrm{d}x$. Nous pouvons maintenant exprimer l'intégrale de volume et terminer le calcul :

$$V = \int \mathrm{d}V = \int_0^2 2\pi x \cdot \cosh x \ \mathrm{d}x$$

```
(%i4) float(integrate(2*%pi*x*cosh(x),x,0,2));
(%o4) 28.22108466978007
```

Nous obtenons comme valeur pour le volume $V \approx 28.2$ en unités de volume.

3.3 Longueur d'un arc

3.3.1 En tant que processus limite

La longueur d'un arc d'une courbe $f(x)$ peut être vue comme un processus limite : on divise $[a, b]$ en n sous-intervalles aux points $x_0, x_1, ..., x_n$, puis on calcule les longueurs des segments de droite reliant tous les points adjacents sur la courbe $(x_i, f(x_i))$ et $(x_{i+1}, f(x_{i+1}))$. Enfin, on additionne les longueurs de tous les segments de droite pour obtenir une approximation de la longueur d'arc de $f(x)$ sur $[a, b]$. Il est clair que plus n est grand, plus l'approximation est précise.

Exemple 3.3.1. Tracer les approximations de la longueur d'arc pour $n = 2$, $n = 5$, $n = 10$ et $n = 20$ pour $f(x) = \dfrac{e^x}{x^3}$ sur $[1, 5]$. Calculer la valeur numérique de chaque approximation. Enfin, améliorer la précision de l'approximation en calculant la longueur d'arc pour $n = 1000$.

Tout d'abord, nous définissons $f(x)$, ainsi qu'une fonction `LINE(a,b,c,d)` calculant l'équation de la droite reliant deux points quelconques (a, b) et (c, d), et une fonction `LENGTH(a,b,c,d)` pour calculer la longueur du segment de droite reliant (a, b) et (c, d) :

```
(%i1) f(x):=exp(x)/(x^3)$
      SLOPE(a,b,c,d):=(d-b)/(c-a)$
      LINE(a,b,c,d):=SLOPE(a,b,c,d)*(x-a)+b$
      LENGTH(a,b,c,d):=sqrt((c-a)^2+(d-b)^2)$
```

L'étape suivante consiste à définir x_i comme une fonction de i et n. Nous utilisons $\Delta x = (5 - 1)/n$:

```
(%i5) delx(n):=(5-1)/n$
      x(n,i):=1+(i)*delx(n)$
```

Nous assignons respectivement à n les valeurs demandées et utilisons `makelist` pour générer des segments de droite dans le format correct pour le traitement par `wxdraw2d` :

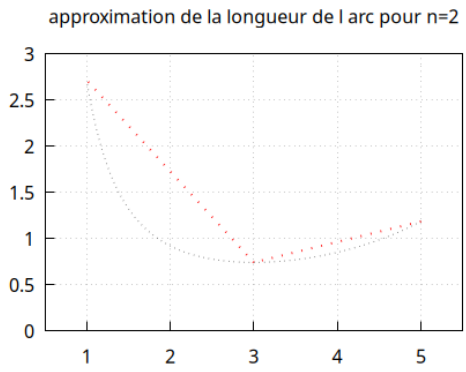
```
(%i7) n:2$
      SEGMENTS2:makelist(explicit(LINE(x(n,i),f(x(n,i)),
      x(n,i+1),f(x(n,i+1))),x,x(n,i),x(n,i+1)),i,0,n-1)$
(%i9) n:5$
      SEGMENTS5:makelist(explicit(LINE(x(n,i),f(x(n,i)),
      x(n,i+1),f(x(n,i+1))),x,x(n,i),x(n,i+1)),i,0,n-1)$
(%i11) n:10$
       SEGMENTS10:makelist(explicit(LINE(x(n,i),f(x(n,i)),
       x(n,i+1),f(x(n,i+1))),x,x(n,i),x(n,i+1)),i,0,n-1)$
(%i13) n:20$
       SEGMENTS20:makelist(explicit(LINE(x(n,i),f(x(n,i)),
       x(n,i+1),f(x(n,i+1))),x,x(n,i),x(n,i+1)),i,0,n-1)$
```

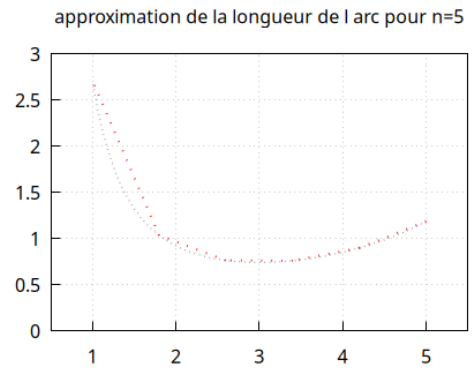
Finalement, nous réalisons un graphique pour chaque approximation :

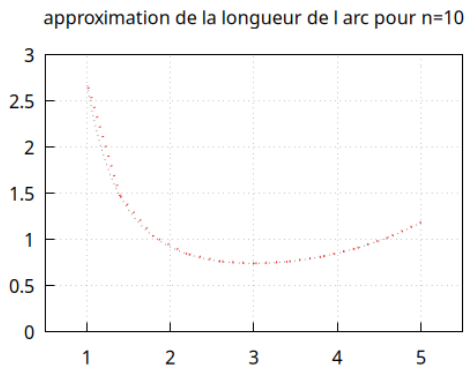
```
(%i15) wxdraw2d(
       grid=true,
```

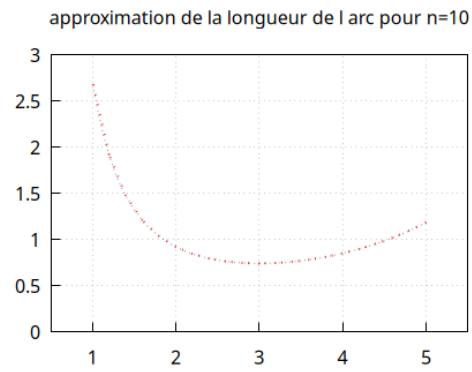
```
       xaxis=true,
       yaxis=true,
       xrange=[0.5,5.5],
       yrange=[0,3],
       title="approximation de la longueur de l arc pour n=2",
       color=black,
       line_type=dots,
        explicit(f(x),x,1,5),
       line_width=2,
       color=red,
        SEGMENTS2
       );

(%i16) wxdraw2d(
       grid=true,
       xaxis=true,
       yaxis=true,
       xrange=[0.5,5.5],
       yrange=[0,3],
       title="approximation de la longueur de l arc pour n=5",
       color=black,
       line_type=dots,
        explicit(f(x),x,1,5),
       line_width=2,
       color=red,
        SEGMENTS5
       );

(%i17) wxdraw2d(
       grid=true,
       xaxis=true,
       yaxis=true,
       xrange=[0.5,5.5],
       yrange=[0,3],
       title="approximation de la longueur de l arc pour n=10",
       color=black,
       line_type=dots,
        explicit(f(x),x,1,5),
       line_width=2,
       color=red,
        SEGMENTS10
       );

(%i18) wxdraw2d(
       grid=true,
       xaxis=true,
       yaxis=true,
       xrange=[0.5,5.5],
       yrange=[0,3],
```

```
title="approximation de la longueur de l arc pour n=10",
color=black,
line_type=dots,
 explicit(f(x),x,1,5),
line_width=2,
color=red,
 SEGMENTS20
);
```









Nous pouvons maintenant calculer les valeurs numériques de toutes les approximations, y compris pour $n = 1000$:

```
(%i19) n:2$
       APPROX2:float(sum(LENGTH(x(n,i),f(x(n,i)),x(n,i+1),f(x(n,i+1))),
       i,0,n-1));

(%o20) 4.858925140077406

(%i21) n:5$
       APPROX5:float(sum(LENGTH(x(n,i),f(x(n,i)),x(n,i+1),f(x(n,i+1))),
       i,0,n-1));

(%o22) 5.168139881862706
```

```
(%i23) n:10$
       APPROX10:float(sum(LENGTH(x(n,i),f(x(n,i)),x(n,i+1),f(x(n,i+1))),
       i,0,n-1));

(%o24) 5.213749217721892

(%i25) n:20$
       APPROX20:float(sum(LENGTH(x(n,i),f(x(n,i)),x(n,i+1),f(x(n,i+1))),
       i,0,n-1));

(%o26) 5.224566917514401

(%i27) n:1000$
       APPROX1000:float(sum(LENGTH(x(n,i),f(x(n,i)),x(n,i+1),f(x(n,i+1))),
       i,0,n-1));

(%o28) 5.228147830313162
```

Nous constatons que la longueur d'arc tend vers une valeur limite d'environ 5,23.

Exemple 3.3.2. Estimer π en utilisant une approximation de longueur d'arc avec $n = 1000$ pour calculer la longueur d'arc d'un demi-cercle de rayon 1. Quel pourcentage d'erreur obtient-on ?

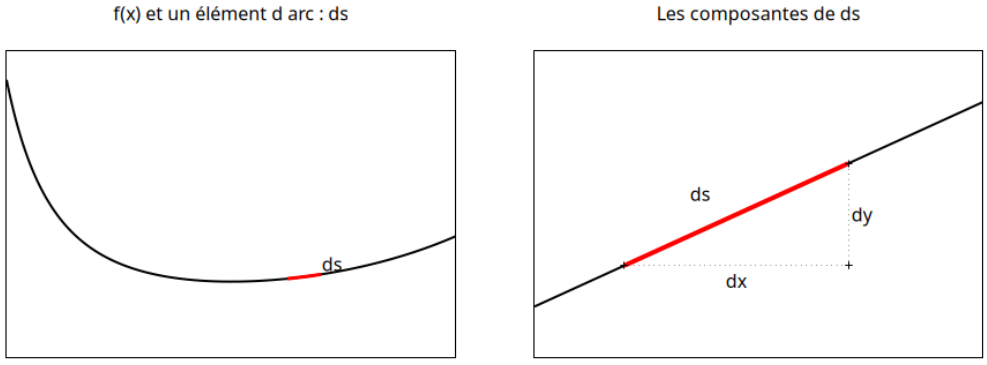
La fonction correspondante à la moitié supérieure du cercle unité est $f(x) = \sqrt{1 - x^2}$. La circonférence du cercle unité est $2\pi r = 2\pi$, donc le demi-cercle doit avoir une longueur d'arc de π. Nous copions le code fonctionnel du dernier exemple, en utilisant l'intervalle $[-1, 1]$, de sorte que `delx(n)=2/n` et `x(n,i)` commence à -1 :

```
(%i29) kill(all)$

       f(x):=sqrt(1-x^2)$
       LENGTH(a,b,c,d):=sqrt((c-a)^2+(d-b)^2)$
       delx(n):=2/n$
       x(n,i):=-1+(i)*delx(n)$
       n:1000$
       float(sum(LENGTH(x(n,i),f(x(n,i)),x(n,i+1),f(x(n,i+1))),i,0,n-1));
(%o6) 3.141566356216483

(%i7) float(100*(%-%pi)/%pi);
(%o7) -8.3707139052572146*10^-4
```

Notre approximation de π ne s'écarte que de 0,0008% de la valeur exacte.

3.3.2 En tant qu'intégrale physique

Nous pouvons également aborder le problème de la longueur d'arc de manière symbolique en visualisant un petit élément de longueur d'arc ds sur le graphe de $f(x)$:



Lorsque l'on zoome sur ds, on constate qu'il peut être décomposé selon ses composantes sur x et y, et l'on applique le théorème de Pythagore pour obtenir $\mathrm{d}s = \sqrt{(\mathrm{d}x)^2 + (\mathrm{d}y)^2}$.

Enfin, on factorise dx dans cette expression pour obtenir $\mathrm{d}s = \sqrt{1 + \left(\frac{\mathrm{d}y}{\mathrm{d}x}\right)^2} \cdot \mathrm{d}x$, et l'on exprime la longueur d'arc comme une intégrale :

$$S = \int \mathrm{d}s = \int_a^b \sqrt{1 + (f'(x))^2}\ \mathrm{d}x$$

En pratique, cette intégrale peut rarement être calculée à la main, mais il est simple d'en obtenir une approximation numérique avec wxMaxima.

Exemple 3.3.3. Calculer la longueur d'arc de $f(x) = \dfrac{e^x}{x^3}$ sur $[1, 5]$ et comparer avec l'approximation pour $n = 1000$ obtenue à l'Exemple 3.3.1.

```
(%i8) f(x):=%e^x/(x^3);
(%o8) f(x):=%e^x/x^3
(%i9) f_prime:diff(f(x),x);
(%o9) %e^x/x^3-(3*%e^x)/x^4
(%i10) integrate(sqrt(1+(f_prime)^2),x,1,5);
(%o10) integrate(sqrt((%e^x/x^3-(3*%e^x)/x^4)^2+1),x,1,5)
```

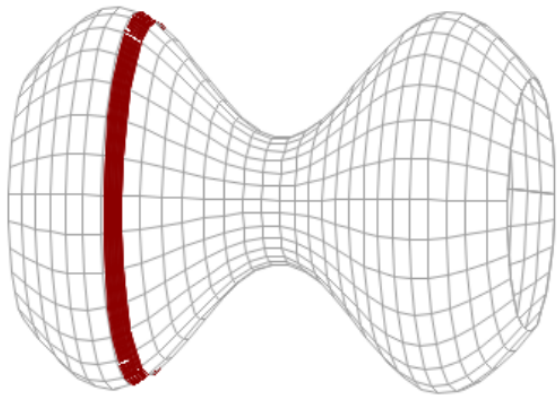
wxMaxima nous renvoie l'intégrale telle quelle, indiquant qu'il ne peut pas trouver de solution analytique. Nous utilisons `quad_qag` pour obtenir une approximation :

```
(%i36) quad_qag(sqrt(1+(f_prime)^2),x,1,5,1);
(%o36) [5.228149260997872,6.6513611026699459*10^-9,75,0]
```

Nous constatons que la longueur d'arc est d'environ 5,23, comme trouvé précédemment. En fait, les deux réponses coïncident jusqu'aux cinq premières décimales !

3.4 Aire de surface

Une aire de surface de révolution peut être calculée en la découpant en fines bandelettes. Chaque bandelette forme un élément d'aire, $\mathrm{d}A$:



On « déroule » $\mathrm{d}A$ pour trouver son aire : la largeur de chaque bandelette est donnée par un élément d'arc, $\mathrm{d}s$, de la courbe que l'on a fait tourner, et la longueur est obtenue en appliquant la formule de la circonférence d'un cercle. Une fois que $\mathrm{d}A$ est exprimé entièrement en fonction d'une seule variable, on intègre pour sommer les éléments d'aire.

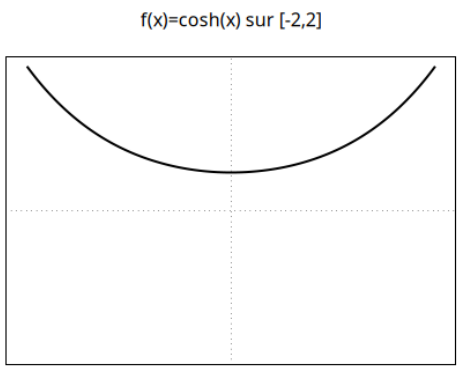
Remarque : une fois encore, les graphiques 3D s'adressent davantage aux étudiants ayant déjà suivi un cours d'analyse multivariable, mais nous les incluons ici par souci d'exhaustivité.

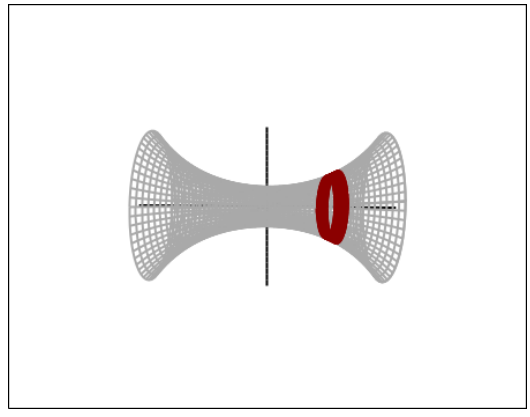
Exemple 3.4.1. Calculer l'aire de la surface créée en faisant tourner $f(x) = \cosh x$ autour de l'axe des x sur $[-2, 2]$.

Nous commençons par tracer le graphique de $f(x)$ et la surface de révolution correspondante :

```
(%i1) wxdraw2d(
      dimensions=[600,600],
      xrange=[-2.2,2.2],
      yrange=[-4,4],
      xaxis=true,
      yaxis=true,
      xtics=false,
      ytics=false,
      line_width=2,
      title="f(x)=cosh(x) sur [-2,2]",
      color=black,
      explicit(cosh(x),x,-2,2)
      );

(%i2) wxdraw3d(
    axis_3d    = false,
```

```
    dimensions = [600,600],
    view       = [85,10],
    xrange     = [-2.2,2.2],
    yrange     = [-4,4],
    zrange     = [-4,4],
    xtics      = false,
    ytics      = false,
    ztics      = false,
    color      = black,
    line_width = 2,
    parametric(t,0,0,t,-2.2,2.2),
    parametric(0,t,0,t,-4,4),
    parametric(0,0,t,t,-4,4),
    nticks     = 600,
    enhanced3d = none,
    color      = dark_grey,
    parametric_surface(r, cosh(r)*cos(t), cosh(r)*sin(t),
                       r,-2,2, t,0,2*%pi),
    color      = dark_red,
    parametric_surface(r, 1.01*cosh(r)*cos(t), 1.01*cosh(r)*sin(t),
                       r,1,1.2, t,0,2*%pi)
);
```





ds est un petit élément d'arc sur le graphe de $f(x) = \cosh x$. Là encore, on exprime ds en fonction de $f'(x)$: $\mathrm{d}s = \sqrt{(\mathrm{d}x)^2 + (\mathrm{d}y)^2} = \sqrt{1 + \left(\frac{\mathrm{d}y}{\mathrm{d}x}\right)^2} \cdot \mathrm{d}x$. Puisque $\cosh x$ est le rayon de l'élément d'aire en x, on peut écrire la longueur de l'élément d'aire « déroulé » comme $2\pi \cdot \cosh x$. Nous utilisons wxMaxima pour exprimer la formule de l'élément d'aire dA entièrement en fonction de x :

```
(%i3) f(x):=cosh(x)$
      dels:sqrt(1+(diff(f(x),x))^2)$
      delA:2*%pi*f(x)*dels;
```

(%o5) $2\,\pi\cosh\left(x\right)\,\sqrt{\sinh\left(x\right)^2+1}$

Enfin, nous additionnons tous les éléments d'aire de la surface à l'aide d'une intégrale :

$$A = \int \mathrm{d}A = \int_{-2}^{2} 2\pi \cdot \cosh x \sqrt{(\sinh x)^2 + 1}\ \mathrm{d}x$$

```
(%i6) integrate(%,x,-2,2);
(%o6) (%e^(-4)*(%e^8+8*%e^4-1)*%pi)/2
(%i7) float(%);
(%o7) 98.30017399792946
```

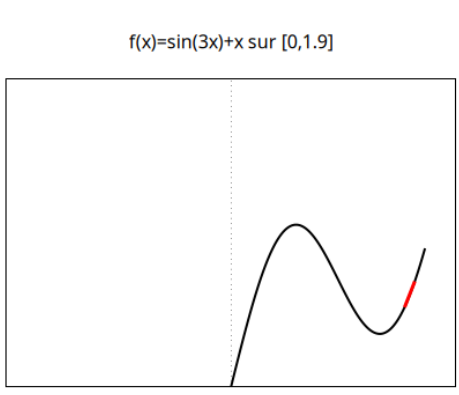
Nous obtenons une aire de surface d'environ 98.3 unités d'aire.

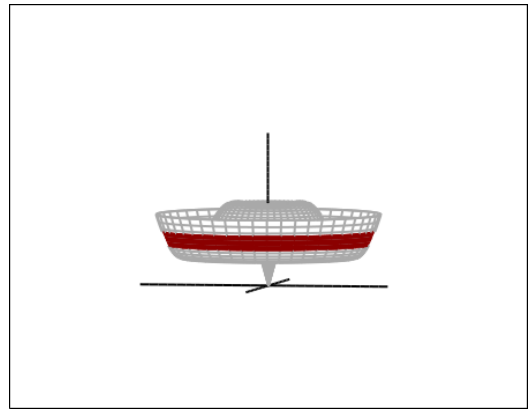
Exemple 3.4.2. Calculer l'aire de la surface obtenue en faisant tourner $f(x) = \sin 3x + x$ autour de l'axe des y sur [0; 1,9].

Comme précédemment, nous commmençons par les représentations graphiques en 2d et en 3d :

```
(%i8) f(x):=sin(3*x)+x$
      wxdraw2d(
      dimensions=[600,600],
      xrange=[-2.2,2.2],
      yrange=[0,3],
      xaxis=true,
      yaxis=true,
      xtics=false,
      ytics=false,
      line_width=2,
      title="f(x)=sin(3x)+x sur [0,1.9]",
      color=black,
      explicit(f(x),x,0,1.9),
      line_width=3,
      color=red,
      explicit((f(1.8)-f(1.7))/.1*(x-1.7)+f(1.7),x,1.7,1.8)
      );

(%i9) wxdraw3d(
      axis_3d=false,
      dimensions=[600,600],
      view=[85,10],
      xrange=[-2.2,2.2],
      yrange=[-2.2,2.2],
      zrange=[0,3],
      xtics=false,
      ytics=false,
      ztics=false,
      color=black,
      line_width=2,
```

```
 parametric(t,0,0,t,-2.2,2.2),
 parametric(0,t,0,t,-2.2,2.2),
 parametric(0,0,t,t,0,3),
nticks=600,
surface_hide=true,
enhanced3d=none,
color=dark_grey,
 parametric_surface(r*cos(t),r*sin(t),f(r),r,0,1.9,t,0,2*%pi),
color=dark_red,
  parametric_surface(r*cos(t),r*sin(t),f(r),r,1.7,1.8,t,0,2*%pi)
);
```





À nouveau, on écrit $\mathrm{d}s = \sqrt{1 + (f'(x))^2}\ \mathrm{d}x$. Cette fois, chaque anneau a un rayon égal à x, donc $\mathrm{d}A$ a une longueur de $2\pi \cdot x$. On exprime $\mathrm{d}A$ dans wxMaxima :

```
(%i10) sqrt(1+(diff(f(x),x))^2)*2*%pi*x;
```

(%o10) $2\,\pi\,x\,\sqrt{\left(3\cos\left(3\,x\right)+1\right)^2+1}$

Enfin, on additionne tous les éléments d'aire à l'aide d'une intégrale :

$$A = \int \mathrm{d}A = \int_0^{1.9} 2\,\pi\,x\,\sqrt{\left(3\cos\left(3\,x\right)+1\right)^2+1}\ \mathrm{d}x$$

Lorsque nous calculons cette intégrale avec `integrate`, wxMaxima ne parvient pas à produire un résultat, aussi cherchons nous approximation décimale à l'aide de `quad_qag` :

```
(%i11) quad_qag(%,x,0,1.9,1);
(%o11) [22.52375662706631,1.6729090642515012*10^-7,195,0]
```

Nous trouvons une aire de surface d'environ $A \approx 22.5$ unités.

3.5 Exercices du Chapitre 3

1. Calculer une approximation décimale de l'aire algébrique délimitée par $f(x) = -x^2 + 2x$ sur $[0, 4]$. Réaliser un graphique en colorant les aires correspondantes, en mettant en rouge la contribution associée à l'aire algébrique positive et en bleu celle de l'aire algébrique négative.
2. Calculer une approximation décimale de l'aire délimitée entre $f(x) = \sqrt{1 - x^2}$ et $g(x) = \cos\left(\frac{\pi}{2}\, x\right)$. Vous pouvez utiliser la symétrie pour simplifier votre calcul, mais vous devez inclure l'argument algébrique approprié pour justifier l'utilisation de la symétrie. Produire un graphique coloré illustrant l'aire calculée.
3. Utiliser un graphique pour étudier les courbes de $f(x) = (x-3)(x-2)(x-1)x(x+1)(x+2)(x+3)$ et $g(x) = e^x$. Calculer avec wxMaxima toutes les intersections de ces deux fonctions, puis trouver l'aire délimitée entre elles. Comme dans l'Exemple 3.1.3, il faut veiller à rendre chaque contribution d'aire positive en choisissant la bonne différence entre les fonctions dans chaque intégrale d'aire.
4. Calculez le volume du solide obtenu par révolution de $f(x) = \cos x$ sur $[0, \pi]$ autour de l'axe des y.

   Remarque : puisque nous travaillons sur le domaine de restriction « classique » de la fonction cosinus, on peut ignorer l'avertissement « certaines solutions peuvent être perdues ».

   (a) Utilisez la méthode des disques. Assurez-vous d'énoncer explicitement l'élément de volume $\mathrm{d}V$ avant de sommer les éléments à l'aide d'une intégrale.

   (b) Utilisez la méthode des coques cylindriques. Assurez-vous d'énoncer explicitement l'élément de volume $\mathrm{d}V$ avant de sommer les éléments à l'aide d'une intégrale.

   (c) * Utilisez `wxdraw3d` pour tracer le solide ainsi que chaque type d'élément de volume.
5. Reprenez le dernier exemple en utilisant cette fois une révolution autour de l'axe des x au lieu de l'axe des y.
6. Tracez le graphique ombré de la région délimitée par $f(x) = 1 + 0{,}5x$ et $g(x) = x$ sur $[0, 2]$.

   (a) Utilisez la méthode des rondelles pour calculer le volume du solide obtenu par révolution de cette région autour de l'axe des y.

   (b) Utilisez la méthode des rondelles pour calculer le volume du solide obtenu par révolution de cette région autour de l'axe des x.

   (c) * Utilisez `wxdraw3d` pour tracer chaque solide, en incluant un élément de volume.
7. Tracez le graphique ombré de la région délimitée par $f(x) = \sin x$ et $g(x) = \cos x$ entre leurs deux premières intersections sur $\left[0, \frac{3\pi}{2}\right]$. Utilisez la méthode des rondelles pour calculer le volume du solide obtenu par révolution de cette région autour de l'axe des y.

   Remarque : bien que le cosinus soit restreint au domaine classique $[0, \pi]$, la fonction sinus ne l'est *pas*. Vous devrez donc réfléchir attentivement à la façon d'ajuster le rayon extérieur sur $\left[\frac{\pi}{2}, \frac{3\pi}{2}\right]$.

   Vérifiez que la même réponse est obtenue par la méthode des coques cylindriques.

8. Calculer la longueur d'arc de la fonction $f(x) = \sin x$ sur $[0, 2\pi]$ en utilisant deux méthodes différentes :
   (a) Utiliser un processus limite avec $n = 4, 8, 16, 32, 64$ sous-intervalles, en incluant les graphiques de chaque approximation.
   (b) Utiliser la formule intégrale de la longueur d'arc pour calculer une approximation décimale.
   (c) Calculer l'erreur relative (en pourcentage) en comparant l'approximation pour $n = 64$ et la méthode intégrale.
9. Calculer l'aire de la surface obtenue en faisant tourner la courbe représentative de $f(x) = e^{-0.2x} \cos 3x$ sur $[0, 10]$ autour de l'axe des x. Tracer la surface avec `wxdraw3d`.
10. Calculer l'aire de la surface obtenue en faisant tourner la courbe représentative de $f(x) = e^{-0.2x} \cos 3x$ sur $[0, 10]$ autour de l'axe des y. Tracer la surface avec `wxdraw3d`.
11. Calculer l'aire d'un disque de rayon R de deux façons différentes :
   (a) Utiliser l'équation du demi-cercle supérieur et calculer directement l'aire en intégrant par rapport à x.
   (b) Utiliser des anneaux concentriques comme éléments d'aire. Chaque anneau doit avoir un rayon r et une épaisseur dr. « Dérouler » un anneau et calculer l'élément d'aire dA, puis intégrer pour sommer les éléments d'aire.
12. Calculez le volume d'un cylindre de rayon R et de hauteur h en le considérant comme un solide de révolution obtenu par rotation de la droite $x = R$ autour de l'axe y.
   (a) Utilisez la méthode des disques (*disk method*). Assurez-vous d'énoncer explicitement l'élément de volume dV avant de sommer les éléments à l'aide d'une intégrale.
   (b) Utilisez la méthode des tubes (*shell method*). Assurez-vous d'énoncer explicitement l'élément de volume dV avant de sommer les éléments à l'aide d'une intégrale.
   (c) * Utilisez `wxdraw3d` pour produire une représentation graphique du cylindre avec un rayon de 1.
13. Calculez le volume d'une couronne cylindrique de rayon intérieur a, de rayon extérieur b et de hauteur h.
   (a) Utilisez la méthode des rondelles (*washer method*). Assurez-vous d'énoncer explicitement l'élément de volume dV avant de sommer les éléments à l'aide d'une intégrale.
   (b) Utilisez la méthode des tubes (*shell method*). Assurez-vous d'énoncer explicitement l'élément de volume dV avant de sommer les éléments à l'aide d'une intégrale.
   (c) * Utilisez `wxdraw3d` pour produire une représentation graphique de la couronne cylindrique avec un rayon intérieur de 1 et un rayon extérieur de 2. Vous devrez créer trois surfaces paramétriques : le cylindre intérieur, le cylindre extérieur et la « rondelle » qui coiffe l'extrémité.
14. Calculez le volume d'une sphère de rayon R en considérant un hémisphère comme une surface de révolution d'une courbe $f(x)$ définie sur $[0, R]$.

(a) Utilisez la méthode des disques. Assurez-vous d'énoncer explicitement l'élément de volume $\mathrm{d}V$ avant de sommer les éléments à l'aide d'une intégrale.

(b) Utilisez la méthode des coquilles cylindriques. Assurez-vous d'énoncer explicitement l'élément de volume $\mathrm{d}V$ avant de sommer les éléments à l'aide d'une intégrale.

(c) * Utilisez `makelist` et `wxdraw3d` pour générer 20 disques empilés dans un graphique et 20 coquilles cylindriques imbriquées dans un autre graphique, illustrant chaque méthode d'intégration pour une sphère de rayon 1.

15. Calculez le volume d'un cône de hauteur h et de base de rayon R en considérant le cône comme un solide de révolution d'un segment de droite défini sur $[0, R]$.

(a) Utilisez la méthode des disques. Assurez-vous d'énoncer explicitement l'élément de volume $\mathrm{d}V$ avant de sommer les éléments à l'aide d'une intégrale.

(b) Utilisez la méthode des coquilles cylindriques. Assurez-vous d'énoncer explicitement l'élément de volume $\mathrm{d}V$ avant de sommer les éléments à l'aide d'une intégrale.

(c) * Utilisez `makelist` et `wxdraw3d` pour générer 20 disques empilés dans un graphique et 20 coquilles cylindriques imbriquées dans un autre graphique, illustrant chaque méthode d'intégration pour un cône de hauteur 1 et de rayon 1.

16. Calculez l'aire de la surface latérale du cône de l'exemple précédent.

(a) En utilisant une intégrale en x. Assurez-vous d'énoncer explicitement l'élément d'aire $\mathrm{d}A$ avant de sommer les éléments à l'aide d'une intégrale.

(b) En utilisant une intégrale en y. Assurez-vous d'énoncer explicitement l'élément d'aire $\mathrm{d}A$ avant de sommer les éléments à l'aide d'une intégrale.

(c) * Utilisez `wxdraw3d` pour tracer le cône, en incluant un élément d'aire $\mathrm{d}A$.

Chapitre 4

Equations Différentielles Ordinaires

Commandes essentielles de ce chapitre

`declare`	`sublis`	`makelist`	`bc2`
`diff`	`float`	`implicit`	`fullratsimp`
`solve`	`rhs`	`ode2`	`load(drawdf)`
`subst`	`lhs`	`ic1`	`wxdrawdf`
`wxdraw2d`	`integrate`	`ic2`	`for-do`

4.1 Définitions fondamentales

Une **équation différentielle ordinaire** (EDO) est une équation faisant intervenir les dérivées d'une fonction $y(x)$. L'**ordre** d'une EDO est l'ordre de la dérivée la plus élevée apparaissant dans l'équation. Une **solution** d'une équation différentielle est une *fonction*, $y(x)$, satisfaisant l'équation différentielle pour tout x sur un certain intervalle.

Une EDO est dite **linéaire** si $y(x)$ et ses dérivées n'apparaissent qu'à la puissance un au plus dans l'équation. Les EDO linéaires vérifient la propriété utile suivante : toute combinaison linéaire de solutions est également une solution.

La **solution générale** d'une EDO contient une ou plusieurs constantes arbitraires ; autrement dit, la solution générale est en réalité une famille de courbes déterminée par un ou plusieurs paramètres. Une EDO du premier ordre possède une solution générale avec une constante arbitraire, une EDO du second ordre possède une solution générale avec deux constantes arbitraires, et ainsi de suite. Lorsque les constantes de la solution générale sont fixées, on obtient une **solution particulière** de l'équation.

Si l'on connait la solution générale d'une EDO du premier ordre, on peut spécifier une valeur pour $y(0)$ ou, plus généralement, $y(a)$ afin de déterminer la solution particulière vérifiant cette condition. La valeur ainsi spécifiée est appelée une **condition initiale**. Pour une équation du second ordre, il est nécessaire de spécifier deux conditions afin de déterminer les deux constantes de la solution générale. En général, on spécifie les conditions initiales $y(0)$ et $y'(0)$, ou bien on spécifie un ensemble de **conditions aux limites** $y(a)$ et $y(b)$.

Exemple 4.1.1. Montrez que $y(x) = \frac{x^4}{2} + 5x + c$, où c est une constante arbitraire, est la solution générale de l'équation différentielle $y'(x) = 2x^3 + 5$. Trouvez la solution particulière correspondant à la condition initiale $y(0) = 2$. Tracez la famille de courbes correspondant à plusieurs valeurs de c de la solution générale, et tracez la solution particulière en rouge.

Nous utilisons la commande `declare` pour définir la constante avec Maxima, puis nous dérivons pour vérifier que $y(x)$ satisfait bien l'équation différentielle donnée :

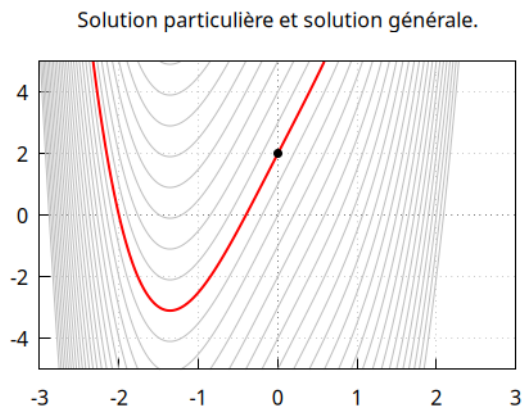
```
(%i1) declare(c,constant)$
      y(x):=x^4/2+5*x+c$
      diff(y(x),x);
(%o3) 2*x^3+5
```

Nous constatons que $y'(x) = 2x^3 + 5$, donc $y(x)$ est bien une solution de l'équation différentielle. Puisque $y(x)$ contient une constante arbitraire, il s'agit de la solution générale de l'équation différentielle d'ordre 1. Pour trouver la solution particulière, nous substituons le point $[0, 2]$ dans la solution générale et résolvons pour trouver c :

```
(%i4) solve(y(0)=2,c);
(%o4) [c=2]
(%i5) y_p:subst(%[1],y(x));
(%o5) x^4/2+5*x+2
```

Enfin, nous traçons la solution particulière $y_p(x)$ ainsi que la famille de courbes de la solution générale. Nous générons la famille de courbes en utilisant `makelist` pour substituer différentes valeurs de c dans la formule de $y(x)$. Nous incluons le point $[0, 2]$ pour mettre en évidence que $y_p(x)$ est l'*unique* solution passant par ce point :

```
(%i6) FAMILY:makelist(explicit(y(x),x,-3,3),c,-20,20)$
(%i7) wxdraw2d(
      grid=true,
      xrange=[-3,3],
      yrange=[-5,5],
      xaxis=true,
      yaxis=true,
      title="Solution particulière et solution générale.",
      color=grey,
       FAMILY,
      color=red,
      line_width=2,
       explicit(y_p,x,-3,3),
      color=black,
      point_type=7,
       points([[0,2]])
      );
```



Exemple 4.1.2. Montrez que $y(x) = 5e^{2x}$ est une solution particulière de l'équation différentielle $y'(x) = 2y$. Donner l'expression de la solution *générale*, et utilisez `diff` pour vérifier qu'elle est correcte.

```
(%i8) y(x):=5*%e^(2*x)$
      diff(y(x),x);
      2*y(x);
(%o9) 10*%e^(2*x)
(%o10) 10*%e^(2*x)
```

Nous constatons que $y'(x)$ et $2y$ donnent exactement la même expression, donc $y(x)$ satisfait l'équation $y'(x) = 2y$. Nous remarquons que le facteur 2 provient de la règle de dérivation en chaîne appliquée à l'exposant. Le coefficient 5 n'interfère pas du tout avec l'équation différentielle, nous essayons donc comme solution générale $y(x) = c \cdot e^{2x}$. Vérification avec `diff` :

```
(%i11) y(x):=c*%e^(2*x)$
       declare(c,constant)$
       diff(y(x),x);
       2*y(x);
(%o13) 2*c*%e^(2*x)
(%o14) 2*c*%e^(2*x)
```

$y(x) = c \cdot e^{2x}$ est l'expression de la solution générale de l'équation différentielle du premier ordre $y'(x) = 2y$.

Exemple 4.1.3. Montrez que $y_1(x) = \sin x$ et $y_2(x) = \cos x$ sont des solutions de l'équation différentielle ordinaire du second ordre $y''(x) = -y$. Expliquez pourquoi cette équation est linéaire, puis montrez que la combinaison linéaire de solutions $c_1 \cdot y_1(x) + c_2 \cdot y_2(x)$ est également une solution de l'équation différentielle.

```
(%i15) y_1(x):=sin(x)$
       y_2(x):=cos(x)$
       diff(y_1(x),x,2);
       diff(y_2(x),x,2);
(%o17) -sin(x)
(%o18) -cos(x)
```

Dans chaque cas, nous constatons que la dérivée seconde donne l'opposé de la fonction originale, donc y_1 et y_2 sont toutes deux des solutions.

L'équation différentielle est linéaire car y et ses dérivées ne sont exprimées qu'avec 1 comme puissance. Une combinaison linéaire de solutions doit également être une solution :

```
(%i19) declare(c_1,constant,c_2,constant)$
       y_gen(x):=c_1*y_1(x)+c_2*y_2(x)$
(%i21) diff(y_gen(x),x,2);
(%o21) -c_1*sin(x)-c_2*cos(x)
```

Après deux dérivations, nous constatons que $y''_{gen}(x) = -y_{gen}(x)$, donc la combinaison linéaire précédemment définie est également une solution de l'équation différentielle. $y_{gen}(x)$ est la solution *générale*, car elle contient deux constantes arbitraires.

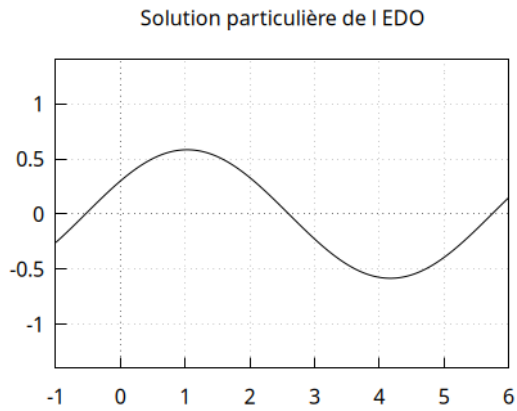
Exemple 4.1.4. Appliquez les conditions initiales $y(0) = 0.3$ et $y'(0) = 0.5$ à la solution générale de l'exemple 4.1.3. Tracez la solution particulière obtenue et commentez la relation entre les conditions initiales et le graphe.

Chaque condition initiale correspond à une équation liant c_1 et c_2. En général, les conditions initiales conduisent à un système d'équations non trivial en c_1 et c_2, mais cet exemple est particulièrement simple car $\sin x$ s'annule en $x = 0$:

```
(%i22) EQN1:y_gen(0)=0.3;
       diff(y_gen(x),x);
       EQN2:subst(0,x,%)=0.5;
(%o22) c_2=0.3
(%o23) c_1*cos(x)-c_2*sin(x)
(%o24) c_1=0.5
(%i25) y_p:sublis([c_1=0.5,c_2=0.3],y_gen(x));
(%o25) 0.5*sin(x)+0.3*cos(x)
```

Nous constatons que $y_p(x) = 0.5 \sin x + 0.3 \cos x$ est la fonction qui vérifie l'équation différentielle originale et les conditions initiales. Enfin, nous traçons cette solution particulière :

```
(%i26) wxdraw2d(
       grid=true,
       xrange=[-1,6],
       yrange=[-1.4,1.4],
       xaxis=true,
       yaxis=true,
       title="Solution particulière de l EDO",
       color=black,
        explicit(y_p,x,-1,6)
       );
```



En examinant attentivement le graphe, nous constatons que la valeur initiale de la solution est environ 0.3 et que la pente initiale est environ 0.5, comme attendu.

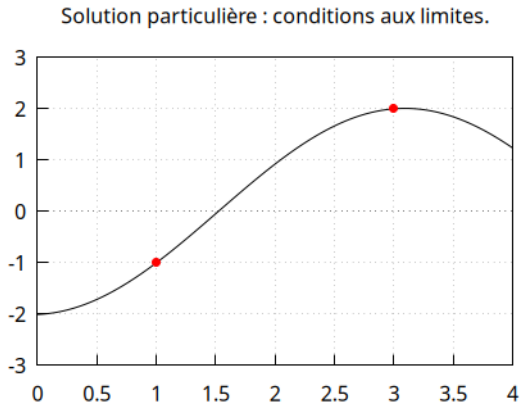
Exemple 4.1.5. Appliquez les conditions aux limites $y(1) = -1$ et $y(3) = 2$ à la solution générale de l'exemple 4.1.3. Tracez la solution particulière obtenue en incluant les conditions aux limites sous forme de points.

Les conditions aux limites conduisent cette fois à un système d'équations non trivial, nous utilisons donc `solve` pour trouver les valeurs de c_1 et c_2 :

```
(%i27) EQN1:y_gen(1)=-1;
       EQN2:y_gen(3)=2;
(%o27) cos(1)*c_2+sin(1)*c_1=-1
(%o28) cos(3)*c_2+sin(3)*c_1=2
(%i29) SOLUTION:solve([EQN1,EQN2],[c_1,c_2]);
(%o29) [[c_1=-(cos(3)+2*cos(1))/(sin(1)*cos(3)-cos(1)*sin(3)),
       c_2=(sin(3)+2*sin(1))/(sin(1)*cos(3)-cos(1)*sin(3))]]
```

Le résultat de `solve` est une liste contenant une liste, nous devons donc extraire soigneusement les solutions et les substituer dans l'expression générale de $y_{gen}(x)$. Nous utilisons également `float` pour obtenir une approximation décimale plus lisible :

```
(%i30) y_p:float(sublis([c_1=rhs((%[1])[1]),c_2=rhs((%[1])[2])],y_gen(x)));
(%o30) 0.099650689051389*sin(x)-2.006012470576742*cos(x)
(%i31) wxdraw2d(
       grid=true,
       xaxis=true,
       yaxis=true,
       xrange=[0,4],
       yrange=[-3,3],
       title="Solution particulière : conditions aux limites.",
       color=black,
        explicit(y_p,x,0,4),
       color=red,
       point_type=7,
        points([[1,-1],[3,2]])
       );
```



4.2 Equations à variables séparables

Une équation différentielle ordinaire du premier ordre est dite **à variables séparables** si l'on peut isoler algébriquement toutes les dépendances en x et en y de part et d'autre de l'équation. Une équation séparable peut s'écrire sous la forme $f(y)\,\mathrm{d}y = g(x)\,\mathrm{d}x$, ce qui permet d'intégrer simplement pour obtenir $\displaystyle\int f(y)\,\mathrm{d}y = \int g(x)\,\mathrm{d}x$. Chaque intégrale indéfinie est définie à une constante arbitraire près, mais on peut regrouper ces constantes en une seule constante arbitraire afin d'écrire la solution générale. La solution générale peut être obtenue explicitement sous la forme d'une fonction $y(x)$, mais les solutions peuvent également être définies implicitement comme des équations reliant y et x.

Exemple 4.2.1. Trouver la solution générale de $y'(x) = 2y$ en séparant les variables.

On commence par passer à la notation de Leibniz : $\dfrac{\mathrm{d}y}{\mathrm{d}x} = 2y$. On peut alors obtenir la forme $f(y)\,\mathrm{d}y = g(x)\,\mathrm{d}x$ en divisant les deux membres par y et en multipliant les deux membres par $\mathrm{d}x$: $\dfrac{\mathrm{d}y}{y} = 2\,\mathrm{d}x$. Enfin, on utilise wxMaxima pour réaliser le calcul des intégrales :

```
(%i1) integrate(1/y,y);
      integrate(2,x);
(%o1) log(y)
(%o2) 2*x
```

Pour exprimer la solution obtenue, il faut noter que la forme la plus correcte de l'intégrale en y est en réalité $\ln|y|$, et que nous devons introduire une constante arbitraire, c :

```
(%i3) EQN:log(abs(y))=2*x+c;
      declare(c,constant)$
(%o4) log(abs(y))=2*x+c
(%i5) solve(EQN,y);
(%o5) [abs(y)=%e^(c+2*x)]
```

Le membre de droite de cette équation peut être réécrit sous la forme $e^c e^{2x}$, mais comme c est une constante arbitraire, on peut simplement écrire $c \cdot e^{2x}$, où c est une nouvelle constante arbitraire positive. De plus, $|y| = y$ si $y > 0$ et $|y| = -y$ si $y < 0$, donc on peut simplement écrire $y = c \cdot e^{2x}$, où c peut être positif ou négatif. On vérifie notre solution générale à l'aide de `diff` :

```
(%i6) y_gen:c*%e^(2*x)$
      diff(y_gen,x);
(%o7) 2*c*%e^(2*x)
```

Nous voyons que $y'(x) = 2y$, donc nous avons bien une solution générale valide.

Exemple 4.2.2. Utilisez la méthode de séparation des variables pour résoudre l'EDO $y'(x) = \dfrac{\cos^2 x}{y^3 - 2}$. Trouvez la solution particulière passant par le point $[1, 1]$. Tracez la

famille de courbes de la solution générale ainsi que la solution particulière, représentée en rouge.

On commence par transformer l'équation sous la forme $(y^3 - 2)\,\mathrm{d}y = \cos^2 x\,\mathrm{d}x$, puis on utilise wxMaxima pour trouver les intégrales.

```
(%i8) lhs:integrate(y^3-2,y);
      rhs:integrate((cos(x))^2,x);
(%o8) y^4/4-2*y
(%o9) (sin(2*x)/2+x)/2
```

Nous ajoutons maintenant une constante arbitraire afin d'exprimer correctement la solution générale :

```
(%i10) gensolution:lhs=rhs+c;
       declare(c,constant)$
(%o10) y^4/4-2*y=(sin(2*x)/2+x)/2+c
```

La solution générale est définie implicitement, mais nous pouvons tout de même utiliser la condition initiale $y = 1$ lorsque $x = 1$ afin de déterminer c pour la solution particulière :

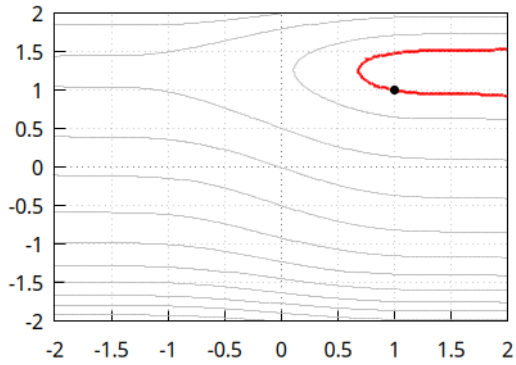
```
(%i11) sublis([x=1,y=1],gensolution);
(%o11) -7/4=(sin(2)/2+1)/2+c
(%i12) solve(%,c);
(%o12) [c=-(sin(2)+9)/4]
(%i13) partsolution:sublis([%[1]],gensolution);
(%o13) y^4/4-2*y=(sin(2*x)/2+x)/2-(sin(2)+9)/4
```

Enfin, nous définissons la solution générale comme une fonction de c et traçons la famille de courbes de la solution générale à l'aide de `makelist`. Nous traçons la solution particulière en rouge.

```
(%i14) gensolution;
(%o14) y^4/4-2*y=(sin(2*x)/2+x)/2+c
(%i15) gsfunc(c):=''%;
(%o15) gsfunc(c):=y^4/4-2*y=(sin(2*x)/2+x)/2+c
(%i16) FAMILY:makelist(implicit(gsfunc(c),x,-2,2,y,-2,2),c,-10,10)$
(%i17) wxdraw2d(
       grid=true,
       dimensions=[600,600],
       xaxis=true,
       yaxis=true,
       title="Courbes de la solution générale et solution particulière",
       xrange=[-2,2],
       yrange=[-2,2],
       color=red,
       line_width=2,
        implicit(partsolution,x,-2,2,y,-2,2),
       color=black,
       point_type=7,
        points([[1,1]]),
```

```
line_width=1,
color=grey,
 FAMILY
);
```

Courbes de la solution générale et solution particulière



4.3 Solveur intégré de wxMaxima pour les EDO

wxMaxima peut trouver les solutions générales pour certaines équations différentielles ordinaires du premier et du second ordre à l'aide de la commande intégrée `ode2`. Nous pouvons choisir de calculer des solutions particulières manuellement (en résolvant un système d'équations), ou à l'aide des commandes intégrées `ic1` (une condition initiale), `ic2` (deux conditions initiales) ou `bc2` (deux valeurs limites). Notez que beaucoup d'équations différentielles n'ont pas de solution s'exprimant de manière analytique.

Exemple 4.3.1. Utilisez `ode2` pour trouver la solution générale de $y'(x) = 2y$, puis utilisez `ic1` pour trouver la solution particulière correspondant à la condition initiale $[1, 3]$. Vérifiez que la solution particulière satisfait l'EDO et la condition initiale. Enfin, tracez la solution particulière ainsi que le point $[1, 3]$.

Nous commençons par résoudre l'équation différentielle puis nous appliquons la condition initiale :

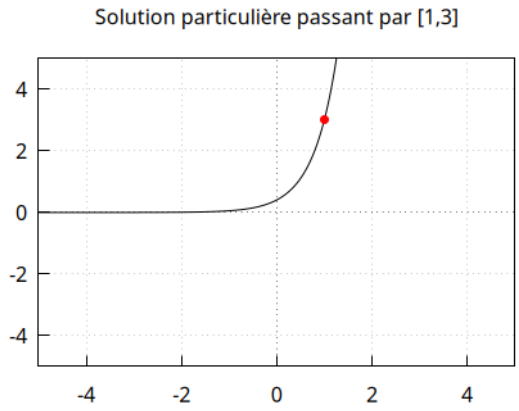
```
(%i1) eqn:'diff(y,x)=2*y;
(%o1) 'diff(y,x,1)=2*y
(%i2) sol:ode2(eqn,y,x);
(%o2) y=%c*%e^(2*x)
(%i3) ic1(sol,x=1,y=3);
(%o3) y=3*%e^(2*x-2)
```

Vérifions maintenant que cette solution est correcte :

```
(%i4) partsolution:rhs(%)$
(%i5) diff(partsolution,x);
(%o5) 6*%e^(2*x-2)
(%i6) subst(1,x,partsolution);
(%o6) 3
```

Nous voyons que la dérivée première est égale au double de la fonction originale, donc la solution particulière satisfait bien $y' = 2y$. La solution vérifie également la condition initiale car nous obtenons $y = 3$ lorsque nous substituons $x = 1$. Enfin, nous traçons le graphique de la solution particulière :

```
(%o7) wxdraw2d(
       grid=true,
       xaxis=true,
       yaxis=true,
       xrange=[-5,5],
       yrange=[-5,5],
       title="Solution particulière passant par [1,3]",
       color=black,
        explicit(partsolution,x,-5,5),
       color=red,
       point_type=7,
        points([[1,3]])
       );
```



Exemple 4.3.2. Utilisez `ode2` pour trouver la solution générale de $\frac{\mathrm{d}y}{\mathrm{d}x} + xy = x^3$. Trouver la solution particulière vérifiant la condition initiale $[1, 1]$ en résolvant manuellement l'équation correspondante pour trouver la valeur de la constante, puis vérifiez que `ic1` donne la même réponse.

```
(%i8) eqn:'diff(y,x)+x*y=x^3;
(%o8) 'diff(y,x,1)+x*y=x^3
(%i9) sol:ode2(eqn,y,x);
(%o9) y=%e^(-x^2/2)*(((2*x^2-4)*%e^(x^2/2))/2+%c)
(%i10) sublis([x=1,y=1],%);
(%o10) 1=(%c-sqrt(%e))/sqrt(%e)
(%i11) solve(%,%c);
(%o11) [%c=2*sqrt(%e)]

(%i12) sublis(%,sol);
(%o12) y=%e^(-x^2/2)*(((2*x^2-4)*%e^(x^2/2))/2+2*sqrt(%e))

(%i13) ic1(sol,x=1,y=1);
(%o13) y=%e^(-x^2/2)*((x^2-2)*%e^(x^2/2)+2*sqrt(%e))
```

La seule différence entre les deux solutions particulières trouvées est un facteur de 2 qui a été simplifié.

Exemple 4.3.3. Trouvez la solution générale de $\frac{\mathrm{d}^2y}{\mathrm{d}x^2} - 0.3 \cdot \frac{\mathrm{d}y}{\mathrm{d}x} - 2y = 0$ à l'aide de `ode2`, puis appliquez les conditions initiales $y(0) = 1.2$ et $y'(0) = 0.4$ en utilisant `ic2`. Vérifiez que votre solution particulière résout l'EDO originale.

```
(%i14) ratprint:false$
(%i15) eqn:'diff(y,x,2)-0.3*'diff(y,x)-2*y=0;
(%o15) 'diff(y,x,2)-0.3*('diff(y,x,1))-2*y=0
(%i16) gensolution:ode2(eqn,y,x);
```

```
(%o16) y=%k1*%e^(((sqrt(809)/10+3/10)*x)/2)
        +%k2*%e^(((3/10-sqrt(809)/10)*x)/2)
(%i17) partsolution:rhs(ic2(gensolution,x=0,y=1.2,'diff(y,x)=0.4));
(%o17) ((11*sqrt(809)+2427)*%e^(((sqrt(809)/10+3/10)*x)/2))/4045
      -((11*sqrt(809)-2427)*%e^(((3/10-sqrt(809)/10)*x)/2))/4045
```

Nous vérifions maintenant notre solution particulière en la substituant à y dans l'équation différentielle originale :

```
(%i18) fullratsimp(diff(partsolution,x,2)-0.3*diff(partsolution,x)
      -2*partsolution);
(%o18) 0
```

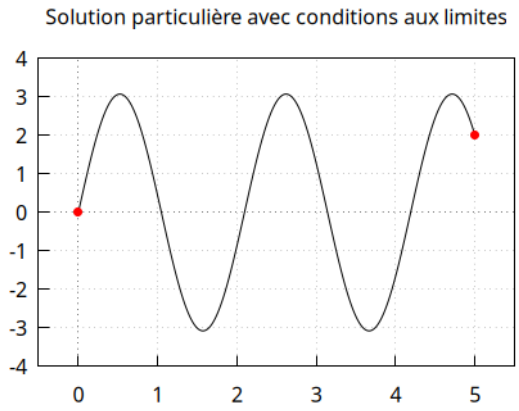
Exemple 4.3.4. Trouvez la solution de $\dfrac{\mathrm{d}^2y}{\mathrm{d}x^2} = -9 \cdot y(x)$ vérifiant les conditions aux limites $y(0) = 0$ et $y(5) = 2$. Réalisez un graphique pour vérifier que les conditions aux limites sont satisfaites.

Nous trouvons la solution générale, puis utilisons `bc2` pour appliquer les conditions aux limites :

```
(%i19) eqn:'diff(y,x,2)=-9*y;
(%o19) 'diff(y,x,2)=-9*y
(%i20) gensoln:ode2(eqn,y,x);
(%o20) y=%k1*sin(3*x)+%k2*cos(3*x)
(%i21) bc2(gensoln,x=0,y=0,x=5,y=2);
(%o21) y=(2*sin(3*x))/sin(15)
```

Nous construisons maintenant le graphique :

```
(%i22) partsolution:rhs(%);
(%o22) (2*sin(3*x))/sin(15)
(%i23) wxdraw2d(
       grid=true,
       xaxis=true,
       yaxis=true,
       xrange=[-.5,5.5],
       yrange=[-4,4],
       title="Solution particulière avec conditions aux limites",
       color=black,
        explicit(partsolution,x,0,5),
       color=red,
       point_type=7,
        points([[0,0],[5,2]])
       );
```


Solution particulière avec conditions aux limites
4
3
2
1
0
-1
-2
-3
-4
0
1
2
3
4
5

4.4 Champ de directions

Les champs de directions sont un outil graphique permettant de comprendre les solutions des équations différentielles du premier ordre de la forme $\frac{\mathrm{d}y}{\mathrm{d}x} = f(x, y)$. Pour une équation de cette forme, on connaît immédiatement la *pente* de $y(x)$ en tout point (x, y) en substituant simplement x et y dans le membre de droite. Un champ de directions (aussi appelé champ de pentes) est construit en calculant la pente en de nombreux points du plan. Une fois qu'une condition initiale est spécifiée, la solution particulière suit simplement le « flux » du champ de directions. wxMaxima dispose d'une commande intégrée `wxdrawdf` permettant de créer un champ de directions (avec ou sans solution particulière).

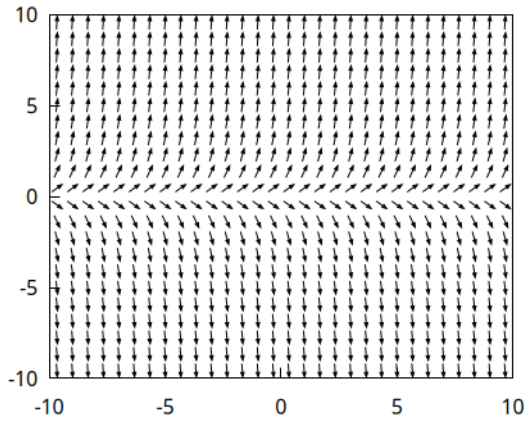
Exemple 4.4.1. Tracez un champ de directions pour l'EDO $y'(x) = 2y$, puis réalisez un graphique montrant la solution particulière passant par le point $[1, 3]$.

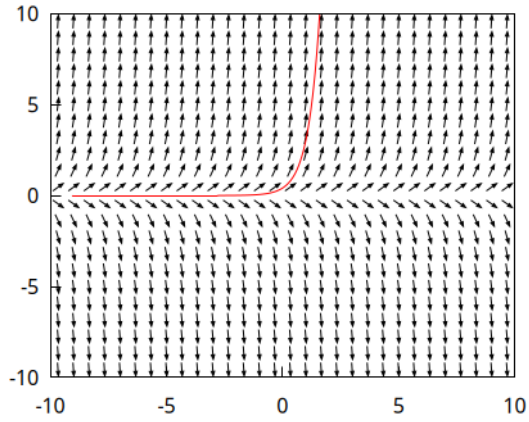
Tout d'abord, nous traçons le champ de directions sans solution particulière :

```
(%i1) load(drawdf)$
(%i2) wxdrawdf(2*y,[x,-10,10],[y,-10,10]);
```

Toute solution particulière suit le « flux » de ce champ (la pente de toute solution correspond à la pente des segments de droite en tout point). Nous ajoutons maintenant la solution particulière au champ de directions :

```
(%i3) wxdrawdf(2*y,[x,-10,10],[y,-10,10],[trajectory_at,1,3]);
```

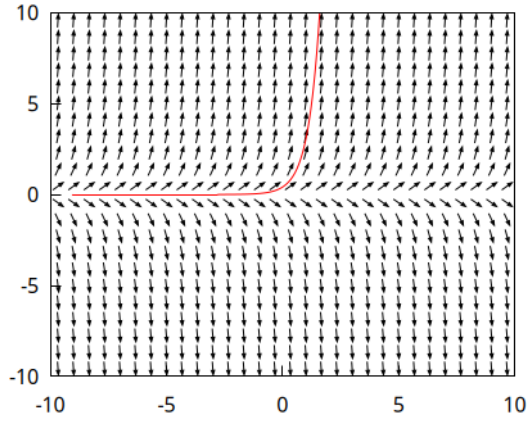




Exemple 4.4.2. Tracez un champ de directions pour l'EDO $y'(x) = y(1-y)$. Précisez les intervalles de conditions initiales (en $x = 0$) qui produisent des types de solutions particulières nettement différents. De plus, il existe deux conditions initiales qui produisent des solutions particulières constantes (solutions d'*équilibre*). Tracez un exemple.

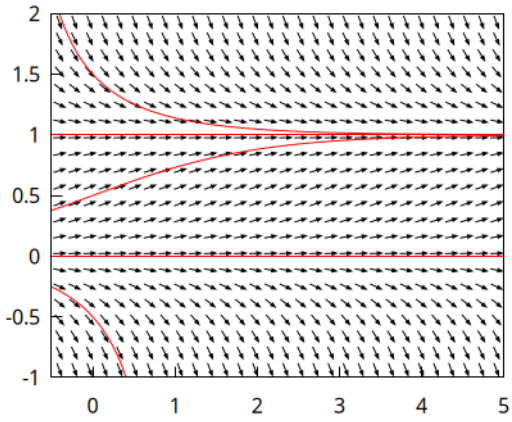
Nous traçons le champ de directions avec la commande `wxdrawdf` :

```
(%i4) wxdrawdf(y*(1-y),[x,-.5,5],[y,-1,2]);
```



Nous observons que les solutions particulières se comportent différemment sur les intervalles en y suivants : $(-\infty, 0)$ (ces solutions divergent vers $-\infty$), $(0, 1)$ (ces solutions s'approchent de $y = 1$ par valeurs inférieures) et $(1, \infty)$ (ces solutions s'approchent de $y = 1$ par valeurs supérieures). Les conditions initiales en $(0, 0)$ et $(1, 1)$ produisent des solutions particulières *constantes*. Nous traçons les solutions particulières correspondant aux conditions initiales $y(0) = -0{,}5$, $y(0) = 0$, $y(0) = 0{,}5$, $y(0) = 1$ et $y(0) = 1{,}5$:

```
(%i5) wxdrawdf(y*(1-y),[x,-.5,5],[y,-1,2],[trajectory_at,0,-0.5],
      [trajectory_at,0,0],[trajectory_at,0,0.5],[trajectory_at,0,1],
      [trajectory_at,0,1.5]);
```


2
1.5
1
0.5
0
-0.5
-1
0
1
2
3
4
5

4.5 Euler's Method

La méthode d'Euler est une méthode numérique permettant d'approcher une solution particulière d'une EDO du premier ordre de la forme $\frac{\mathrm{d}y}{\mathrm{d}x} = f(x, y)$.
Pour approcher une solution avec la condition initiale (x_0, y_0), on utilise l'EDO pour trouver la pente y_0' en (x_0, y_0), puis on effectue un petit pas Δx vers la droite, en utilisant cette pente pour approximer la valeur suivante de y :

$$x_1 = x_0 + \Delta x, \qquad y_1 = y_0 + y_0' \cdot \Delta x.$$

Ce processus est répété jusqu'à obtenir l'approximation souhaitée de la solution particulière — soit $y(c)$ en un point particulier, soit $y(x)$ sur un intervalle entier. L'approximation devient plus précise à mesure que le pas Δx diminue.

La méthode d'Euler utilise le résultat de chaque étape comme entrée de l'étape suivante, ce qui rend l'utilisation d'une structure de boucle particulièrement appropriée. On utilise une boucle `for-do` pour générer une liste de coordonnées (en commençant par la condition initiale) jusqu'à obtenir l'approximation souhaitée.

Exemple 4.5.1. Utiliser la méthode d'Euler pour approcher $y(2)$ pour l'EDO $y'(x) = 2y$ avec la condition initiale $(1, 3)$. Effectuer les approximations en divisant l'intervalle $(1, 2)$ en $n = 2, 5, 10$ sous-intervalles. Comparer avec la solution obtenue par la commande `ode2` de wxMaxima. Enfin, choisir n suffisamment grand pour obtenir une approximation par la méthode d'Euler précise à deux décimales près.

On définit l'équation différentielle, le point de départ $(x_0, y_0) = (X, Y)$, et $\Delta x = \frac{2-1}{n}$, puis on met en place la boucle pour calculer les valeurs suivantes de x et y à chaque étape :

$$x_{i+1} = x_i + \Delta x \qquad \text{et} \qquad y_{i+1} = y_i + y_i' \cdot \Delta x.$$

On exécute la boucle pour $n = 2, 5, 10$ (seul le cas $n = 2$ est présenté ci-dessous). Les résultats des trois approximations sont affichés côte à côte ci-dessous :

```
(%i1) DIFF(y):=2*y$
      delx(n):=(2-1)/n$
(%i3) X:1$
      Y:3$
      n:2$
(%i6) (print(" x ....... y"),
       print(float(X),"......",float(Y)),
       for k:1 thru n do
       (SLOPE:DIFF(Y),
        NEXTY:Y+SLOPE*delx(n),
        NEXTX:X+delx(n),
        print(float(NEXTX),"......",float(NEXTY)),
        Y:NEXTY,
        X:NEXTX)
      );
```

```
 x .......y       x .......y            x .......y
1.0......3.0     1.0......3.0          1.0......3.0
1.5......6.0     1.2......4.2          1.1......3.6
2.0......12.0    1.4......5.88         1.2......4.32
                 1.6......8.231999     1.3......5.184
                 1.8......11.5248      1.4......6.2208
                 2.0......16.13472     1.5......7.46496
                                       1.6......8.957952
                                       1.7......10.749542
                                       1.8......12.899450
                                       1.9......15.479341
                                       2.0......18.575209
```

Nous obtenons les approximations de $y(2) \approx 12.0$ ($n = 2$), $y(2) \approx 16.1$ ($n = 5$) et $y(2) \approx 18.6$ ($n = 10$).

Pour terminer, nous calculons l'approximation avec la commande `ode2` :

```
(%i7) EQN:'diff(y,x)=2*y$
      SOLN:ode2(EQN,y,x);
(%o8) y=%c*%e^(2*x)
(%i9) PARTSOLN:ic1(SOLN,x=1,y=3);
(%o9) y=3*%e^(2*x-2)
(%i10) float(subst(2,x,PARTSOLN));
(%o10) y=22.16716829679195
```

Avec $y(2) \approx 22{,}2$, on constate que les approximations d'Euler pour $n = 2$, $n = 5$ et $n = 10$ ne sont pas très précises (bien qu'elles se rapprochent de la bonne valeur au fur et à mesure que n augmente). On peut améliorer l'approximation en augmentant simplement n. On modifie la boucle `do` avec une instruction `if-else` afin de n'afficher que la dernière étape pour obtenir une approximation de $y(2)$, et on utilise $n = 9999$:

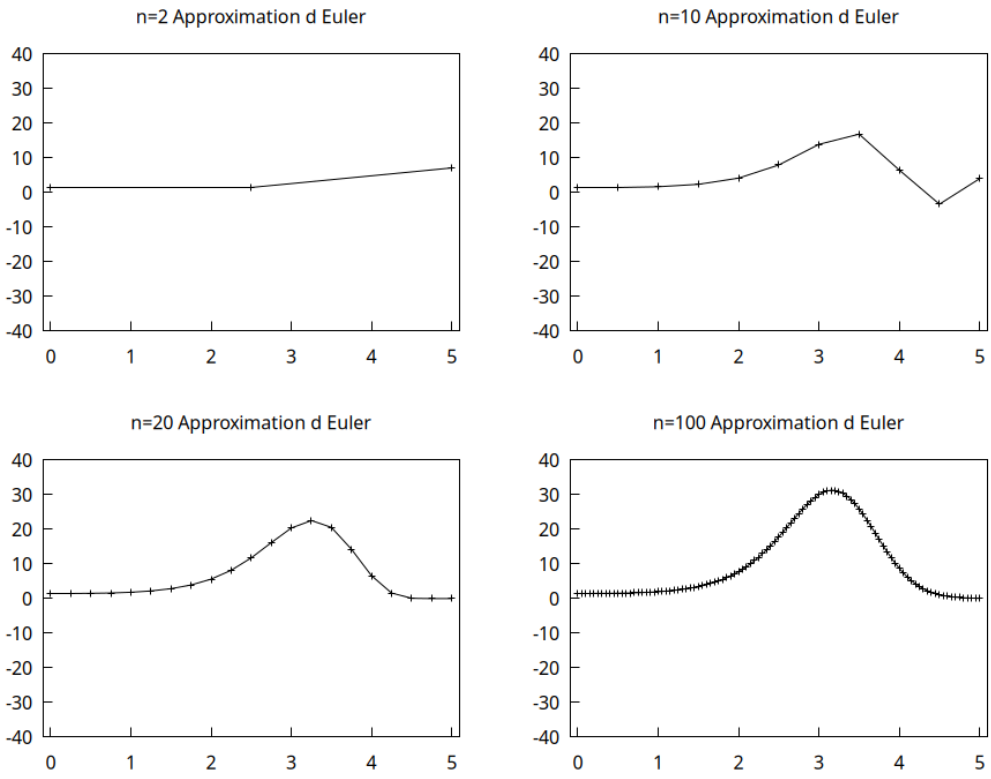
```
(%i11) kill(all)$
(%i1) DIFF(y):=2*y$
      delx(n):=(2-1)/n$
(%i3) X:1$
      Y:3$
      n:9999$
(%i6) (for k:1 thru n do
       (SLOPE:DIFF(Y),
        NEXTY:Y+SLOPE*delx(n),
        NEXTX:X+delx(n),
        if(k=n) then print("y(2)=",float(NEXTY)),
        Y:NEXTY,
        X:NEXTX)
      );

y(2)=22.1627354541832
(%o6) done
```

L'approximation avec $n = 9999$ coïncide à deux décimales près avec celle obtenue par la résolution de l'EDO, mais son calcul prend un temps très long sur un ordinateur ordinaire. La méthode d'Euler est un schéma d'approximation relativement peu performant, mais elle reste néanmoins instructive.

Exemple 4.5.2. Utiliser la méthode d'Euler pour tracer une solution approchée de $y'(x) = xy\sin(y)$ sur $[0, 5]$ avec la condition initiale $(0, 1{,}5)$. Appliquer l'approximation pour $n = 2$, $n = 10$, $n = 20$ et $n = 100$.

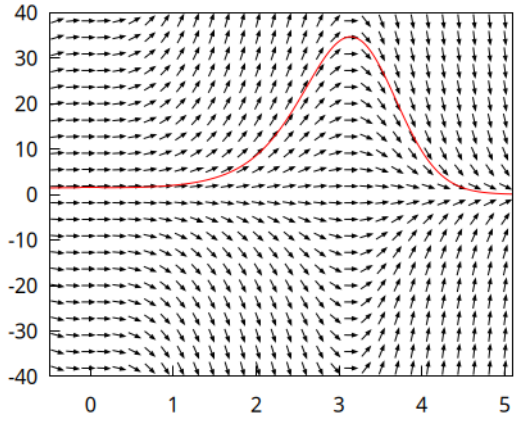
Pour tracer les résultats issus d'une boucle `do`, on utilise `append` pour ajouter un nouveau point à une liste de points à chaque itération. Le code est présenté ci-dessous uniquement pour $n = 2$, mais les quatre approximations sont tracées. On remarque que l'on utilise `float` pour obtenir des approximations décimales des points, afin de réduire la charge de calcul sur la machine (sans cette amélioration du code, wxMaxima peut se figer selon la puissance du processeur dans le cas $n = 100$) :

```
(%i7) kill(all)$
      DIFF(x,y):=x*y*sin(x)$
      delx(n):=(5-0)/n$
      X:0$
      Y:1.5$
      n:2$
      POINTS:[[X,Y]]$

(%i8) (for k:1 thru n do
      (SLOPE:DIFF(X,Y),
       NEXTY:float(Y+SLOPE*delx(n)),
       NEXTX:float(X+delx(n)),
       POINTS: append(POINTS,[[NEXTX,NEXTY]]),
       Y:NEXTY,
       X:NEXTX)
       );
(%o8) done

(%i10) wxdraw2d(
       xrange=[-0.1,5.1],
       yrange=[-40,40],
       title="n=2 Approximation d Euler",
       points_joined=true,
       color=black,
       points(POINTS)
       );
```



Pour vérifier notre approximation, on utilise `wxdrawdf` afin de visualiser la solution particulière au sein d'un champ de directions :

```
(%i11) load(drawdf)$
(%i12) wxdrawdf(x*y*sin(x),[x,-.5,5.1],[y,-40,40],[trajectory_at,0,1.5]);
```



4.6 Exercices du Chapitre 4

1. Montrer que $y(x) = c \cdot e^{x^2/2} + 3$ est la solution générale de l'EDO $\dfrac{\mathrm{d}y}{\mathrm{d}x} = xy - 3x$. Utiliser `makelist` pour générer 20 membres de la famille de solutions vérifiant l'EDO. Tracer la famille de solutions en gris ainsi que la solution particulière vérifiant $y(0) = 5$ en rouge.
2. Montrez que $y_1(x) = \sin(\omega x)$ et $y_2(x) = \cos(\omega x)$ sont toutes deux solutions de l'équation différentielle ordinaire (EDO) du second ordre $y''(x) = -\omega^2 y(x)$ (utilisez `%omega` dans wxMaxima). Quelle est la solution générale de cette EDO ? Montrez que $y(x) = A \cdot \cos(\omega x + \phi)$ est une solution générale alternative (avec les constantes arbitraires A et ϕ).
3. Trouvez la solution particulière de $y''(x) = -5\,y(x)$ soumise aux conditions aux limites $y(0) = 2$ et $y(3{,}1) = -0{,}2$. Plutôt que d'utiliser `bc2`, utilisez `subst` et `solve` pour trouver « manuellement » la solution particulière sous deux formes différentes, en utilisant les différentes expressions de la solution générale discutées dans l'exercice précédent. Utilisez `trigexpand` pour montrer que les deux solutions particulières sont équivalentes. Enfin, tracez la solution particulière en incluant les points donnés par les conditions aux limites.
4. Montrez que $c_1 e^{kx} + c_2 e^{-kx}$ est la solution générale de $y''(x) = k^2 \cdot y(x)$. Cette solution générale ressemble fortement à une paire de fonctions « spéciales » — quelles sont-elles ? Montrez que chacune de ces fonctions spéciales vérifie également l'EDO, puis exprimez la solution générale à l'aide de ces fonctions.
5. Utilisez la méthode de séparation des variables pour trouver la solution générale de $y'(x) = -y/x$. Tracez la famille de courbes de la solution générale en utilisant `makelist`.
6. Utilisez la méthode de séparation des variables pour trouver la solution générale de $y'(x) = +y/x$. Tracez la famille de courbes de la solution générale en utilisant `makelist`.
7. Utilisez la méthode de séparation des variables pour trouver la solution particulière de
$$\frac{\mathrm{d}y}{\mathrm{d}x} = \frac{x^2\sqrt{x^2+3}}{(1-y^2)^{3/2}}$$
vérifiant la condition initiale $y(3) = 0{,}5$. Tracez cette solution particulière en vérifiant qu'elle passe par le point (3, 0,5).
8. Utilisez `ode2` pour trouver la solution générale de $y'(x) = -\dfrac{1}{2} y \cdot \tan x$. Tracez la famille de solutions pour 20 valeurs de `%c` dans la solution générale.
9. Utilisez `ode2` pour trouver la solution générale de $y'(x) = -\sin(y) \cdot \cos(x)$. Tracez la famille de solutions pour 20 valeurs de `%c` dans la solution générale.
10. Tracer un champ de directions pour l'équation différentielle $y'(x) = \dfrac{\sin^2 x}{y}$, et y inclure un tracé de la solution particulière passant par le point $(0, 1)$.
11. Utiliser la méthode d'Euler avec $n = 100$ pour tracer la solution particulière de l'exercice précédent sur $[0, 5]$.
12. Tracer le champ de directions pour $y'(x) = -\dfrac{x}{y}$, en incluant les solutions particulières passant par $(0, 1)$, $(0, 2)$ et $(0, 3)$.

13. Pour l'équation différentielle de l'exercice précédent, utiliser la méthode d'Euler avec $n = 100$ pour tracer la solution particulière passant par $(0, 3)$.

14. Un problème classique lié aux équations différentielles consiste à trouver un ensemble de *trajectoires orthogonales* à une famille donnée de courbes. Les trajectoires orthogonales sont des courbes qui coupent la famille donnée à angle droit en chaque point. Considérons la famille de courbes $y = -\frac{C}{x}$. Pour trouver ses trajectoires orthogonales, nous devons d'abord déterminer la pente en tout point, pour toute valeur de C ; c'est-à-dire calculer $\frac{\mathrm{d}y}{\mathrm{d}x}$ en fonction de C. Dans ce cas, la dérivée est triviale : $\frac{\mathrm{d}y}{\mathrm{d}x} = \frac{C}{x^2}$.

    Pour que les courbes se coupent à angle droit, l'ensemble des trajecctoires orthogonales doit avoir en chaque point la pente opposée inverse. Autrement dit, les trajectoires orthogonales vérifient l'équation différentielle $\frac{\mathrm{d}y}{\mathrm{d}x} = -\frac{x^2}{C}$ (remarquons que $C = 0$ doit être exclu à ce stade). Pour résoudre cette équation différentielle, remplacez C par son expression en fonction de x et y, puis séparez les variables et intégrez (alternativement, vous pouvez utiliser `ode2`). Vous obtiendrez une nouvelle famille de courbes (avec une nouvelle constante arbitraire D) qui doit couper la famille originale à angle droit. Enfin, utilisez `makelist` pour générer des tracés des deux familles de courbes pour $C = -10, -9, \ldots, 10$ (en excluant $C = 0$) et $D = -10, -9, \ldots, 10$.

15. *La croissance logistique* se produit lorsqu'une population croît presque de manière exponentielle lorsqu'elle est petite, puis se stabilise vers une « capacité de charge » lorsqu'elle devient suffisamment grande. Le modèle de croissance logistique est une équation différentielle

$$\frac{\mathrm{d}N}{\mathrm{d}t} = kN \cdot (L - N),$$

    où N est la taille de la population et L la capacité de charge. On voit de manière évidente que la croissance est presque exponentielle lorsque N est petit :

$$\frac{\mathrm{d}N}{\mathrm{d}t} \approx (kL)N,$$

    et que le taux de croissance tend vers zéro lorsque $N \to L$. De plus, on observe que le taux de croissance devient négatif si $N > L$; autrement dit, la population diminue si elle dépasse la capacité de charge du milieu.

    Supposons qu'une population de lapins croît selon un modèle logistique avec $k = 0.312$ et $L = 1000$. On suppose que les unités de temps sont des semaines.

    (a) Tracer la population en fonction du temps en utilisant les conditions initiales $N(0) = 2$ et $N(0) = 5$.

    (b) Combien de temps faut-il à chaque population pour atteindre 80% de la capacité de charge ?

    (c) Supposons que les lapins aient un problème de surpopulation : $N(0) = 2000$. Tracer la population en fonction du temps. Combien de temps faut-il avant que la population diminue à 1200 lapins ?

    (d) Déterminer le comportement de la population si $N(0) = 1000$.

16. *La désintégration radioactive* est régie par une physique probabiliste – chaque atome d'un isotope instable porte exactement la même probabilité de désintégration par seconde. Par exemple, si un isotope a une probabilité de désintégration de $k = .001$ par atome et par seconde, et que nous avons une collection de 2000 atomes, nous attendons 2 désintégrations dans la seconde suivante. Ce comportement peut être modélisé par une équation différentielle du premier ordre : $\frac{\mathrm{d}N}{\mathrm{d}t} = -kN$; autrement dit, le taux de désintégration (atomes par seconde) est proportionnel au nombre d'atomes. La solution de cette équation différentielle est facile à deviner – écrivez-la, en veillant à bien inclure une constante arbitraire. Substituez maintenant $t = 0$, et interprètez la constante arbitraire.

    Les atomes de carbone-14 ont une probabilité de désintégration par atome et par *année* d'environ $k \approx .000121$.

    (a) En partant de $N(0) = 10^{23}$, produisez un tracé de $N(t)$ sur une période de 20 000 ans.
    (b) Calculez le pourcentage d'atomes initiaux restants après 10 000 ans.
    (c) Calculez le temps nécessaire pour que la moitié des atomes se désintègre (c'est ce qu'on appelle la demi-vie du carbone-14).

    Remarque : le C-14 est utilisé pour la *datation radiométrique* des vestiges archéologiques. La technique réelle nécessite de connaître le *pourcentage* de carbone présent sous forme de C-14 au moment de la mort d'un organisme. Ce pourcentage est relativement stable au cours du temps, car le C-14 est produit par un processus prévisible dans la haute atmosphère terrestre. Lorsqu'un organisme est vivant, il incorpore du carbone provenant de son environnement, y compris cette même fraction de C-14. Lorsqu'un organisme meurt, la fraction de C-14 diminue à mesure que le C-14 se désintègre en N-14, tandis que le carbone « ordinaire » C-12 reste stable. La *fraction* de C-14 suit la même courbe de désintégration que la *quantité* de C-14, en très bonne approximation, car le C-14 n'apparaît qu'à l'état de traces (environ une partie par billion).

17. Si l'accélération d'un objet est constante, nous pouvons décrire son comportement par une équation différentielle du second ordre, $x''(t) = a$, où a est l'accélération constante. Résoudre cette équation différentielle avec les conditions initiales $x(0) = x_0$ et $v(0) = v_0$ pour obtenir les équations standard du mouvement à accélération constante $x(t)$ et $v(t)$.

18. Trouvez les équations du mouvement pour un objet dont l'accélération est $a(t) = te^{-t}$. Utilisez les conditions initiales standard de l'exercice précédent.

Chapitre 5

Courbes paramétriques et polaires

Commandes essentielles de ce Chapitre

```
makelist        trigsimp          integrate
parametric      diff              quad_qag
eliminate       float             polar
solve           find_root         subst
dimensions      ratprint:false
```

5.1 Equations paramétriques

Nous définissons une courbe de manière *paramétrique* en explicitant les coordonnées x et y individuellement en fonction d'un paramètre, généralement t. Les **équations paramétriques** de la courbe sont

$$x(t) = f(t) \text{ et } y(t) = g(t)$$

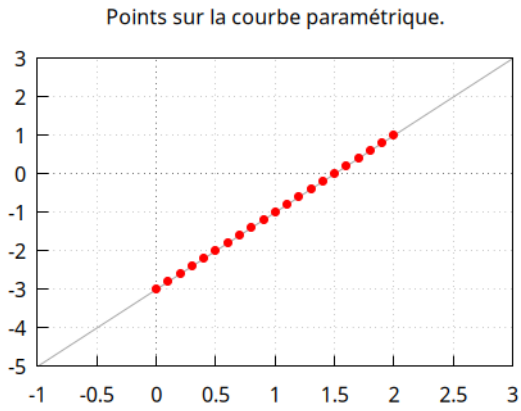
Lorsque nous choisissons une valeur particulière de t, un couple ordonné (x, y) est déterminé grace à ces équations. Si nous évaluons les équations sur un intervalle comprenant les valeurs de t sur $[a, b]$, le résultat est une courbe dans le plan commençant en $(x(a), y(a))$ et se terminant en $(x(b), y(b))$. La courbe possède une *direction* correspondant à l'ordre dans lequel les points sont tracés lorsque t augmente.

Il est souvent possible d'éliminer t en combinant algébriquement les équations paramétriques en une seule équation reliant uniquement x et y. Ce processus peut parfois nous apporter une meilleure compréhension de la courbe, mais le sens de parcours de la courbe est perdue au cours de l'opération.

Exemple 5.1.1. Évaluer les équations paramétriques $x(t) = t$ et $y(t) = 2t - 3$ pour $t = 0, 0.1, 0.2, \ldots, 2$. Réaliser un graphique montrant les points discrets ainsi que la courbe paramétrique continue tracée pour l'intervalle de t donné par $[-1, 3]$. Enfin, résoudre algébriquement le système d'équations paramétriques pour obtenir une équation reliant uniquement x et y, et vérifier que cette équation correspond bien au graphique.

On définit les équations paramétriques pour x et y, puis on utilise `makelist` pour créer l'ensemble des points discrets. Enfin, on génère le graphique incluant la courbe paramétrique continue :

```
(%i1) x(t):=t$
      y(t):=2*t-3$

(%i3) POINTS:makelist([x(0.1*n),y(0.1*n)],n,0,20);
(%o3) [[0,-3],[0.1,-2.8],[0.2,-2.6],[0.3,-2.4],[0.4,-2.2],
      [0.5,-2.0],[0.6,-1.8],[0.7,-1.6],[0.8,-1.4],[0.9,-1.2],
      [1.0,-1.0],[1.1,-0.8],[1.2,-0.6],[1.3,-0.4],[1.4,-0.2],
      [1.5,0.0],[1.6,0.2],[1.7,0.4],[1.8,0.6],[1.9,0.8],[2.0,1.0]]

(%i4) wxdraw2d(
      grid=true,
      xaxis=true,
      yaxis=true,
      title="Points sur la courbe paramétrique.",
      color=dark_grey,
       parametric(x(t),y(t),t,-1,3),
      color=red,
      point_type=7,
       points(POINTS)
      );
```



La courbe est parcourue dans le sens des t croissants. On observe, à partir de la liste des points discrets, que la direction le long de la courbe peut être résumée par « la courbe est parcourue vers le haut et vers la droite ». Enfin, on « élimine le paramètre » algébriquement pour obtenir une équation reliant uniquement x et y. On redéfinit les fonctions paramétriques comme des *équations* dans wxMaxima, puis on applique `eliminate` au système pour éliminer t.
Notez que le résultat de `eliminate` est implicitement posé égal à zéro – on utilise `solve` pour exprimer l'équation résultante sous une forme plus standard.

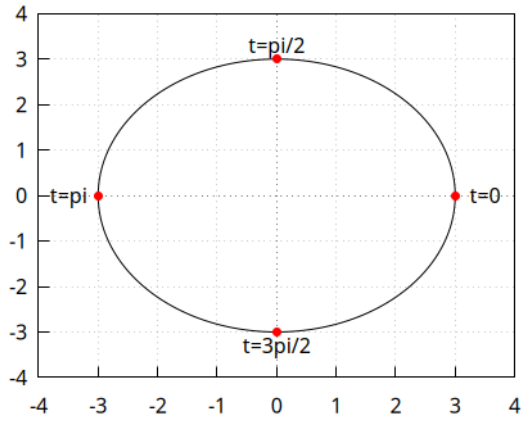
```
(%i5) Xeqn:x=x(t)$
      Yeqn:y=y(t)$
      eliminate([Xeqn,Yeqn],[t]);
(%o7) [-y+2*x-3]
(%i8) solve(%[1]=0,y);
(%o8) [y=2*x-3]
```

On obtient l'équation d'une droite de pente 2 et d'ordonnée à l'origine -3, comme on l'observe sur le graphique.

Exemple 5.1.2. Tracer la courbe paramétrique définie par $x(t) = 3\cos t$ et $y(t) = 3\sin t$ sur l'intervalle de t donné par $[0, 2\pi]$. Calculer les coordonnées des points en $t = 0, \frac{\pi}{2}, \pi, \frac{3\pi}{2}$ et commenter la direction parcourue sur la courbe. Enfin, exprimer l'équation de la courbe uniquement en termes de x et y.

```
(%i9) x(t):=3*cos(t)$
      y(t):=3*sin(t)$
(%i11) wxdraw2d(
       grid=true,
       xaxis=true,
       yaxis=true,
       xrange=[-4,4],
       yrange=[-4,4],
       dimensions=[600,600],
       nticks=600,
```

```
color=black,
 parametric(x(t),y(t),t,0,2*%pi),
color=red,
point_type=7,
 points([[x(0),y(0)],[x(%pi/2),y(%pi/2)],
 [x(%pi),y(%pi)],[x(3*%pi/2),y(3*%pi/2)]]),
color=black,
 label(["t=0",3.5,0]),
 label(["t=pi/2",0,3.3]),
 label(["t=pi",-3.5,0]),
 label(["t=3pi/2",0,-3.3])
);
```



On observe que la direction correspondant aux t croissants est le sens antihoraire. Enfin, on élimine le paramètre pour trouver une équation que nous reconnaitrons facilement en ce qui concerne cette courbe.
Notez que `eliminate` ne fonctionnera pas dans ce cas, car cette commande est dédiée uniquement aux équations polynomiales. On procède en utilisant une identité trigonométrique :

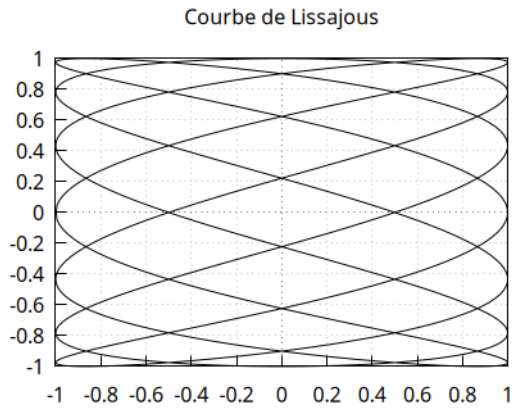
```
(%i12) Xeqn:x=x(t);
       Yeqn:y=y(t);
(%o12) x=3*cos(t)
(%o13) y=3*sin(t)
(%i14) Xeqn^2+Yeqn^2;
(%o14) y^2+x^2=9*sin(t)^2+9*cos(t)^2
(%i15) trigsimp(%);
(%o15) y^2+x^2=9
```

On reconnaît l'équation d'un cercle de rayon 3 centré à l'origine.

Exemple 5.1.3. Tracer la courbe paramétrique définie par $x(t) = \cos(7t)$ et $y(t) = \sin(3t)$. Choisir un intervalle pour t suffisamment grand pour que la courbe soit fermée.

Lorsque t varie de 0 à 2π, $x(t)$ varie sur 7 périodes et $y(t)$ varie sur 3 périodes. En $t = 2\pi$, les coordonnées reviennent à leur point de départ initial, donc il suffit de tracer la courbe sur $[0, 2\pi]$ pour obtenir la courbe complète. Ce type de courbe est connu sous le nom de figure de Lissajous :

```
(%i21) wxdraw2d(
       grid=true,
       xaxis=true,
       yaxis=true,
       title="Courbe de Lissajous",
       color=black,
       nticks=600,
       dimensions=[600,600],
        parametric(cos(7*t),sin(3*t),t,0,2*%pi)
       );
```



Dans les exercices, on utilisera une animation pour explorer le sens de parcours sur cette courbe.

5.2 Applications du calcul différentiel et intégral aux courbes paramétriques

5.2.1 Pente d'une courbe paramétrique

Lorsqu'une courbe est définie de manière paramétrique, on peut trouver la pente, $\frac{\mathrm{d}y}{\mathrm{d}x}$ en $t = a$, en utilisant les dérivées par rapport à t de x et y :

$$\frac{\mathrm{d}y}{\mathrm{d}x} = \frac{\mathrm{d}y/\mathrm{d}t}{\mathrm{d}x/\mathrm{d}t} = \frac{y'(a)}{x'(a)}$$

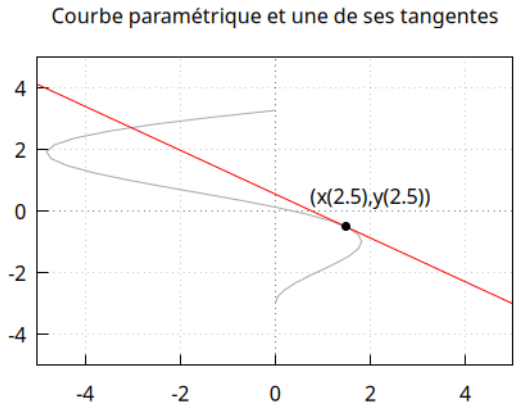
On peut utiliser la pente pour tracer la droite tangente à une courbe paramétrique en tout point souhaité.

Exemple 5.2.1. Une courbe paramétrique est définie par $x(t) = t\sin t$ et $y(t) = t - 3$ sur l'intervalle de t donné par $[0, 2\pi]$. Calculer la pente et l'équation de la droite tangente en $t = 2.5$, puis tracer la courbe ainsi que la droite tangente.

On définit la pente comme le rapport de $y'(t)$ et $x'(t)$, on calcule la pente en $t = 2.5$, et on calcule l'équation de la droite tangente en substituant $(x(2.5), y(2.5))$ dans la formule point-pente qui donne une équation de cette tangente. On utilise `float` pour obtenir des approximations décimales car `wxdraw2d` risque de ne pas achever ces calculs complexes avec les expressions exactes :

```
(%i1) x(t):=t*sin(t)$
      y(t):=t-3$

(%i3) diff(x(t),t);
(%o3) sin(t)+t*cos(t)
(%i4) x_prime(t):=''(%);
(%o4) x_prime(t):=sin(t)+t*cos(t)
(%i5) diff(y(t),t);
(%o5) 1
(%i6) y_prime(t):=''(%);
(%o6) y_prime(t):=1

(%i7) y_prime(t)/x_prime(t);
(%o7) 1/(sin(t)+t*cos(t))
(%i8) SLOPE(t):=''(%);
(%o8) SLOPE(t):=1/(sin(t)+t*cos(t))

(%i9) slope:float(SLOPE(2.5));
(%o9) -0.71205449419157
(%i10) tanline:float(slope*(x-x(2.5))+y(2.5));
(%o10) -0.71205449419157*(x-1.496180360259891)-0.5

(%i11) wxdraw2d(
       grid=true,
       xaxis=true,
       yaxis=true,
```

```
xrange=[-5,5],
yrange=[-5,5],
title="Courbe paramétrique et une de ses tangentes",
color=dark_grey,
 parametric(x(t),y(t),t,0,2*%pi),
color=red,
 explicit(tanline,x,-5,15),
color=black,
point_type=7,
 points([[x(2.5),y(2.5)]]),
 label(["(x(2.5),y(2.5))",2,.5])
);
```

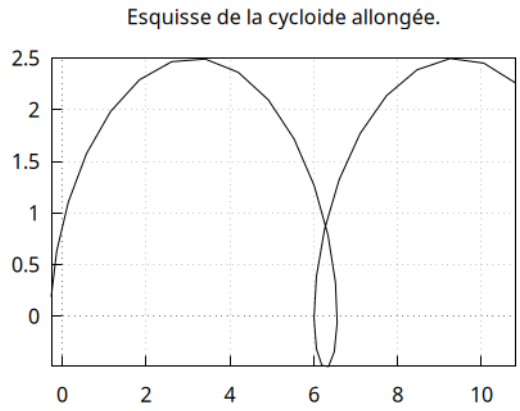


Exemple 5.2.2. Les équations $x(t) = t - 1.5\sin t$ et $y(t) = 1 - 1.5\cos t$ définissent une *cycloïde allongée*. Sur l'intervalle de t donné par $[1, 10]$, trouver toutes les valeurs de t pour lesquelles une droite tangente a une pente nulle ou une pente indéfinie. Enfin, tracer la cycloïde allongée ainsi que les droites tangentes.

On calcule la pente en fonction de t et on réalise une esquisse préliminaire de la courbe pour faciliter notre recherche des points particuliers définis dans l'énoncé :

```
(%i1) x(t):=t-1.5*sin(t)$
      y(t):=1-1.5*cos(t)$
      diff(x(t),t)$
      x_prime(t):=''(%);
      diff(y(t),t)$
      y_prime(t):=''(%);
(%o4) x_prime(t):=1-1.5*cos(t)
(%o6) y_prime(t):=1.5*sin(t)

(%i7) y_prime(t)/x_prime(t)$
      SLOPE(t):=''(%);
(%o8) SLOPE(t):=(1.5*sin(t))/(1-1.5*cos(t))
```

```
(%i9) wxdraw2d(
      grid=true,
      xaxis=true,
      yaxis=true,
      title="Esquisse de la cycloide allongée.",
      color=black,
       parametric(x(t),y(t),t,1,10)
      );
```



Maintenant, on localise les valeurs de t pour lesquelles le numérateur et le dénominateur de `SLOPE` s'annulent. Le numérateur, $1.5 \sin t$, s'annule en $t = \pi$, $t = 2\pi$ et $t = 3\pi$. On utilise `find_root` pour localiser les valeurs de t pour lesquelles le dénominateur s'annule. Les tangentes verticales se produisent juste avant et juste après $t = 2\pi$ (l'emplacement de la tangente horizontale), donc on se place sur les intervalles $[4, 2\pi]$ et $[2\pi, 8]$:

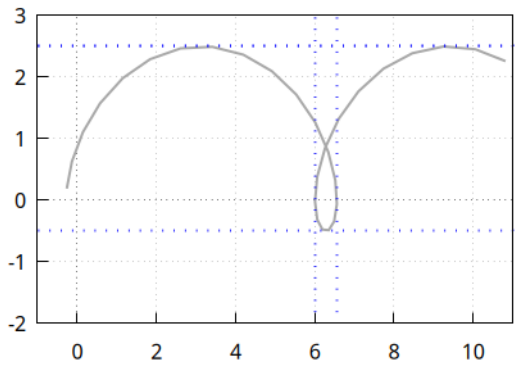
```
(%i10) find_root(denom(SLOPE(t)),t,4,2*%pi);
(%o10) 5.442116636611656

(%i11) find_root(denom(SLOPE(t)),t,2*%pi,8);
(%o11) 7.124253977747517
```

On a donc des tangentes verticales en $t \approx 5.44$ et $t \approx 7.12$. Enfin, on produit un graphique de la cycloïde allongée ainsi que des droites tangentes (rappelons que les droites tangentes verticales doivent être définies paramétriquement) :

```
(%i12) wxdraw2d(
       grid=true,
       xaxis=true,
       yaxis=true,
       yrange=[-2,3],
       xrange=[-1,11],
       title="Cycloide allongée avec ses tangentes horizontales et verticales",
       color=dark_grey,
       line_width=2,
        parametric(x(t),y(t),t,1,10),
```

```
line_width=1,
line_type=dots,
color=red,
 explicit(y(%pi),x,-1,11),
 explicit(y(2*%pi),x,-1,11),
 explicit(y(3*%pi),x,-1,11),
 parametric(x(5.442),t,t,-2,3),
 parametric(x(7.124),t,t,-2,3)
 );
```

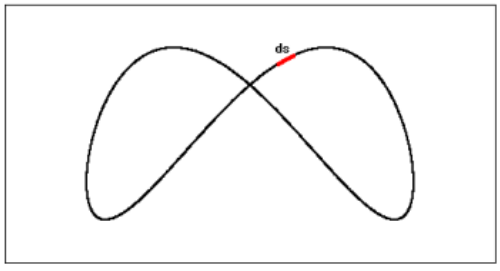
Cycloide allongée avec ses tgt horizontales et verticales



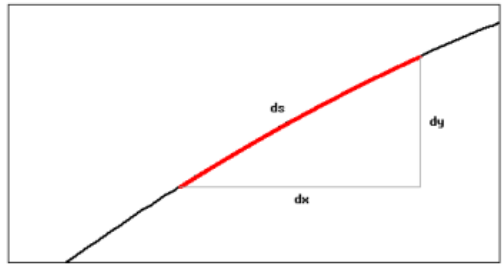
5.2.2 Longueur d'arc d'une courbe paramétrique

On aborde le problème de la longueur d'arc en visualisant un petit élément d'arc sur la courbe définie par les équations paramétriques $x(t)$ et $y(t)$:

Courbe paramétrique avec un élément d'arc ds



Décomposition de l'élément ds



Lorsqu'on effectue un zoom sur ds, on constate qu'il peut être décomposé en composantes selon x et y, ce qui permet d'appliquer le théorème de Pythagore pour obtenir $\mathrm{d}s = \sqrt{(\mathrm{d}x)^2 + (\mathrm{d}y)^2}$. Cette fois, on procède en factorisant dt hors de la racine carrée pour obtenir $\mathrm{d}s = \sqrt{\left(\frac{\mathrm{d}x}{\mathrm{d}t}\right)^2 + \left(\frac{\mathrm{d}y}{\mathrm{d}t}\right)^2}\ \mathrm{d}t$.

Enfin, on établit la longueur d'arc sous forme d'intégrale pour l'intervalle de t donné par $[a, b]$:

$$S = \int \mathrm{d}s = \int_a^b \sqrt{(x'(t))^2 + (y'(t))^2}\ \mathrm{d}t$$

Exemple 5.2.3. Calculer la longueur d'arc de l'ellipse définie par les équations paramétriques $x(t) = 3\cos t$ et $y(t) = 1.5\sin t$ sur l'intervalle de t donné par $[0, 2\pi]$.

On définit $x(t)$ et $y(t)$, on applique `diff`, et on demande à Maxima de calculer l'intégrale :

```
(%i13) x(t):=3*cos(t)$
       y(t):=1.5*sin(t)$

(%i15) diff(x(t),t)$
       x_prime:%;
(%o16) -3*sin(t)

(%i17) diff(y(t),t)$
       y_prime:%;
(%o18) 1.5*cos(t)

(%i19) ratprint:false$

(%i20) integrate(sqrt(x_prime^2+y_prime^2),t,0,2*%pi);
```

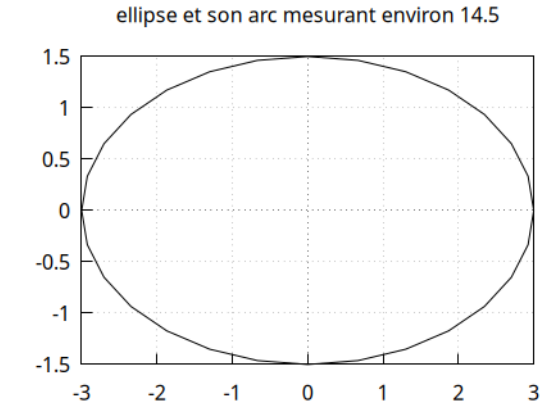
(%o20) $\displaystyle\int_0^{2\,\pi} \sqrt{9\sin\left(t\right)^2 + 2.25\cos\left(t\right)^2}dt$

wxMaxima échoue à calculer l'intégrale, donc on utilise `quad_qag` à la place.

```
(%i21) quad_qag(sqrt(x_prime^2+y_prime^2),t,0,2*%pi,1);
(%o21) [14.53267233082058,1.0286549418261626*10^-7,165,0]
```

On obtient une longueur d'arc pour $S \approx 14.5$. Enfin, on réalise une représentation de l'ellipse pour s'assurer que notre réponse est cohérente :

```
(%i22) wxdraw2d(
       grid=true,
       xaxis=true,
       yaxis=true,
       title="ellipse et son arc mesurant environ 14.5",
       color=black,
        parametric(x(t),y(t),t,0,2*%pi)
       );
```


ellipse et son arc mesurant environ 14.5
1.5
1
0.5
0
-0.5
-1
-1.5
-3
-2
-1
0
1
2
3

5.3 Coordonnées polaires

5.3.1 Coordonnées polaires et transformation des coordonnées

Pour définir un point en coordonnées polaires, on précise une distance à l'origine, r, et une direction, θ (mesurée en radians dans le sens contraire des aiguilles d'une montre à partir de l'axe des x positifs). Le point est alors donné par un couple ordonné (r, θ). On peut spécifier un point en coordonnées polaires d'une infinité de façons en ajoutant des multiples de 2π à θ, et on peut même utiliser des valeurs négatives de r ou de θ.

On peut passer des coordonnées polaires aux coordonnées rectangulaires (et inversement) en utilisant des formules trigonométriques :

$$\text{De polaires à rectangulaires :} \qquad x = r \cdot \cos\theta \qquad y = r \cdot \sin\theta$$

$$\text{De rectangulaires à polaires :} \qquad r = \sqrt{x^2 + y^2} \qquad \theta = \tan^{-1}\tfrac{y}{x}$$

Notons que la fonction inverse de tangente $\tan^{-1}$ (qui est en fait la fonction réciproque de tangente, c'est à dire arctangente, notée atan) ne fournit que des valeurs de θ sur le domaine restreint $\left[-\frac{\pi}{2}, \frac{\pi}{2}\right]$, donc il faut ajuster soigneusement la réponse lorsque θ n'appartient pas à ce domaine.

Exemple 5.3.1. Convertissez les couples ordonnés de coordonnées polaires suivants en coordonnées rectangulaires, puis représentez graphiquement les points correspondant :

a. $\left(3, \frac{\pi}{4}\right)$ b. $\left(2, -\frac{2\pi}{3}\right)$ c. $\left(-1, \frac{\pi}{6}\right)$ d. $(1, 5\pi)$

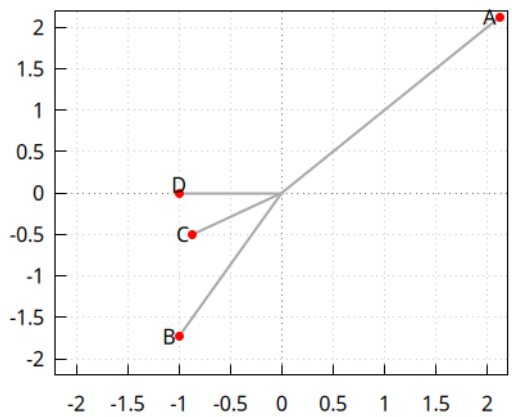
Nous définissons les conversions « polaires vers rectangulaires » par des fonctions, puis nous attribuons un nom à chacun des points définis par ses coordonnées rectangulaires :

```
(%i1) x(r,t):=r*cos(t)$
      y(r,t):=r*sin(t)$

      pointA:[x(3,%pi/4),y(3,%pi/4)];
      pointB:[x(2,-2*%pi/3),y(2,-2*%pi/3)];
      pointC:[x(-1,%pi/6),y(-1,%pi/6)];
      pointD:[x(1,5*%pi),y(1,5*%pi)];

(%o3) [3/sqrt(2),3/sqrt(2)]
(%o4) [-1,-sqrt(3)]
(%o5) [-sqrt(3)/2,-1/2]
(%o6) [-1,0]
```

Enfin, nous représentons les quatre points en utilisant les coordonnées rectangulaires. Nous dessinons aussi des lignes en gris pour des valeurs constantes de θ afin de faciliter la visualisation de chaque angle et de chaque rayon (les lignes sont définies paramétriquement dans `wxdraw2d`).

```
(%i7) wxdraw2d(
      grid=true,
```

```
dimensions=[600,600],
xrange=[-2.2,2.2],
yrange=[-2.2,2.2],
xaxis=true,
yaxis=true,
color=dark_grey,
line_width=2,
 parametric(cos(%pi/4)*t,sin(%pi/4)*t,t,0,3),
 parametric(-t*cos(%pi/3),-sin(%pi/3)*t,t,0,2),
 parametric(-cos(%pi/6)*t,-sin(%pi/6)*t,t,0,1),
 parametric(t,0,t,-1,0),
color=red,
point_type=7,
 points([pointA,pointB,pointC,pointD]),
color=black,
 label(["A", pointA[1]-.1,pointA[2]]),
 label(["B", pointB[1]-.1,pointB[2]]),
 label(["C", pointC[1]-.1,pointC[2]]),
 label(["D", pointD[1],pointD[2]+.1])
);
```



La valeur négative de θ pour le point défini en b. signifie que l'angle est mesuré dans le sens des aiguilles d'une montre à partir de l'axe des x positifs. La valeur négative de r pour le point défini en c. signifie que ce dernier est situé dans la direction *opposée* à θ. Enfin, l'angle 5π pour le point défini en d. est équivalent à π puisque les multiples de 2π ne modifient pas la direction.

Exemple 5.3.2. Utilisez des approximations décimales pour exprimer le point défini en coordonnées rectangulaires par $[2, 3]$ en coordonnées polaires de quatre façons différentes : en utilisant un r positif/négatif et un θ positif/négatif. Vérifiez dans chaque cas que les coordonnées rectangulaires sont correctes.

Nous commençons par le cas où r est positif et θ est positif :

```
(%i8) kill(all)$
```

```
(%i1) x(r,t):=r*cos(t)$
      y(r,t):=r*sin(t)$

(%i3) R:float(sqrt(2^2+3^2));
      T:float(atan(3/2));
(%o3) 3.605551275463989
(%o4) 0.98279372324733

(%i5) POINT:[x(R,T),y(R,T)];
(%o5) [2.0,3.0]
```

Pour le cas où r est positif et θ est négatif, il suffit de tourner dans le sens des aiguilles d'une montre de 2π :

```
(%i9) R:float(sqrt(2^2+3^2));
      T:float(atan(3/2)-2*%pi);
(%o9) 3.605551275463989
(%o10) -5.300391583932258

(%i11) POINT:[x(R,T),y(R,T)];
(%o11) [2.0,3.0]
```

Pour le cas où r est négatif et θ est positif, nous devons passer dans la direction opposée à l'angle original en ajoutant π :

```
(%i12) R:-float(sqrt(2^2+3^2));
       T:float(atan(3/2)+%pi);
(%o12) -3.605551275463989
(%o13) 4.124386376837122

(%i14) POINT:[x(R,T),y(R,T)];
(%o14) [2.000000000000001,2.999999999999999]
```

Enfin, pour le cas où r est négatif et θ est négatif, nous devons positionner la direction opposée à l'angle original en soustrayant π :

```
(%i15) R:-float(sqrt(2^2+3^2));
T:float(atan(3/2)-%pi);
(%o15) -3.605551275463989
(%o16) -2.158798930342464

(%i17) POINT:[x(R,T),y(R,T)];
(%o17) [1.999999999999999,3.0]
```

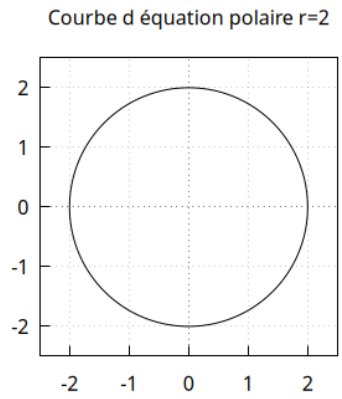
Dans chaque cas, il existe une infinité de réponses correctes correspondant à des rotations supplémentaires de l'angle d'un multiple entier de 2π.

5.3.2 Représenter les courbes en coordonnées polaires

Une équation reliant r et θ définit une courbe polaire. wxMaxima peut rapidement tracer des courbes polaires si r est défini explicitement en fonction de θ.

Exemple 5.3.3. Tracez la courbe d'équation polaire $r = 2$, en utilisant `dimensions` pour imposer un rapport d'aspect carré. Convertissez l'équation en utilisant les coordonnées rectangulaires pour vérifier ce que vous observez sur le graphique.

```
(%i1) wxdraw2d(
      grid=true,
      dimensions=[600,600],
      proportional_axes =xy,
      nticks=1000,
      xaxis=true,
      yaxis=true,
      xrange=[-2.5,2.5],
      yrange=[-2.5,2.5],
      title="Courbe d'équation polaire r=2",
      color=black,
       polar(2,t,0,2*%pi)
      );
```



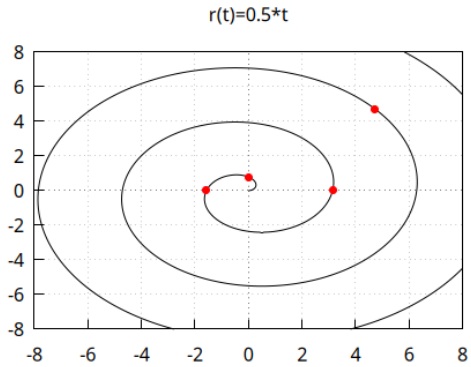
Nous convertissons maintenant en utilisant les coordonnées rectangulaires :

```
(%i2)  x(t):=2*cos(t)$
       y(t):=2*sin(t)$
       EQN:X^2+Y^2=(x(t))^2+(y(t))^2;
(%o4) Y^2+X^2=4*sin(t)^2+4*cos(t)^2
(%i5) trigsimp(%);
(%o5) Y^2+X^2=4
```

Nous reconnaissons l'équation d'un cercle de rayon 2 centré sur l'origine.

Exemple 5.3.4. Tracez la courbe d'équation polaire $r = 0.5 \cdot \theta$ sur l'intervalle $[0, 20]$ de θ. Positionnez explicitement plusieurs points sur le graphique pour illustrer comment ces points sont obtenus à partir de l'équation polaire.

```
(%i6) R(t):=0.5*t$
(%i7) x(r,t):=r*cos(t)$
      y(r,t):=r*sin(t)$
(%i9) POINTS:[[x(R(%pi/2),(%pi/2)),y(R(%pi/2),(%pi/2))],
        [x(R(%pi),(%pi)),y(R(%pi),(%pi))],
        [x(R(2*%pi),(2*%pi)),y(R(2*%pi),(2*%pi))],
        [x(R(17*%pi/4),(17*%pi/4)),y(R(17*%pi/4),(17*%pi/4))]]$
(%i10) wxdraw2d(
       grid=true,
       dimensions=[600,600],
       xrange=[-8,8],
       yrange=[-8,8],
       nticks=1000,
       xaxis=true,
       yaxis=true,
       title="r(t)=0.5*t",
       color=black,
        polar(R(t),t,0,20),
       color=red,
       point_type=7,
        points(POINTS)
       );
```

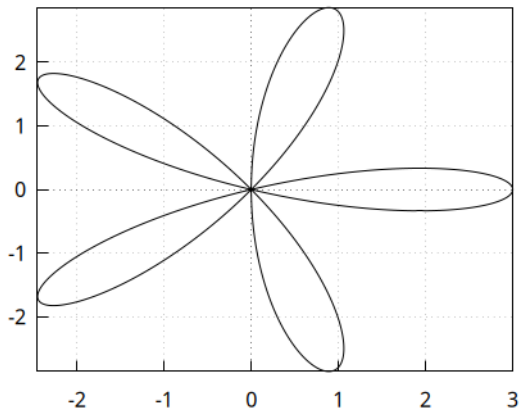


Les quatre points représentés sur le graphique nous aident à comprendre comment la courbe est générée pour chaque valeur de θ. Par exemple, lorsque $\theta = 2\pi$, $r(2\pi) = 0.5 \cdot 2\pi \approx 3.14$, donc on en déduit l'existence d'un point de la courbe obtenu en « pointant » dans la direction de $\theta = 2\pi$ à une distance légèrement supérieure à 3 unités de l'origine.

Exemple 5.3.5. La courbe définie en polaire par $r(t) = 3\cos(5\theta)$ est un exemple de *rosace*. Tracez $r(t)$ sur l'intervalle $[0, \pi]$ ainsi que les cinq points situés aux extrémités des « pétales ».

Nous commençons par réaliser une esquisse de la courbe :

```
(%i1) wxdraw2d(
      grid=true,
      xaxis=true,
      yaxis=true,
      color=black,
      nticks=1000,
      polar(3*cos(5*t),t,0,%pi)
      );
```

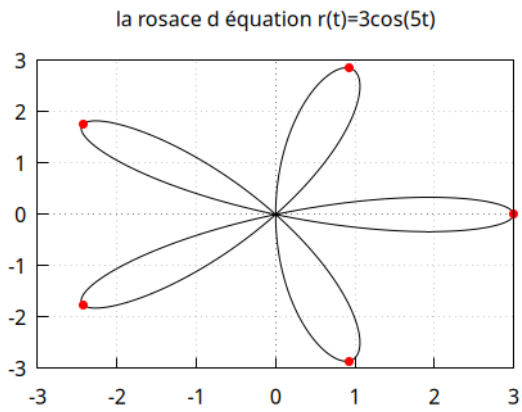


Les extrémités des pétales correspondent à la valeur maximale de r, qui est atteinte lorsque la fonction cosinus atteint sa valeur maximale de 1 ou sa valeur minimale de -1. Nous déterminons les angles correspondants en résolvant $5\theta = n \cdot \pi$ pour $n = 0, 1, 2, \ldots, 4$. Notez que $n = 5$ donne $\theta = \pi$, ce qui donne $r = -3$: il s'agit exactement du même point que celui donné par $n = 0$ où $r = 3$, donc nous excluons le cas $n = 5$ de la liste.

Nous utilisons `makelist` en incluant les conversions de coordonnées polaires en coordonnées rectangulaires aux cinq angles particuliers, puis nous traçons la rosace dans une fenêtre carrée avec ces points particuliers :

```
(%i2) x(r,t):=r*cos(t)$
      y(r,t):=r*sin(t)$
      R(t):=3*cos(5*t)$
(%i5) POINTS:makelist(float([x(R(n*%pi/5),n*%pi/5),y(R(n*%pi/5),n*%pi/5)]),n,0,4)$

(%i6) wxdraw2d(
      grid=true,
      dimensions=[600,600],
      xaxis=true,
      yaxis=true,
      xrange=[-3,3],
      yrange=[-3,3],
      nticks=1000,
      title="la rosace d équation r(t)=3cos(5t)",
```

```
color=black,
 polar(R(t),t,0,%pi),
color=red,
point_type=7,
 points(POINTS)
);
```



5.4 Applications du calcul différentiel et intégral aux courbes polaires

5.4.1 Pente en coordonnées polaires

A partir de l'équation polaire d'une courbe, $r(\theta)$, on peut calculer la pente en tout point en substituant les formules $x = r\cos\theta$ et $y = r\sin\theta$ dans $\frac{\mathrm{d}y}{\mathrm{d}x}$. On applique la règle du produit pour la dérivation, puisque x et y dépendent à la fois de r et de θ :

$$\frac{\mathrm{d}y}{\mathrm{d}x} = \frac{\frac{\mathrm{d}y}{\mathrm{d}\theta}}{\frac{\mathrm{d}x}{\mathrm{d}\theta}} = \frac{\frac{\mathrm{d}}{\mathrm{d}\theta}(r\sin\theta)}{\frac{\mathrm{d}}{\mathrm{d}\theta}(r\cos\theta)} = \frac{\frac{\mathrm{d}r}{\mathrm{d}\theta}\cdot\sin\theta + r\cdot\cos\theta}{\frac{\mathrm{d}r}{\mathrm{d}\theta}\cdot\cos\theta - r\cdot\sin\theta}$$

Exemple 5.4.1. Pour la courbe d'équation polaire $r(\theta) = \cos^2\theta + \sin^3(2\theta)$, calculez la pente en $\theta = \frac{\pi}{4}$. Tracez la courbe sur l'intervalle $[0, 2\pi]$ pour θ, en incluant la tangente en $\theta = \frac{\pi}{4}$.

Nous définissons $r(\theta)$ et le substituons dans la formule de la pente :

```
(%i1) r(t):=(cos(t))^2+(sin(2*t))^3$

(%i2) DERIV:(diff(r(t),t)*sin(t)+r(t)*cos(t))/
            (diff(r(t),t)*cos(t)-r(t)*sin(t));
```

$$\text{(\%o2)}\quad \frac{\cos(t)\left(\sin(2\,t)^3+\cos(t)^2\right)+\sin(t)\left(6\cos(2\,t)\sin(2\,t)^2-2\cos(t)\sin(t)\right)}{\cos(t)\left(6\cos(2\,t)\sin(2\,t)^2-2\cos(t)\sin(t)\right)-\sin(t)\left(\sin(2\,t)^3+\cos(t)^2\right)}$$

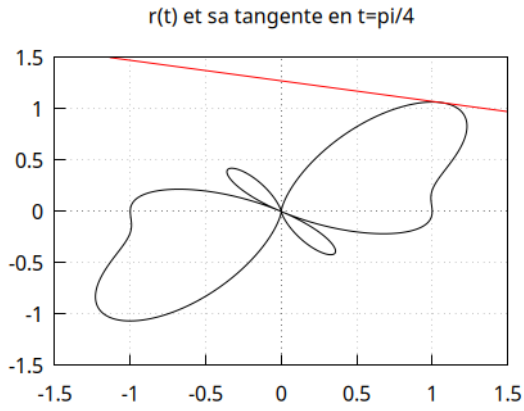
```
(%i3) SLOPE:subst(%pi/4,t,DERIV);
(%o3) -1/5
```

Nous trouvons une pente de $-\frac{1}{5}$ en $\theta = \frac{\pi}{4}$.

Afin de tracer la tangente, nous devons travailler en coordonnées rectangulaires. Nous déterminons le point de tangence et le substituons dans la formule point-pente pour obtenir une équation de la tangente. Enfin, nous produisons un graphique :

```
(%i4) x(t):=r(t)*cos(t)$
      y(t):=r(t)*sin(t)$
      TANLINE:(x-x(%pi/4))*SLOPE+y(%pi/4)$

(%i5) wxdraw2d(
      grid=true,
      dimensions=[600,600],
      nticks=1000,
      xaxis=true,
      yaxis=true,
      xrange=[-1.5,1.5],
      yrange=[-1.5,1.5],
      title="r(t) et sa tangente en t=pi/4",
      color=black,
       polar(r(t),t,0,2*%pi),
```

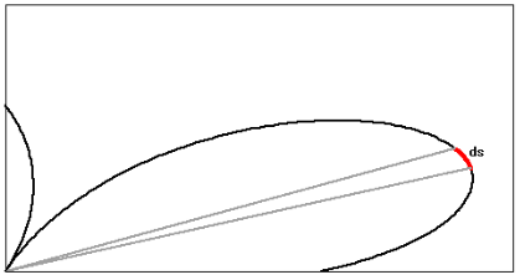
```
color=red,
 explicit(TANLINE,x,-2,2)
);
```



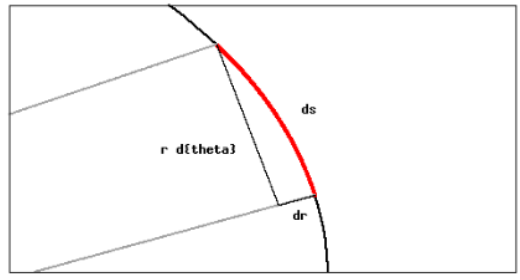
5.4.2 Longueur d'arc d'une courbe en polaire

Une fois encore, nous abordons le problème de la longueur d'un arc en visualisant un petit élément d'arc sur une courbe. La courbe a pour équation polaire $r(\theta)$, et ds est balayé sur un petit angle dθ. Cette fois, il est plus utile de décomposer l'élément d'arc en composantes *tangentielle* et *radiale*. La composante tangentielle de ds est donnée par $r \cdot \mathrm{d}\theta$, et la composante radiale est simplement dr :

Courbe polaire avec un élément d'arc ds



Décomposition de l'élément ds



Nous appliquons le théorème de Pythagore pour exprimer ds en fonction de dθ :

$$\mathrm{d}s = \sqrt{(r\mathrm{d}\theta)^2 + (\mathrm{d}r)^2} = \sqrt{r^2 + \left(\frac{\mathrm{d}r}{\mathrm{d}\theta}\right)^2} \cdot \mathrm{d}\theta$$

Enfin, la longueur d'arc sur l'intervalle $[\theta_1, \theta_2]$ de θ est donnée par :

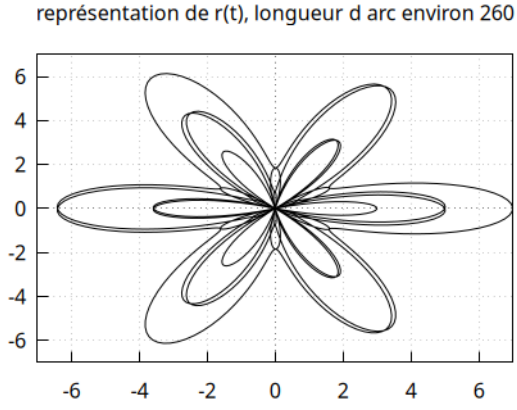
$$S = \int \mathrm{d}s = \int_{\theta_1}^{\theta_2} \sqrt{r^2 + \left(\frac{\mathrm{d}r}{\mathrm{d}\theta}\right)^2} \cdot \mathrm{d}\theta$$

Exemple 5.4.2. Trouver un intervalle en θ qui permet de tracer exactement une fois la courbe d'équation polaire $r(\theta) = 5\cos^2(3\theta) - 2\sin(0.75\theta)$. Calculer la longueur d'arc de $r(\theta)$ sur cet intervalle et réaliser le graphique correspondant.

Nous commençons à $\theta = 0$ et calculons un intervalle en θ sur lequel chaque fonction trigonométrique composant $r(\theta)$ effectue un nombre entier de périodes. Le premier terme de la différence a une période de $\frac{\pi}{3}$, tandis que le second terme a une période de $\frac{8\pi}{3}$. Ainsi, chaque fonction retournera à son état initial lorsque l'on rajoutera $\frac{8\pi}{3}$. Pour que la courbe soit complète, chaque fonction trigonométrique doit reprendre la même valeur que dans son état initial non seulement par rapport au même angle mais *aussi dans la direction initiale* – c'est-à-dire que cet angle doit être un multiple de 2π. Cela se produira après trois intervalles de $\frac{8\pi}{3}$, donc $[0, 8\pi]$ est l'intervalle nécessaire pour obtenir toute la courbe et ne la tracer exactement qu'une fois.

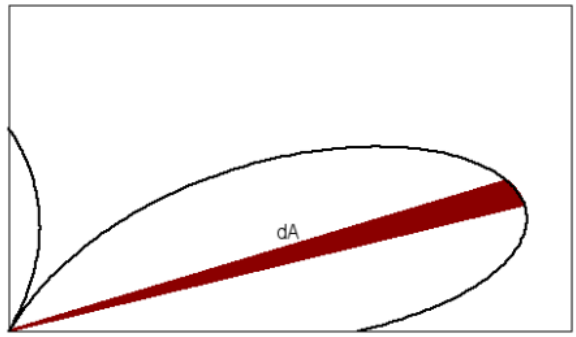
Nous appliquons `quad_qag` car wxMaxima n'arrive pas à finir le calcul si on utilise la commande `integrate` :

```
(%i1) r(t):=5*(cos(3*t))^2-2*sin(0.75*t)$
      DERIV:diff(r(t),t)$
      quad_qag(sqrt((r(t))^2+DERIV^2),t,0,8*%pi,2);
(%o3) [260.3422173127943,2.5463420336505348*10^-6,7497,0]

(%i4) wxdraw2d(
      grid=true,
      xaxis=true,
      yaxis=true,
      dimensions=[600,600],
      nticks=1000,
      xrange=[-7,7],
      yrange=[-7,7],
      title="représentation de r(t), longueur d arc environ 260",
      color=black,
       polar(r(t),t,0,8*%pi)
      );
```



5.4.3 Aire en coordonnées polaires

Pour calculer l'aire délimitée par une courbe en polaire, nous visualisons un petit élément d'aire $\mathrm{d}A$ défini par une fine tranche d'angle $\mathrm{d}\theta$ et de rayon $r(\theta)$.



$\mathrm{d}A$ est presque égal à un *secteur* de disque, ainsi nous pouvons trouver son aire en utilisant des résultats classiques de géométrie : le rapport de $\mathrm{d}\theta$ à 2π est égal au rapport de $\mathrm{d}A$ sur l'aire totale d'un disque de même rayon :

$$\frac{\mathrm{d}A}{\pi r^2} = \frac{\mathrm{d}\theta}{2\pi} \implies \mathrm{d}A = \frac{1}{2} r^2 \mathrm{d}\theta$$

Enfin, nous utilisons une intégrale pour calculer l'aire délimitée par la courbe sur $[\theta_1, \theta_2]$:

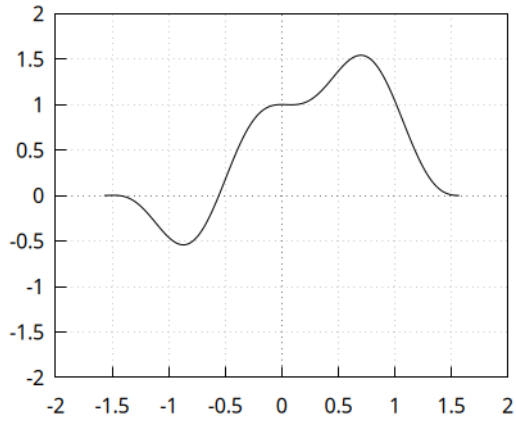
$$A = \int \mathrm{d}A = \frac{1}{2} \int_{\theta_1}^{\theta_2} r^2 \mathrm{d}\theta$$

Exemple 5.4.3. Calculer l'aire délimitée par l'un des grands « pétales » de la courbe d'équation polaire $r(\theta) = \cos^2\theta + \sin^3(2\theta)$ (la courbe de l'exemple 5.4.1).
*Tracer $r(\theta)$ sur $[0, 2\pi]$, et hachurer l'aire calculée.

Nous commençons par déterminer l'intervalle en θ qui délimite le pétale. Nous pouvons trouver facilement que $r(t) = 0$ en $\pm\frac{\pi}{2}$, mais il devrait y avoir d'autres solutions entre $-\frac{\pi}{2}$ et 0, intervalle sur lequel le sinus est négatif. Nous cherchons donc la racine la plus proche de 0 – notre pétale est tracé depuis cette valeur racine jusqu'au zéro suivant en $\frac{\pi}{2}$

Pour aider à la recherche, nous traçons d'abord une première esquisse de la courbe :

```
(%i1) r(t):=(cos(t))^2+(sin(2*t))^3$
(%i2) wxdraw2d(
      grid=true,
      xaxis=true,
      yaxis=true,
      xrange=[-2,2],
      yrange=[-2,2],
      color=black,
       explicit(r(t),t,-%pi/2,%pi/2)
      );
```



Nous observons l'existence d'une racine quelque part près de -0.5, et nous utilisons `find_root` pour en obtenir une approximation :

```
(%i3) find_root(r(t),t,-.7,-.3);
(%o3) -0.55623041931838
```

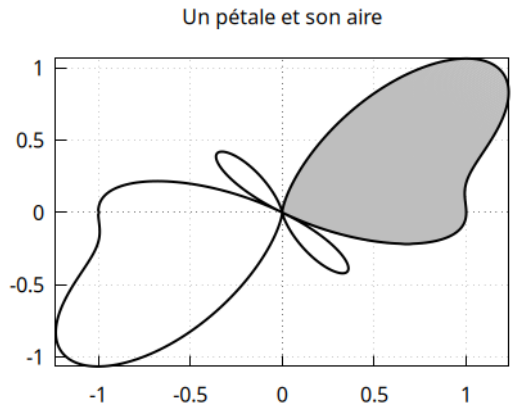
L'intervalle définissant le pétale est donc $[-0.556, \frac{\pi}{2}]$. Nous calculons maintenant l'aire délimitée par ce pétale :

```
(%i4) float(integrate(0.5*(r(t))^2,t,-0.556,%pi/2));
(%o4) 1.026989023118201
```

Nous obtenons une valeur d'environ 1 unité pour cette aire.

Enfin, nous réalisons le graphique demandé comportant l'aire hachurée définie par $r(t)$. wxMaxima ne possède pas d'équivalent à `filled_func` en coordonnées polaires. Nous utilisons plutôt `makelist` pour remplir la région avec de nombreux segments de droite radiaux rapprochés.

```
(%i5) x(t):=r(t)*cos(t)$
      y(t):=r(t)*sin(t)$
      RADIUS(t):=(y(t)/x(t))*(x-x(t))+y(t)$
      RADII:makelist(explicit(RADIUS(0.01*n),x,0,x(0.01*n)),n,-55,157)$

(%i6) wxdraw2d(
      grid=true,
      xaxis=true,
      yaxis=true,
      dimensions=[600,600],
      nticks=600,
      title="Un pétale et son aire",
      line_width=2,
      color=grey,
       RADII,
      color=black,
       polar(r(t),t,0,2*%pi)
      );
```



Nous pouvons estimer à partir du graphique avec l'aire hachurée que $A \approx 1$ est cohérent – l'aire du pétale semble à peu près équivalente à celle d'un rectangle de 1 unité par 1 unité sur le graphique.

5.5 Exercices du Chapitre 5

1. Pour la courbe définie par les équations paramétriques $x(t) = t^2$ et $y(t) = t - 2$, générer un ensemble de 21 points sur $[0, 2]$ en utilisant `makelist`. Tracer ces points avec la courbe paramétrique, et observer la liste des points pour déterminer le sens de parcours de la courbe.
2. Pour la courbe définie par les équations paramétriques $x(t) = 3\cos(2t)$, $y(t) = 5\sin(2t)$, déterminer l'intervalle minimal en t (à partir de zéro) sur lequel la courbe est fermée. Tracer cette courbe et construire plusieurs points afin d'illustrer le sens de parcours sur cette courbe. Enfin, transformer les équations et utiliser `trigsimp` pour éliminer le paramètre et trouver une équation ne dépendant que de x et y.
3. Tracer la courbe de Lissajous donnée par $x(t) = \cos t$, $y(t) = \sin(3t)$. Quel est le nombre de périodes pour chacune de ces fonctions lorsque t varie de 0 à 2π ?
4. Pour la courbe paramétrique de l'exemple précédent, trouver tous les points en lesquels la tangente a une pente égale à 1. Il sera utile par exemple de réaliser un tracé de `SLOPE(t)` et d'utiliser `find_root` pour obtenir toutes les valeurs de t pour lesquelles la pente est égale à 1. Enfin, construire le tracé de la courbe entière en noir ainsi que des droites tangentes en rouge.
5. Calculer la longueur d'arc de la courbe paramétrique de l'exemple précédent.
6. Les équations paramétriques $x(t) = t - 0.8\sin t$, $y(t) = 1 - 0.8\cos t$ définissent une *cycloïde raccourcie*. Trouver tous les points en lesquels les tangentes sont horizontales sur $[0, 2\pi]$, puis tracer la courbe en noir avec les tangentes horizontales représentées en rouge.
7. Calculer la longueur d'arc de la courbe paramétrique de l'exercice précédent.
8. Les équations paramétriques $x(t) = 1.7\cos t + 0.4\cos\frac{20t}{3}$, $y(t) = 1.7\sin t - 0.4\sin\frac{20t}{3}$ définissent une *hypocycloïde allongée* sur $[0, 6\pi]$. Tracer cette courbe et calculer sa longueur d'arc.
9. Tracer la courbe appelée *cardioïde* définie en coordonnées polaires par $r(\theta) = 1 - \sin\theta$. Tracer toutes les tangentes horizontales et verticales ainsi que la courbe. De plus, tracer et étiqueter clairement les points de tangence.
10. Calculez la longueur d'arc et l'aire intérieure de la courbe cardioïde de l'exemple précédent.
11. Tracez les courbes en rosace $r(\theta) = \cos(n\theta)$ pour $n = 2, 3, 4, 5$. Quel motif remarquez-vous dans les tracés ? Emettez une hypothèse pour $n = 16$ et produisez un traphique pour tester votre hypothèse.
12. Pour le cas $n = 16$ de l'exercice précédent, calculez l'aire d'un seul pétale et coloriez-le dans le style de l'Exemple 5.4.3.
13. Lorsqu'un objet est lancé depuis l'origine avec une vitesse v_0 selon un angle θ, la trajectoire est donnée par un ensemble d'équations paramétriques : $x(t) = v_0\cos\theta \cdot t$ et $y(t) = v_0\sin\theta \cdot t - \frac{1}{2}gt^2$, où g est l'accélération de la pesanteur. En utilisant une vitesse de lancement de 100 m/s, un angle de lancement de 40° et $g \approx 9.8$ m/s^2, tracer la trajectoire résultante de $t = 0$ jusqu'au moment où le projectile « atterrit » (c'est à dire lorsque $y = 0$).
14. Utiliser `makelist` pour tracer 90 trajectoires (toutes sur le même graphique) pour des projectiles lancés depuis l'origine à 100 m/s avec des angles initiaux

$\theta = 1°, 2°, \ldots, 90°$. On supposera que les projectiles atterrissent en $y = 0$. Quel angle semble donner la portée maximale pour le projectile ?

15. L'équation différentielle d'un oscillateur harmonique non amorti est donnée par $x''(t) = -\frac{k}{m}x(t)$, où k est la constante de raideur du ressort et m est la masse de l'oscillateur. Pour un oscillateur de masse 0.2 kg et de constante de raideur 9 N/m, résoudre cette équation différentielle en utilisant `ode2`. Utiliser `ic2` pour appliquer les conditions initiales $x(0) = 0.15$ m et $v(0) = x'(0) = 2$ m/s. Tracer un graphique montrant $x(t)$ et $v(t)$ dans deux couleurs différentes sur l'intervalle de t $[0, 10]$.

    $x(t)$ et $v(t)$ forment un ensemble d'équations paramétriques pour l'oscillateur harmonique. On peut visualiser tout état de l'oscillateur en traçant les couples (x, v) sur un graphique appelé graphique de *l'espace des phases*. Utiliser `parametric` pour produire un graphique de l'espace des phases pour l'oscillateur harmonique.

16. On peut incorporer un amortissement dépendant de la vitesse à l'oscillateur harmonique de l'exercice précédent en ajoutant un terme d'amortissement à l'équation différentielle :

$$x''(t) = -\frac{k}{m}x(t) - b \cdot x'(t).$$

    Reprendre en intégralité l'exercice précédent pour le même oscillateur, mais cette fois avec le terme d'amortissement $b = 0.3$ ajouté.

17. * Les animations permettent d'illustrer facilement le sens de parcours d'une courbe paramétrique. Utiliser le code suivant pour créer une animation illustrant comment la figure de Lissajous de l'exemple 5.1.3 est tracée lorsque t varie de 0 à 2π.

    Les animations sont réalisées en dessinant une liste de « scènes graphiques » en séquence, chacune commençant par `gr2d`. Le code ci-dessous commence avec une liste vide, puis la boucle do utilise `append` pour ajouter une nouvelle scène pour chaque valeur de i. Chaque scène montre la courbe de Lissajous complète ainsi qu'un point rouge qui se déplace en 1000 étapes du point initial au point final de la courbe sur $[0, 2\pi]$. Enfin, `wxdraw` est utilisé pour générer le .gif animé. `delay` sert à controler la vitesse de l'animation (des valeurs plus petites indiquent un défilement plus rapide des scènes). Sur une machine Windows, le .gif animé devrait apparaître comme un fichier intitulé `maxout_n.gif` dans le dossier de l'utilisateur.

```
(%i1) x(t):=cos(7*t)$
      y(t):=sin(3*t)$
(%i2) scene:[]$

    for i:0 thru 1000 do
    (scene: append(scene,[gr2d(
     grid=true,
     xaxis=true,
     yaxis=true,
     title="Une courbe de Lissajous.",
     color=black,
     nticks=600,
     dimensions=[600,600],
```

```
      parametric(cos(7*t),sin(3*t),t,0,2*%pi),
     color=red,
     point_type=7,
      points([[x(i*2*%pi/1000),y(i*2*%pi/1000)]])
    )]));

(%i4) draw(
      delay=5,
      file_name="lissajous_anim",
      terminal='animated_gif,
      scene
      )$
```

NDT : Les nouvelles versions de wxMaxima ont introduit la commande `with_slider_draw` qui permet de réaliser directement une animation dans la feuille de travail sans avoir l'obligation de générer le fichier gif sur le pc et de le consulter après. Voici par exemple un exemple d'une telle commande, qui réalise la simulation précédente dans wxMaxima :

```
with_slider_draw(
  i, makelist(k, k, 0, 1000,4),
  grid=true,
  xaxis=true,
  yaxis=true,
  title="Une courbe de Lissajous.",
  nticks=150,
  dimensions=[600,600],
  color=black,
  parametric(cos(7*t),sin(3*t),t,0,2*%pi),
  color=red,
  point_type=filled_circle,
  points([[x(i*2*%pi/1000),y(i*2*%pi/1000)]])
)$
```

18. * Une courbe de Lissajous est donnée par les équations paramétriques $x(t) = \cos(t + \phi)$, $y(t) = \sin(2t)$. Créer une animation montrant comment l'angle de phase ϕ affecte la courbe. Chaque scène doit montrer la courbe résultant d'un angle de phase différent, et l'animation doit parcourir les angles de phase de 0 à 2π en 100 incréments.
19. * Créer une animation montrant comment la rosace d'équation polaire $r(\theta) = 3\cos(4\theta)$ est tracée sur $[0, 2\pi]$. Chaque scène de l'animation doit montrer la courbe sur $[0,\ i \cdot 2\pi/1000]$, afin que l'animation trace la courbe du début à la fin.

Chapitre 6

Suites infinies et séries

Commandes essentielles de ce chapitre

```
makelist      float         ratprint:false
limit         rectangle     subst
sum           integrate     taylor
for-do        diff          fourie
simpsum       find_root     foursin
```

6.1 Suites infinies et limites

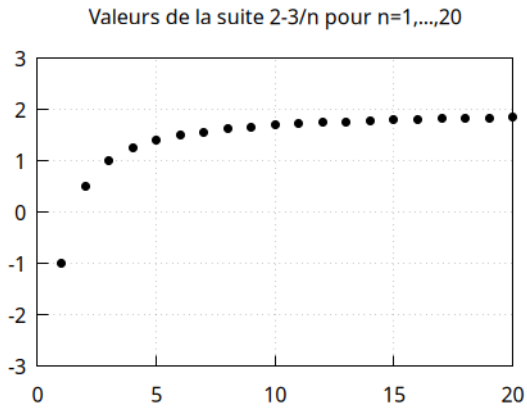
Une **suite infinie** est une liste de termes suivant généralement une règle. Nous utilisons la notation $\{a_n\}$ pour désigner la suite $\{a_1, a_2, a_3, \dots\}$, et nous étudions seulement le cas où a_n est donné explicitement en fonction de n. Nous pouvons voir la suite comme la restriction d'une fonction à un domaine qui ne contient que des nombres naturels : $a_n = a(n)$ pour $n = 1, 2, 3, \dots$.

De façon intuitive, nous disons que $\lim_{n\to\infty} a_n = L$ si les termes a_n deviennent arbitrairement proches de L lorsque n devient arbitrairement grand.

Exemple 6.1.1. Tracez les 20 premiers termes de la suite donnée par $a_n = 2 - \frac{3}{n}$. Par lecture graphique, émettez une hypothèse sur la limite de la suite, puis utilisez `limit` pour vérifier votre réponse.

Nous commençons par définir la fonction $a(n)$, puis nous générons une liste de points correspondant à $n = 1, 2, \dots, 20$:

```
(%i1) a(n):=2-3/n$
      POINTS:makelist([n,a(n)],n,1,20)$
(%i3) wxdraw2d(
      grid=true,
      xrange=[0,20],
      yrange=[-3,3],
      title="Valeurs de la suite {2-3/n} pour n=1,...,20",
      color=black,
       point_type=7,
      points(POINTS)
      );
```

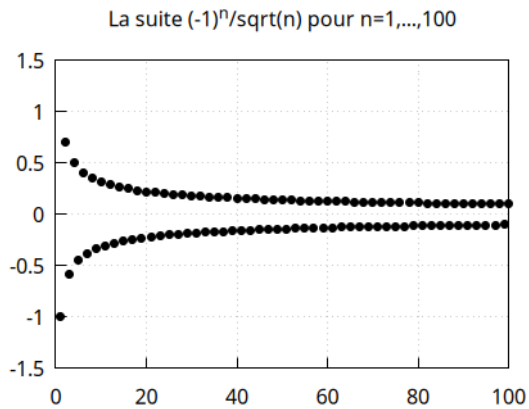


D'après le graphique, il semble que la limite de la suite soit 2. Nous calculons $\lim_{n\to\infty} a(n)$ pour vérifier notre hypothèse :

```
(%i4) limit(a(n),n,inf);
(%o4) 2
```

Exemple 6.1.2. Tracez les 100 premiers termes de la *suite alternée* donnée par $a_n = \frac{(-1)^n}{\sqrt{n}}$. A partir du graphique, quelle hypothèse peut-on émettre sur la limite de la suite ? Vérifiez votre hypothèse en utilisant `limit` :

```
(%i5) a(n):=(-1)^n/sqrt(n)$
(%i6) POINTS:makelist([n,a(n)],n,1,100)$
(%i7) wxdraw2d(
      grid=true,
      xrange=[0,100],
      yrange=[-1.5,1.5],
      title="La suite {(-1)^n/sqrt(n)} pour n=1,...,100",
      color=black,
      point_type=7,
       points(POINTS)
      );
```



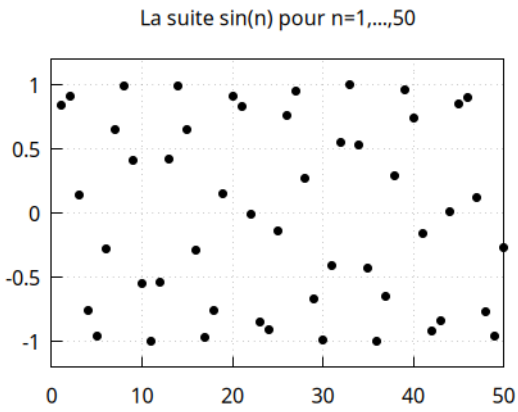
Il semble que la limite de la suite soit zéro. Nous vérifions la limite avec wxMaxima :

```
(%i8) limit(a(n),n,inf);
(%o8) 0
```

Exemple 6.1.3. Tracez les 50 premiers termes de la suite définie par $a_n = \sin(n)$. A partir du graphique, quelle hypothèse peut-on émettre sur la limite de la suite ? Vérifiez votre hypothèse en utilisant `limit` :

```
(%i9) a(n):=sin(n)$
      POINTS:makelist([n,a(n)],n,1,50)$
(%i11) wxdraw2d(
       grid=true,
       xrange=[0,50],
       yrange=[-1.2,1.2],
       title="La suite {sin(n)} pour n=1,...,50",
       color=black,
```

```
point_type=7,
 points(POINTS)
);
```



On voit que les termes ne se rapprochent d'aucune valeur particulière lorsque n devient grand, donc nous en déduisons l'hypothèse que la limite n'existe pas. wxMaxima le confirme :

```
(%i12) limit(a(n),n,inf);
(%o12) ind
```

6.2 Séries et sommes infinies

Lorsque nous additionnons les termes d'une suite infinie $\{a_n\}$, nous obtenons la **série infinie** $\sum_{n=1}^{\infty} a_n$. Pour calculer la somme d'une série, nous définissons une suite de **sommes partielles** : $\{S_n\}$, où S_n est la somme des n premiers termes de la série. Si $\lim_{n\to\infty} S_n = L$, alors nous disons que la série infinie converge vers L, ou nous disons simplement « la somme de la série est L ».

Exemple 6.2.1. Utilisez une boucle do pour afficher les 20 premières sommes partielles de la série $\sum_{n=1}^{\infty} \frac{1}{3^n}$. Les sommes partielles se rapprochent-elles d'une valeur finie ? Vérifiez votre réponse en utilisant `sum`.

La $n^{\text{ième}}$ somme partielle additionne simplement les n premiers termes de la suite : $S_1 = a_1$, $S_2 = a_1 + a_2$, $S_3 = a_1 + a_2 + a_3$, et ainsi de suite. Nous utilisons `sum` à l'intérieur de notre boucle pour calculer chaque somme partielle :

```
(%i1) a(n):=1/3^n$
(%i2) (print("n...... somme partielle"),
        print("1 ......",a(1)),
        for i:2 thru 20 do
        (S:float(sum(a(n),n,1,i)),
         print(i,".......",S))
       );

 n...... somme partielle
 1 ......1/3
 2.......0.44444444444444
 3.......0.48148148148148
 4.......0.49382716049383
 5.......0.49794238683128
 6.......0.49931412894376
 7.......0.49977137631459
 8.......0.49992379210486
 9.......0.49997459736829
10.......0.4999915324561
11.......0.49999717748537
12.......0.49999905916179
13.......0.49999968638726
14.......0.49999989546242
15.......0.49999996515414
16.......0.49999998838471
17.......0.49999999612824
18.......0.49999999870941
19.......0.4999999995698
20.......0.4999999998566
```

Il semble que la somme converge vers 0.5. Nous utilisons `sum` et `simpsum` pour vérifier :

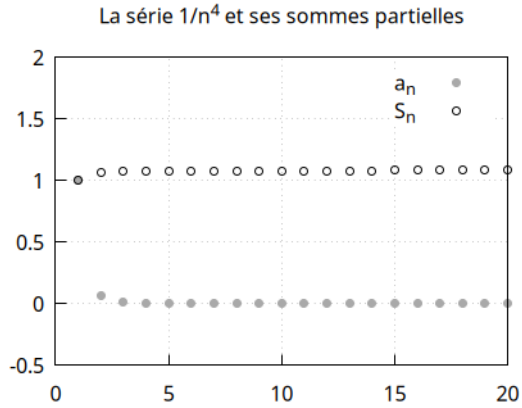
```
(%i3) sum(a(n),n,1,inf),simpsum;
(%o3) 1/2
```

Exemple 6.2.2. Calculez $\sum_{n=1}^{\infty} \frac{1}{n^4}$ en utilisant `sum` et `simpsum`. Utilisez `makelist` pour tracer la suite $\{a_n\}$ ainsi que la suite des sommes partielles $\{S_n\}$.

```
(%i4) a(n):=1/n^4$
(%i5) sum(a(n),n,1,inf),simpsum;

(%o5)   π^4/90

(%i6) float(%);
(%o6) 1.082323233711138
```

Nous voyons que la somme converge vers le résultat étonnant $\frac{\pi^4}{90} \approx 1.1$. Maintenant nous représentons $\{a_n\}$ et $\{S_n\}$:

```
(%i7) Apoints:makelist([i,a(i)],i,1,20)$
      Spoints:makelist([i,float(sum(a(n),n,1,i))],i,1,20)$
(%i9) wxdraw2d(
      grid=true,
      xrange=[0,20],
      yrange=[-.5,2],
      title="La série {1/n^4} et ses sommes partielles",
      color=dark_grey,
      point_type=7,
      key="{a_n}",
       points(Apoints),
      color=black,
      point_type=6,
      key="{S_n}",
       points(Spoints)
      );
```



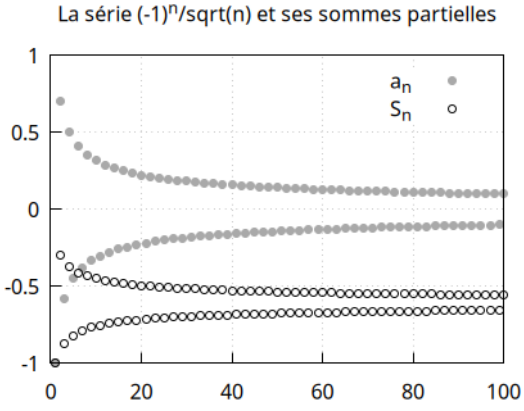
Nous voyons que $a_1 = S_1 = 1$, puis que les a_n s'approchent rapidement de zéro, donc les S_n n'augmentent que légèrement lorsque n croît.

Exemple 6.2.3. Essayez de calculer $\sum_{n=1}^{\infty} \frac{(-1)^n}{\sqrt{n}}$ en utilisant `sum` et `simpsum`. Que constatez vous ? Utilisez `makelist` pour tracer les termes de la suite $\{a_n\}$ ainsi que ceux de la suite des sommes partielles $\{S_n\}$. La somme semble-t-elle converger ? Utilisez `sum` pour obtenir une approximation de la somme de la série.

```
(%i10) a(n):=(-1)^n/sqrt(n)$
       sum(a(n),n,1,inf),simpsum;
(%o10) sum((-1)^n/sqrt(n),n,1,inf)
```

wxMaxima réécrit simplement la somme, indiquant ainsi qu'il ne sait pas calculer la somme exacte de cette série . Nous étudions la somme en traçant `{`a_n`}` et `{`S_n`}` pour $n = 1, 2, \ldots 100$:

```
(%i11) Apoints:makelist([i,a(i)],i,1,100)$
       Spoints:makelist([i,float(sum(a(n),n,1,i))],i,1,100)$
(%i13) wxdraw2d(
       grid=true,
       xrange=[0,100],
       yrange=[-1,1],
       title="La série {(-1)^n/sqrt(n)} et ses sommes partielles",
       color=dark_grey,
       point_type=7,
       key="{a_n}",
        points(Apoints),
       color=black,
       point_type=6,
       key="{S_n}",
        points(Spoints)
       );
```



Il semble que la somme converge vers une valeur légèrement inférieure à -0.5. Nous utilisons `sum` pour calculer $\sum_{n=1}^{10000} \frac{(-1)^n}{\sqrt{n}}$ afin de trouver une approximation de la limite. Notez que lorsque $n \approx 10000$, nous nous attendons à ce que la valeur approchée obtenue soit d'une précision de l'ordre du centième, puisque $\frac{1}{\sqrt{n}} \approx 0.01$ (obtenir un résultat plus précis se réalisera donc en utilisant un n plus grand). Nous redéfinissons $a(n)$ en utilisant `float` pour faciliter le calcul de l'approximation par wxMaxima :

```
(%i14) a(n):=float((-1)^n/sqrt(n))$
(%i15) sum(a(n),n,1,10000);
(%o15) -0.59989876842163
```

6.3 Tests classiques de convergence

A l'aide d'un système de calcul formel, nous pouvons toujours chercher à calculer les valeurs des sommes partielles pour de grandes valeurs de n afin de déterminer si une série converge ou non : nous additionnons simplement un très grand nombre de termes jusqu'à ce que nous soyons persudadés que les sommes partielles se stabilisent vers une limite finie. Cependant, les tests de convergence classiques « à la main » conservent une grande utilité théorique, notamment pour apporter la preuve de nos hypothèses. Dans cette section, nous utilisons wxMaxima pour illustrer les tests classiques à l'aide de graphiques, pour réaliser des manipulations algébriques et enfin pour effectuer la vérification finale de notre travail.

6.3.1 Le test de l'intégrale

Considérons la série infinie $\sum_{n=1}^{\infty} a_n$, où les termes peuvent être générés en évaluant une fonction continue, positive et décroissante $f(x)$ aux valeurs correspondant aux nombres naturels : $a_n = f(n)$. Nous pouvons tester la convergence de la série en étudiant l'intégrale impropre associée $\int_1^{\infty} f(x)\,\mathrm{d}x$ (la borne inférieure ne doit pas nécessairement être 1). La convergence de l'intégrale implique la convergence de la série, et la divergence de l'intégrale implique la divergence de la série.

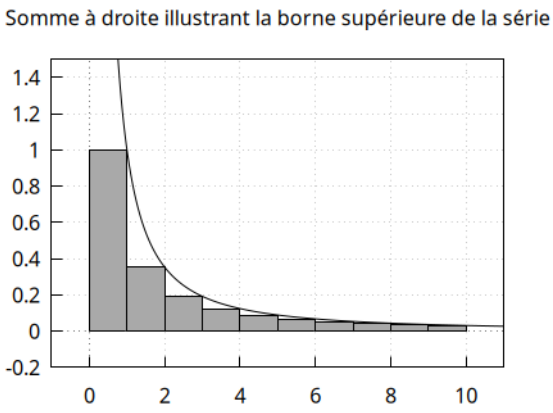
Nous illustrons de manière géométrique ce test de l'intégrale à l'aide d'un exemple.

Exemple 6.3.1. Pour la série $\sum_{n=1}^{\infty} \frac{1}{n^{3/2}}$, tracez la fonction associée $f(x)$ ainsi qu'une somme de Riemann à droite représentant la série pour $n = 1, 2, \ldots, 10$. Montrez que $\int_1^{\infty} f(x)\,\mathrm{d}x$ converge, et utilisez la valeur de l'intégrale pour établir une borne supérieure de la somme de la série. Enfin, utilisez wxMaxima pour approximer la somme en utilisant les 100 000 premiers termes..

Nous voyons que les termes de la série peuvent être générés en évaluant la fonction continue, positive et décroissante $f(x) = \frac{1}{x^{3/2}}$ en $n = 1, 2, \ldots$. Pour tracer la somme de Riemann à droite, nous utilisons `makelist` pour générer des rectangles de largeur 1 avec des hauteurs $f(1)$, $f(2)$,...,$f(10)$. Rappelez-vous que `rectangle` attend une paire de sommets situés aux coins opposés d'un rectangle.

```
(%i1) f(x):=1/x^(3/2)$
      RECTANGLES:makelist(rectangle([i-1,0],[i,f(i)]),i,1,10)$

      wxdraw2d(
      grid=true,
      xaxis=true,
      yaxis=true,
      xrange=[-1,11],
      yrange=[-0.2,1.5],
      title="Somme à droite illustrant la borne supérieure de la série",
      border=true,
      color=black,
      fill_color=dark_grey,
```

```
 RECTANGLES,
 explicit(f(x),x,0,11)
);
```



Les aires des rectangles sont $1 \cdot f(1), 1 \cdot f(2), \cdots = a_1 + a_2 + \ldots$, donc la somme de la série est simplement la somme des aires des rectangles. Puisque les rectangles sont bornés supérieurement par $f(x)$, l'aire délimitée sous $f(x)$ est plus grande que la somme de la série. $f(x)$ diverge en $x = 0$, donc nous commençons à $x = 1$ et nous pouvons affirmer que $\sum_{n=2}^{\infty} a_n < \int_1^{\infty} f(x)\,\mathrm{d}x$. En ajoutant a_1 aux deux membres de l'inégalité, nous obtenons notre borne supérieure pour la somme de la série : $\sum_{n=1}^{\infty} a_n < a_1 + \int_1^{\infty} f(x)\,\mathrm{d}x$. Nous utilisons wxMaxima pour calculer cette borne supérieure :

```
(%i4) f(1)+integrate(f(x),x,1,inf);
(%o4) 3
```

Nous voyons que la série converge vers une valeur inférieure à 3. Nous le vérifions en additionnant les 100 000 premiers termes :

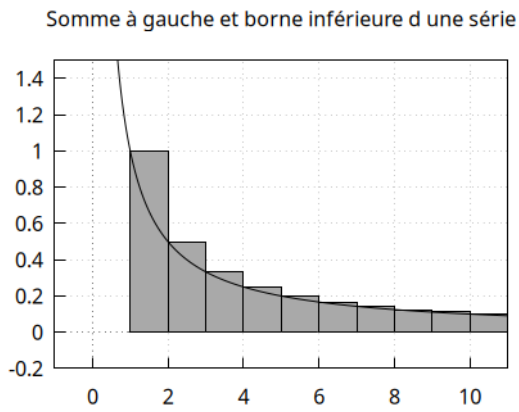
```
(%i5) a(n):=float(1/n^(3/2))$
      sum(a(n),n,1,100000);
(%o5) 2.606050809176471
```

Dans les exercices, nous explorons une méthode permettant d'estimer l'incertitude d'une telle approximation.

Exemple 6.3.2. Pour la série $\sum_{n=1}^{\infty} \frac{1}{x}$, tracer la fonction associée $f(x)$ et représenter une somme de Riemann à gauche représentant la série pour $n = 1, \ldots, 10$. Montrer que $\int_1^{\infty} f(x)\,\mathrm{d}x$ constitue une borne inférieure de la somme de la série. Enfin, utiliser l'intégrale impropre pour montrer que la série infinie diverge.

Les termes de la série sont générés en utilisant la fonction continue, positive et décroissante $f(x) = \frac{1}{x}$. Nous traçons $f(x)$ ainsi que les dix premiers termes de la somme à gauche :

```
(%i6) f(x):=1/x$
      RECTANGLES:(makelist(rectangle([i,0],[i+1,f(i)]),i,1,10))$

      wxdraw2d(
      grid=true,
      xaxis=true,
      yaxis=true,
      xrange=[-1,11],
      yrange=[-0.2,1.5],
      title="Somme à gauche et borne inférieure d une série",
      color=black,
      border=true,
      fill_color=dark_grey,
       RECTANGLES,
       explicit(f(x),x,0,11)
      );
```



Une fois encore, les aires des rectangles sont égales aux termes $a_1, a_2, \ldots$ de la série, et nous observons que $\sum_{n=1}^{\infty} a_n > \int_1^{\infty} f(x)\,\mathrm{d}x$; en d'autres termes, l'intégrale impropre constitue une borne inférieure de la somme de la série. Enfin, nous calculons l'intégrale avec wxMaxima :

```
(%i9) integrate(f(x),x,1,inf);

defint: integral is divergent.
 -- an error. To debug this try: debugmode(true);
```

L'intégrale diverge vers ∞, donc la série diverge également.

6.3.2 Tests de comparaison

Nous pouvons tester la convergence d'une série en la comparant à une autre série dont la convergence est déjà connue. En supposant que chaque série ne comporte que des termes positifs :

1. Si $a_n < b_n$ pour tout $n > N$ et $\sum_{n=1}^{\infty} b_n$ converge, alors $\sum_{n=1}^{\infty} a_n$ converge aussi.
2. Si $a_n > b_n$ pour tout $n > N$ et $\sum_{n=1}^{\infty} b_n$ diverge, alors $\sum_{n=1}^{\infty} a_n$ diverge aussi.
3. Si $\lim_{n\to\infty} \frac{a_n}{b_n}$ est fini, alors $\sum_{n=1}^{\infty} a_n$ et $\sum_{n=1}^{\infty} b_n$ convergent simultanément ou divergent simultanément.

Exemple 6.3.3. Utilisez le premier critère de comparaison pour montrer que $\sum_{n=1}^{\infty} \frac{1}{3^n+5}$ converge. Utilisez une boucle do pour lister les sommes partielles pour $n = 1, 2, \ldots, 30$. Quelle est la somme approximative de la série ?

Nous comparons à la série $\sum_{n=1}^{\infty} \frac{1}{3^n}$ dont nous avons déjà montré qu'elle converge vers 0.5.

Puisque $\frac{1}{3^n+5} < \frac{1}{3^n}$ pour tout n, $\sum_{n=1}^{\infty} \frac{1}{3^n+5}$ converge (vers une valeur inférieure à 0.5).

Nous calculons la suite des sommes partielles à l'aide d'une boucle do :

```
(%i10) a(n):=float(1/(3^n+5))$
       (print ("n........S_n"),
        for k:1 thru 30 do
        (S:sum(a(n),n,1,k),
         print(k,".......",S))
       );

 n........S_n
 1.......0.125
 2.......0.19642857142857
 3.......0.22767857142857
 4.......0.23930647840532
 5.......0.24333873646983
 6.......0.24470113429
 7.......0.24515733866956
 8.......0.24530963839542
 9.......0.24536043075625
10.......0.24537736441019
11.......0.24538300928013
12.......0.24538489093885
13.......0.24538551816236
14.......0.2453857272373
15.......0.24538579692899
16.......0.24538582015956
17.......0.24538582790309
18.......0.24538583048426
19.......0.24538583134466
```

```
20.......0.24538583163145
21.......0.24538583172705
22.......0.24538583175892
23.......0.24538583176954
24.......0.24538583177308
25.......0.24538583177426
26.......0.24538583177465
27.......0.24538583177479
28.......0.24538583177483
29.......0.24538583177484
30.......0.24538583177485
(%o11) done
```

La série converge rapidement vers environ 0.245.

Exemple 6.3.4. Testez la convergence de $\sum_{n=1}^{\infty} \frac{3n+2}{\sqrt{n^4-5}}$ en utilisant le troisième critère de comparaison.

Lorsque n devient grand, les puissances de n "dominent" toutes les combinaisons linéaires de termes : $a_n \to \frac{3n}{\sqrt{n^4}} = \frac{3}{n}$, ce qui nous conduit à comparer à la série de terme général $b_n = \frac{1}{n}$ (nous avons déjà montré que la série correspondante diverge). Nous prenons la limite dans wxMaxima :

```
(%i12) a(n):=(3*n+2)/sqrt(n^4-5)$
       b(n):=1/n$
       limit((a(n)/b(n)),n,inf);
(%o12) 3
```

Puisque la limite donne un nombre fini, nous concluons que les deux séries divergent.

6.3.3 Séries alternées et convergence absolue

Une **série alternée** est une série dont les termes alternent entre valeurs positives et négatives. Une série alternée $\sum_{n=1}^{\infty}(-1)^{n+1}a_n$ ou $\sum_{n=1}^{\infty}(-1)^{n}a_n$ (où $a_n > 0$) converge si $\lim_{n\to\infty} a_n = 0$ et $a_{n+1} < a_n$ pour tout $n > N$; c'est-à-dire que la valeur absolue des termes décroît et tend vers zéro lorsque n devient grand.

Notez que la comparaison entre a_{n+1} et a_n n'est pas toujours simple : il peut être plus facile d'utiliser la fonction continue $f(x)$ où $f(n) = a_n$, puis de calculer la dérivée première pour établir que la fonction est décroissante pour tout $n > N$.

Une série $\sum_{n=1}^{\infty} b_n$ **converge absolument** si $\sum_{n=1}^{\infty} |b_n|$ est convergente. Une série qui converge absolument converge aussi forcément au sens ordinaire. Certaines séries convergentes ne sont pas absolument convergentes – on les appelle **semi-convergentes**.

Exemple 6.3.5. Montrer que la série alternée

$$\sum_{n=1}^{\infty} \frac{(-1)^n \cdot (2n^2 + 5\sqrt{n})}{n^3 + 2}$$

est semi-convergente.

Nous commençons par tester la convergence absolue. Nous définissons $f(x)$, où $f(n) = a_n$ donne la valeur absolue de chaque terme de la série :

```
(%i13) f(x):=(2*x^2+5*sqrt(x))/(x^3+2)$
```

Quand x devient très grand, on peut dire que cette fonction se rapproche de $\frac{2x^2}{x^3} = \frac{2}{x}$, donc nous effectuons une comparaison avec la série divergente donnée par $b_n = \frac{1}{n}$ en calculant la limite du rapport des deux termes :

```
(%i14) limit(f(n)/(1/n),n,inf);
(%o14) 2
```

Nous en concluons que la série des valeurs absolues diverge, et donc que cette série n'est pas absolument convergente. Pour tester la convergence ordinaire de cette série alternée, nous étudions la limite quand n tend vers $+\infty$:

```
(%i15) limit(f(x),x,inf);
(%o15) 0
```

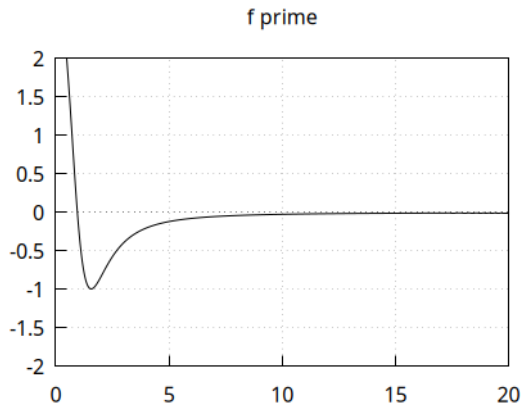
Il est trop difficile de comparer directement les termes a_{n+1} et a_n, donc nous étudions la dérivée première de $f(x)$ et montrons que $f'(x) < 0$ pour tout x suffisamment grand :

```
(%i16) diff(f(x),x)$
       f_prime(x):=''%;
```

(%o17) $\text{f\_prime}\,(x) := \frac{4\,x+\frac{5}{2\,\sqrt{x}}}{x^3+2} - \frac{3\,x^2\left(2\,x^2+5\,\sqrt{x}\right)}{\left(x^3+2\right)^2}$

Nous traçons $f'(x)$ pour avoir une idée de son comportement :

```
(%i18) wxdraw2d(
       grid=true,
       xaxis=true,
       yaxis=true,
       xrange=[0,20],
       yrange=[-2,2],
       title="f_prime",
       color=black,
        explicit(f_prime,x,0,20)
       );
```



$f'(x)$ semble être négative pour tout x supérieur à 1. Nous utilisons `find_root` pour déterminer précisément la racine évidente, puis nous vérifions (avec un indice de confiance élevé) qu'il n'existe aucune racine supplémentaire en essayant `find_root` sur $[5, 1000000]$:

```
(%i19) find_root(f_prime,x,0.5,1.5);
(%o19) 0.95344003838215

(%i20) find_root(f_prime,x,5,1000000);
find_root: function has same sign at endpoints: mequal(f(5.0),-0.11820541726029),
mequal(f(1000000.0),-2.0000000125*10^-12)
 -- an error. To debug this try: debugmode(true);
```

Ainsi, $f'(x)$ devient négative autour de $x = 0.95$ et *reste* négative ensuite. Puisque $f(x)$ est décroissante, nous concluons que $f(n+1) < f(n)$ pour tout $n > 0.95$, et nous en déduisons que la série alternée converge. La série est donc semi-convergente. Nous utilisons wxMaxima pour en obtenir une approximation :

```
(%i21) sum(float((-1)^x*f(x)),x,1,10000);
(%o22) -1.378757553897757
```

6.3.4 Les critères de d'Alembert et de Cauchy

Les critères de d'Alembert et de Cauchy permettent de déterminer si la série $\sum_{n=1}^{\infty} a_n$ converge absolument :

1. Si $\lim_{n\to\infty} \left|\frac{a_{n+1}}{a_n}\right| < 1$ ou $\lim_{n\to\infty} \sqrt[n]{|a_n|} < 1$, alors $\sum_{n=1}^{\infty} a_n$ converge absolument.
2. Si $\lim_{n\to\infty} \left|\frac{a_{n+1}}{a_n}\right| > 1$ ou $\lim_{n\to\infty} \sqrt[n]{|a_n|} > 1$, alors $\sum_{n=1}^{\infty} a_n$ diverge.
3. Quand $\lim_{n\to\infty} \left|\frac{a_{n+1}}{a_n}\right| = 1$ ou $\lim_{n\to\infty} \sqrt[n]{|a_n|} = 1$, les tests ne permettent pas de conclure.

Exemple 6.3.6. Appliquez à la fois le critère de d'Alembert et le critère de Cauchy pour montrer que $\sum_{n=1}^{\infty} \frac{2^n}{e^{0.8n-3}}$ converge absolument.

Nous n'avons pas à nous préoccuper des valeurs absolues puisque le numérateur et le dénominateur sont toujours positifs :

```
(%i23) ratprint:false$
(%i24) a(n):=2^n/%e^(0.8*n-3)$

(%i25) limit(a(n+1)/a(n),n,inf);
(%o25) 0.89865792823444

(%i26) float(limit((a(n))^(1/n),n,inf));
(%o26) 0.89865792823444
```

Dans chaque cas, la limite est égale à une constante inférieure à 1, donc la série converge absolument.

Exemple 6.3.7. Testez la convergence de la série $\sum_{n=1}^{\infty} \frac{n^{\ln 2n}}{n!}$. Si la série converge, calculez une approximation de sa valeur en additionnant les 1000 premiers termes.

```
(%i27) a(n):=n^(log(2*n))/n!$

(%i28) limit((a(n+1))/a(n),n,inf);
(%o28) 0

(%i29) limit((a(n))^(1/n),n,inf);
(%o29) 0
```

Dans chaque cas, nous obtenons comme limite une constante inférieure à 1, donc la série est absolument convergente. Nous terminons en calculant une approximation :

```
(%i30) sum(float(a(n)),n,1,1000);
(%o30) 4.746354216649364
```

6.4 Séries entières

6.4.1 Convergence et rayon de convergence

Une **série entière** est une série infinie de la forme $\sum_{n=0}^{\infty} a_n(x-c)^n$. On dit que la série est « centrée en c ». Pour tester la convergence d'une série entière, on utilise le critère de d'Alembert ou de Cauchy afin de déterminer les conditions sur les valeurs de x pour lesquelles la série converge. À de rares exceptions près, une série entière converge pour toutes les valeurs de x situées à une certaine distance, R, de c. R est appelé le **rayon de convergence** et $(c-R, c+R)$ est appelé l'**intervalle de convergence**. Les extrémités de l'intervalle de convergence doivent être testées séparément pour l'étude de la convergence.

Exemple 6.4.1. Trouvez les valeurs de x pour lesquelles la série $\sum_{n=0}^{\infty} ax^n$ converge.

La série converge lorsque $\lim_{n\to\infty} \left|\frac{a_{n+1}}{a_n}\right| < 1$. Nous définissons les termes de la suite comme $A(n)$ et calculons la limite dans wxMaxima :

```
(%i1) A(n):=a*x^n$
      limit(abs(A(n+1)/A(n)),n,inf);

(%o2) |x|
```

La série est convergente lorsque $|x| < 1$, divergente lorsque $|x| > 1$, et le critère ne permet pas de conclure lorsque $|x| = 1$. Nous étudions les cas correspondant à ces deux valeurs $x = \pm 1$: lorsque $x = 1$, on obtient la série $\sum_{n=0}^{\infty} a = \infty$, et lorsque $x = -1$, on obtient la série $\sum_{n=0}^{\infty} a(-1)^n$ qui est également divergente (les sommes partielles oscillent entre 0 et a, donc la limite n'existe pas). Ainsi, la série converge uniquement sur $]-1, 1[$.

Par ailleurs, cette série est connue sous le nom de *série géométrique*, et wxMaxima en connaît sa somme (nous devons indiquer que x se situe dans l'intervalle de convergence) :

```
(%i3) sum(A(n),n,0,inf),simpsum;

"Is  "abs(x)-1"  positive, negative, or zero?"negative;

(%o3) a/(1-x)
```

Exemple 6.4.2. Trouvez les valeurs de x pour lesquelles la série $\sum_{n=0}^{\infty} \frac{x^n}{n!}$ converge.

Nous définissons le terme de rang n et appliquons à nouveau le critère de d'Alembert :

```
(%i4) A(n):=x^n/n!$

(%i5) limit(abs(A(n+1)/A(n)),n,inf);
(%o5) 0
```

Le résultat est inférieur à 1 quelle que soit la valeur de x. Le rayon de convergence est infini, et l'intervalle de convergence est $]-\infty, \infty[$.

Exemple 6.4.3. Trouvez les valeurs de x pour lesquelles la série $\sum_{n=0}^{\infty} \frac{(x-2)^n}{n(n+1)}$ converge.

```
(%i6) A(n):=(x-2)^n/(n*(n+1))$
(%i7) limit(abs(A(n+1)/A(n)),n,inf);
(%o7) |x-2|
```

$|x-2| < 1 \implies -1 < x-2 < 1 \implies 1 < x < 3$, donc la série converge sur $]1,3[$. En testant l'extrémité de l'intervall $x = 1$, on obtient la série alternée $\sum_{n=0}^{\infty} \frac{(-1)^n}{n(n+1)}$ qui converge car $\lim_{n\to\infty} \frac{1}{n(n+1)} = 0$ et $\frac{1}{(n+1)(n+2)} < \frac{1}{n(n+1)}$. En testant l'extrémité $x = 3$, on obtient $\sum_{n=0}^{\infty} \frac{1}{n(n+1)}$ qui converge par comparaison avec $\sum_{n=0}^{\infty} \frac{1}{n^2}$ car $\frac{1}{n(n+1)} < \frac{1}{n^2}$. Ainsi, l'intervalle de convergence est $[1,3]$.

6.4.2 Séries de Taylor

Une **série de Taylor** est une représentation en série entière d'une fonction dérivable, f : $f(x) = a_0 + a_1(x-c) + a_2(x-c)^2 + \ldots$. Les coefficients a_n sont déterminés en évaluant $f(x)$ et ses dérivées en $x = c$ (comme illustré dans l'exemple suivant). Lorsque x est très proche de c, les puissances élevées de $(x-c)$ sont petites, et nous pouvons tronquer la série de Taylor pour obtenir une approximation polynomiale de $f(x)$ précise au voisinage de c. Dans les applications, il est très courant de tronquer une série de Taylor centrée en $c = 0$ (une **série de Maclaurin**) même s'il faut déplacer l'origine vers un point où l'étude de la série est intéressante.

Exemple 6.4.4. Trouvez les cinq premiers coefficients de Taylor en définissant $f(x) = a_0 + a_1(x-c) + a_2(x-c)^2 + \ldots$ et en calculant $f(c)$, $f'(c)$, $f''(c)$, et ainsi de suite.

Avant de faire appel à wxMaxima, notons qu'évaluer $f(x)$ en $x = c$ élimine tous les termes sauf le premier (a_0). Cette propriété se répète à chaque dérivée successive : seul le premier terme est dépourvu d'un facteur $(x-c)$, il sera donc le seul à apparaitre à chaque étape. Nous obtenons les cinq premiers coefficients en utilisant une série finie pour $f(x)$ dans wxMaxima et en utilisant une boucle « do » pour automatiser la sortie :

```
(%i8) f(x):=a_0+a_1*(x-c)+a_2*(x-c)^2+a_3*(x-c)^3+a_4*(x-c)^4+a_5*(x-c)^5$

(%i9) (for n:0 thru 4 do
  (N:subst([x=c],diff(f(x),x,n)),
   print("f^(",n,")(x)=",diff(f(x),x,n)),
   print("f^(",n,")(c)=",N))
);

f^(0)(x)=a_5*(x-c)^5+a_4*(x-c)^4+a_3*(x-c)^3+a_2*(x-c)^2+a_1*(x-c)+a_0
f^(0)(c)=a_0

f^(1)(x)=5*a_5*(x-c)^4+4*a_4*(x-c)^3+3*a_3*(x-c)^2+2*a_2*(x-c)+a_1
f^(1)(c)=a_1
```

```
f^(2)(x)=20*a_5*(x-c)^3+12*a_4*(x-c)^2+6*a_3*(x-c)+2*a_2
f^(2)(c)=2*a_2

f^(3)(x)=60*a_5*(x-c)^2+24*a_4*(x-c)+6*a_3
f^(3)(c)=6*a_3

f^(4)(x)=120*a_5*(x-c)+24*a_4
f^(4)(c)=24*a_4
(%o9) done
```

Nous voyons que $a_0 = f(c)$, $a_1 = f'(c)$, $a_2 = \frac{f''(c)}{2}$, $a_3 = \frac{f'''(c)}{6} = \frac{f'''(c)}{3!}$, et ainsi de suite. Ce n'est pas une coïncidence si les dénominateurs sont des factorielles : elles découlent du calcul des puissances successives de $(x - c)$ restant devant chaque terme à mesure que l'on prend des dérivées d'ordre de plus en plus élevé. Nous concluons que la formule générale des coefficients de Taylor est $a_n = \frac{f^{(n)}(c)}{n!}$.
Rappelons que $0! = 1$, donc la formule reste valable pour le cas $n = 0$.

Exemple 6.4.5. Calculez les cinq premiers coefficients non nuls de la série de Maclaurin pour les fonctions $f(x) = \sin x$ et $g(x) = \cos x$. Utilisez les résultats observés pour les coefficients pour trouver l'expression générale pour les séries infinies égales à ces deux fonctions. Enfin, montrez que la dérivée de la série de $\sin x$ donne la série de $\cos x$.

Nous utilisons la formule $a_n = \frac{f^{(n)}(0)}{n!}$ avec `makelist` pour générer une liste des premiers termes pour chaque fonction :

```
(%i10) f(x):=sin(x)$
       g(x):=cos(x)$
       SINETERMS:makelist(x^n*subst([x=0],diff(f(x),x,n))/n!,n,0,9);
       COSINETERMS:makelist(x^n*subst([x=0],diff(g(x),x,n))/n!,n,0,9);
(%o12) [0,x,0,-x^3/6,0,x^5/120,0,-x^7/5040,0,x^9/362880]
(%o13) [1,0,-x^2/2,0,x^4/24,0,-x^6/720,0,x^8/40320,0]
```

En utilisant la notation factorielle, nous pouvons écrire :
$\sin x = x - \frac{x^3}{3!} + \frac{x^5}{5!} - \frac{x^7}{7!} + \frac{x^9}{9!} - \ldots$ et $\cos x = 1 - \frac{x^2}{2!} + \frac{x^4}{4!} - \frac{x^6}{6!} + \frac{x^8}{8!} - \ldots$.
En convertissant dans une notation correspondante aux séries :

$$\sin x = \sum_{n=0}^{\infty} \frac{(-1)^n x^{(2n+1)}}{(2n+1)!} \quad \text{and} \quad \cos x = \sum_{n=0}^{\infty} \frac{(-1)^n x^{2n}}{(2n)!}$$

Enfin, nous dérivons la première série pour montrer que nous obtenons bien la seconde :

```
(%i14) A(n):=(-1)^n*x^(2*n+1)/(2*n+1)!$

(%i15) sum(A(n),n,0,inf);
```

$$\text{(\%o15)} \quad \sum_{n=0}^{\infty} \frac{(-1)^n \, x^{2\,n+1}}{(2\,n+1)!}$$

```
(%i16) diff(%,x);
```

(%o16) $$\sum_{n=0}^{\infty} \frac{(2\,n+1)\,(-1)^n\,x^{2\,n}}{(2\,n+1)!}$$

```
(%i17) factcomb(%);
```

(%o17) $$\sum_{n=0}^{\infty} \frac{(-1)^n\,x^{2\,n}}{(2\,n)!}$$

Nous retrouvons la série infinie de $\cos x$.

Exemple 6.4.6. Utilisez la fonction intégrée de wxMaxima `taylor` pour trouver les 5 premiers termes de la série de Taylor de $f(x) = \sqrt{x}\sin x$ centrée en $c = 2$. Tracez $f(x)$ ainsi que les approximations de Taylor linéaire, quadratique, cubique et quartique au voisinage de $c = 2$.

`taylor` nécessite quatre arguments : la fonction à développer en série de Taylor, la variable, le valeur du centre du développement (c), et la puissance maximale de x à conserver dans la série tronquée :

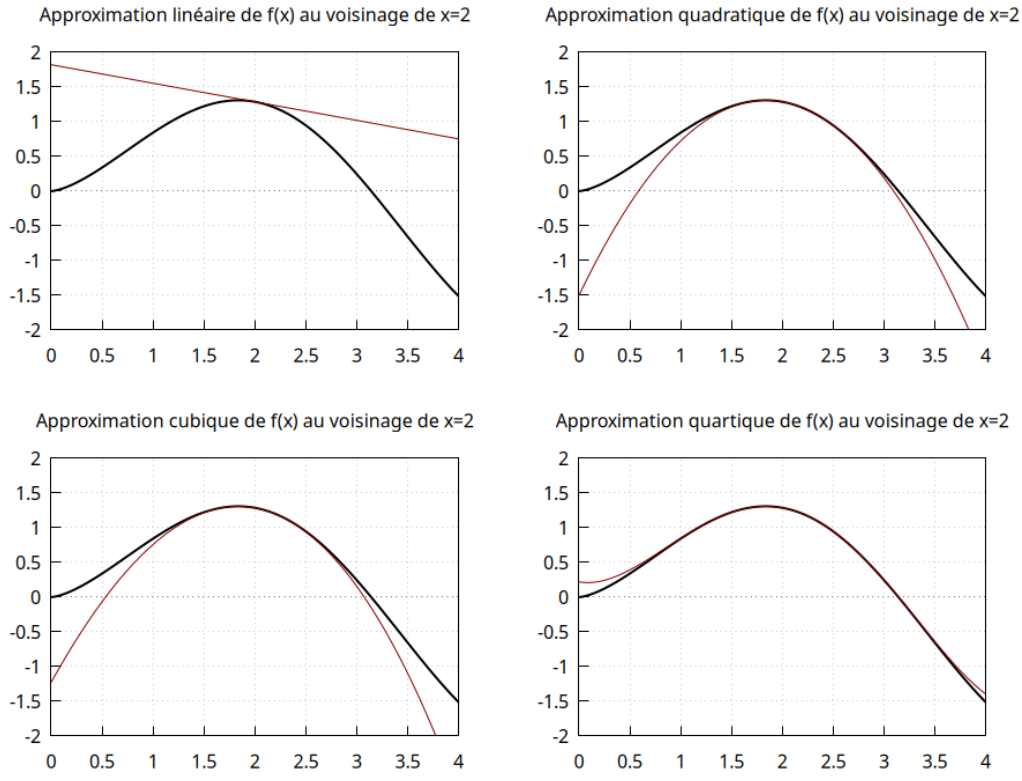
```
(%i18) f(x):=sqrt(x)*sin(x)$
       taylor(f(x),x,2,4);
```

(%o19)/T/ $\sin(2)\,\sqrt{2} + \frac{(\sin(2)+4\cos(2))\,\sqrt{2}\,(x-2)}{4} - \frac{(17\sin(2)-8\cos(2))\,\sqrt{2}\,(x-2)^2}{32} - \frac{(45\sin(2)+76\cos(2))\,\sqrt{2}\,(x-2)^3}{384} + \frac{(337\sin(2)-208\cos(2))\,\sqrt{2}\,(x-2)^4}{6144} + \ldots$

Nous définissons maintenant les approximations linéaire, quadratique, cubique et quartique (ou d'ordre 4 de manière plus courante).
Le code de tracé n'est présenté que pour l'approximation quartique :

```
(%i20) linear:taylor(f(x),x,2,1)$
       quadratic:taylor(f(x),x,2,2)$
       cubic:taylor(f(x),x,2,3)$
       quartic:taylor(f(x),x,2,4)$

(%i21) wxdraw2d(
       grid=true,
       xaxis=true,
       yaxis=true,
       xrange=[0,4],
       yrange=[-2,2],
       title="Approximation quartique de f(x) au voisinage de x=2",
       color=black,
       line_width=2,
        explicit(f(x),x,0,4),
       color=dark_red,
```

```
line_width=1,
 explicit(quartic,x,0,4)
);
```



Nous voyons l'approximation se préciser au fur et à mesure que l'on conserve des termes supplémentaires. L'approximation d'ordre 4 est très proche de $f(x)$ sur $[0.5, 3.5]$, et des valeurs plus grandes de n produiront des intervalles encore plus larges sur lesquels l'approximation fonctionne bien.

6.5 *Série de Fourier en sinus

En utilisant une série de Taylor, nous avons exprimé certaines fonctions $f(x)$ en termes de polynômes de degré infini $\{1, x, x^2, \dots\}$, où une formule faisant appel à la *dérivée* a été utilisée pour calculer le coefficient approprié pour chaque terme. Nous avons également approximé $f(x)$ en tronquant la série de Taylor après un nombre fini de termes.

Il est également possible de représenter $f(x)$ comme une série infinie de fonctions *sinus* définies pour un nombre entier de demi-périodes sur $[0, L]$: $f(x) = \sum_{n=1}^{\infty} a_n \sin\left(\frac{n\pi x}{L}\right)$. Cette représentation est appelée **série de Fourier en sinus**. Les fonctions de la série en sinus possèdent une flexibilité suffisante pour représenter n'importe quelle fonction $f(x)$ usuelle, et elles possèdent également les propriétés nécessaires pour utiliser le calcul par une *intégrale* afin de déterminer les coefficients.

Dans cette section, nous apprenons à calculer les coefficients de Fourier à l'aide de wxMaxima, puis nous traçons plusieurs approximations de Fourier afin de mieux comprendre comment la série en sinus converge vers une fonction $f(x)$.

Exemple 6.5.1. Utilisez wxMaxima pour calculer l'intégrale $\int_0^L \sin\left(\frac{n\pi x}{L}\right) \sin\left(\frac{m\pi x}{L}\right) \mathrm{d}x$ pour $n \neq m$ et $n = m$.

```
(%i1) f(x,n):=sin(n*%pi*x/L)$
      declare(n,integer)$
      declare(m,integer)$

(%i4) integrate(f(x,n)*f(x,m),x,0,L);
"Is  "L"  positive, negative, or zero?"positive;
(%o4) 0

(%i5) integrate(f(x,n)*f(x,n),x,0,L);
"Is  "L"  positive, negative, or zero?"positive;
(%o5) L/2
```

Nous voyons que $\int_0^L \sin\left(\frac{n\pi x}{L}\right) \sin\left(\frac{m\pi x}{L}\right) \mathrm{d}x$ vaut $\frac{L}{2}$ lorsque $n = m$, mais l'intégrale s'annule lorsque $n \neq m$ (il s'agit d'une propriété appelée *orthogonalité*).

Pour calculer les coefficients de Fourier, nous partons du développement en série $f(x) = \sum_{n=1}^{\infty} a_n \sin\left(\frac{n\pi x}{L}\right)$, multiplions les deux membres par $\sin\left(\frac{m\pi x}{L}\right)$ et intégrons de 0 à L :

$$\int_0^L f(x) \sin\left(\frac{m\pi x}{L}\right) \mathrm{d}x = \int_0^L \sin\left(\frac{m\pi x}{L}\right) \sum_{n=1}^{\infty} a_n \sin\left(\frac{n\pi x}{L}\right) \mathrm{d}x$$

Lorsque nous distribuons $\sin\left(\frac{m\pi x}{L}\right)$ dans la série, nous obtenons une infinité de termes, mais *toutes les intégrales résultantes s'annulent sauf le terme obtenu pour $n = m$*. Le membre de droite se simplifie donc en une seule intégrale que nous avons déjà rencontrée :

$$\int_0^L f(x)\sin\left(\frac{m\pi x}{L}\right)\,\mathrm{d}x = a_m\int_0^L \sin\left(\frac{m\pi x}{L}\right)\sin\left(\frac{m\pi x}{L}\right)\,\mathrm{d}x = a_m\frac{L}{2}$$

En fin de compte, nous résolvons en a_m :

$$a_m = \frac{2}{L}\int_0^L f(x)\sin\left(\frac{m\pi x}{L}\right)\,\mathrm{d}x$$

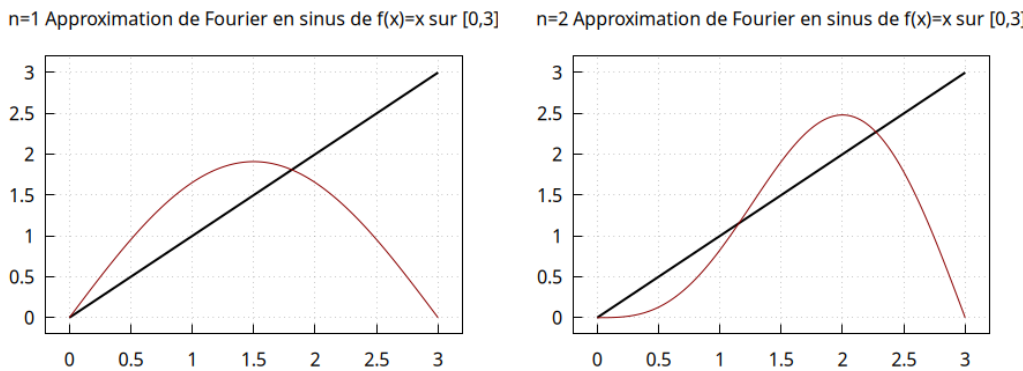
Exemple 6.5.2. Calculez les six premiers coefficients de Fourier pour $f(x) = x$ sur $[0, 3]$. Produisez six graphiques montrant les approximations de Fourier de $f(x)$ obtenues en augmentant le nombre de termes conservés dans la série en sinus.

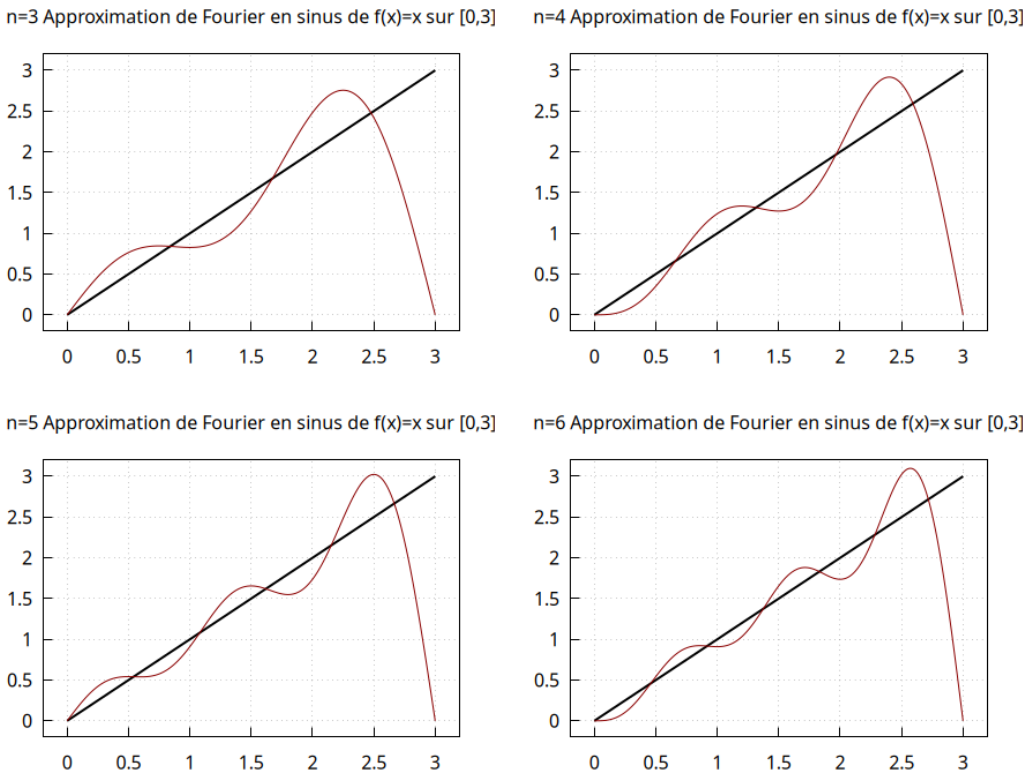
```
(%i6) a(m):=(2/3)*integrate(x*sin(m*%pi*x/3),x,0,3)$
(%i7) LIST:makelist(a(m),m,1,6);
```

(%o7) $[\frac{6}{\pi}, -\frac{3}{\pi}, \frac{2}{\pi}, -\frac{3}{2\,\pi}, \frac{6}{5\,\pi}, -\frac{1}{\pi}]$

Nous écrivons le code de sorte qu'il suffit d'entrer n pour obtenir l'approximation comprenant n termes, puis nous réalisons les graphiques (le code est présenté pour le cas $n = 3$) :

```
(%i8) APPROX(n):=sum(LIST[k]*sin(k*%pi*x/3),k,1,n)$
(%i9) wxdraw2d(
      grid=true,
      xrange=[-0.2,3.2],
      yrange=[-0.2,3.2],
      title="n=3 Approximation de Fourier en sinus de f(x)=x sur [0,3]",
      color=black,
      line_width=2,
       explicit(x,x,0,3),
      color=dark_red,
      line_width=1,
       explicit(APPROX(3),x,0,3)
      );
```





Nous voyons que l'approximation s'améliore au fur et à mesure que nous conservons davantage de termes de la série en sinus.

Exemple 6.5.3. Utilisez la fonction intégrée de wxMaxima `foursin` pour calculer les coefficients de Fourier en sinus de $f(x) = x^2$ sur $[0, 5]$. Tracez $f(x)$ ainsi que l'approximation pour $n = 50$.

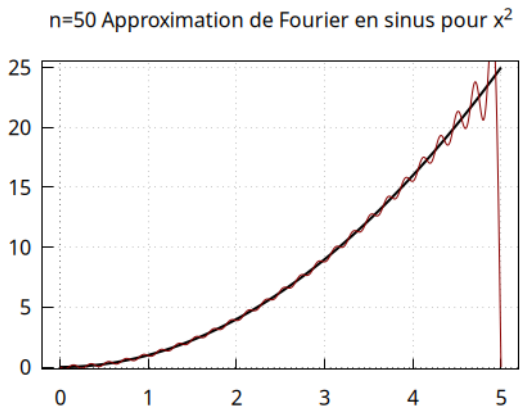
Nous devons charger le paquetage `fourie` avant d'appliquer `foursin` à notre fonction. wxMaxima calcule une formule explicite pour les coefficients en fonction de n, puis nous devons définir une fonction basée sur cette formule et enfin construire la somme partielle pour $n = 50$ avant de réaliser le tracé :

```
(%i1) load(fourie)$
(%i2) foursin(x^2,x,5);


(%t2)b[n]=(2*((250*sin(%pi*n))/(%pi^2*n^2)-(125*cos(%pi*n))/(%pi*n)+
     (250*cos(%pi*n))/(%pi^3*n^3)-250/(%pi^3*n^3)))/5
(%o2) [%t2]

(%i3) rhs(%t2)$
```

```
(%i4) a(n):=''%;
```

(%o4) $$\mathrm{a}\left(n\right) := \frac{2\left(\frac{250\sin(\pi\,n)}{\pi^2\,n^2} - \frac{125\cos(\pi\,n)}{\pi\,n} + \frac{250\cos(\pi\,n)}{\pi^3\,n^3} - \frac{250}{\pi^3\,n^3}\right)}{5}$$

```
(%i5) APPROX50:sum(a(n)*sin(n*%pi*x/5),n,1,50)$

       wxdraw2d(
       grid=true,
       xaxis=true,
       yaxis=true,
       dimensions=[600,600],
       xrange=[-0.2,5.2],
       yrange=[-0.2,25.5],
       title="n=50 Approximation de Fourier en sinus pour x^2",
       color=black,
       line_width=2,
        explicit(x^2,x,0,5),
       color=dark_red,
       line_width=1,
        explicit(APPROX50,x,0,5)
       );
```



Nous voyons que la série en sinus converge vers la fonction $f(x) = x^2$, sauf lorsque l'on se rapproche de la valeur 5. En effet, chaque fonction sinus de la série s'annule aux extrémités de l'intervalle, donc la somme s'annule également aux extrémités. Notre approximation chute donc brusquement à 0 au voisinage de 5 après avoir bien suivi x^2 sur la majeure partie de l'intervalle $[0, 5]$.

6.6 Exercices du Chapitre 6

1. Représentez les premiers termes de la suite de terme général $a_n = f(n)$ où $f(x) = \frac{\sqrt{x^2-4}}{x+3}$ et calculez $\lim_{n\to\infty} a_n$.
2. Tracez les termes de la suite donnée par $a_n = \frac{3n^4-2n}{n^6}$ ainsi que la suite des sommes partielles $S_n = \sum_{k=1}^{n} a_k$. Calculez $\lim_{n\to\infty} S_n$ en utilisant simplement `sum` avec $n = \infty$.
3. On peut définir un « théorème des gendarmes » pour la limite d'une suite : si $a_n \leq b_n \leq c_n$ pour tout $x \geq N$, et $\lim_{n\to\infty} a_n = \lim_{n\to\infty} c_n = L$, alors $\lim_{n\to\infty} b_n = L$. Pour illustrer le théorème des gendarmes, tracez les termes de $\{-\frac{1}{n}\}$ et $\{+\frac{1}{n}\}$ (en noir) comme bornes inférieure et supérieure des termes de $\{\frac{\sin n}{n}\}$ (en rouge). Montrez que la limite de $\{\frac{\sin n}{n}\}$ coïncide avec les limites des bornes inférieure et supérieure.
4. Si le terme de rang n d'une suite ne tend pas vers zéro lorsque n devient très grand, alors la suite des sommes partielles diverge nécessairement. Montrez que la suite $\{1 + \frac{1}{n^2}\}$ converge vers une valeur finie. Tracez 20 termes de $\{1 + \frac{1}{n^2}\}$ ainsi que la suite des sommes partielles afin d'illustrer pourquoi la suite des sommes partielles diverge.
5. Utilisez l'idée de l'exercice précédent pour montrer que $\sum_{n=1}^{\infty} \frac{\sqrt{n}}{\ln n}$ diverge.
6. Le critère de l'intégrale peut montrer la convergence à condition d'utiliser une fonction continue qui est *à partir d'une certaine valeur* positive et décroissante. Utilisez les graphes de $f(x) = \frac{\ln x}{x^2}$ et de $f'(x)$ avec `find_root` pour établir la valeur de x à partir de laquelle $f(x)$ est décroissante et positive. Trouvez et calculez une intégrale permettant de montrer que $\sum_{n=1}^{\infty} \frac{\ln x}{x^2}$ converge. Déterminer une borne supérieure pour la série à l'aide d'une intégrale, et illustrez votre réponse en traçant la somme de Riemann appropriée.
7. Nous pouvons utiliser l'idée du critère de l'intégrale pour déterminer des bornes d'erreur sur une somme partielle. Supposons que nous additionnions les n premiers termes d'une série pour obtenir une somme partielle S_n, et que les termes de la série soient positifs et décroissants pour $x \geq n$, et engendrés par une fonction continue $f(x)$. La somme partielle est une sous-estimation de la somme totale, et l'on appelle reste la quantité $R_n = S - S_n$, où S est la somme exacte. Si l'on considère les termes restants de la série $a_{n+1}, a_{n+2}, \ldots$ comme une somme de Riemann à droite, alors l'intégrale $\int_n^{\infty} f(x)\ \mathrm{d}x$ nous donne une borne supérieure sur le reste. Si l'on considère les termes restants comme une somme de Riemann à gauche, alors l'intégrale $\int_{n+1}^{\infty} f(x)\ \mathrm{d}x$ nous donne une borne inférieure sur le reste.

   Calculez la somme partielle $\sum_{n=1}^{10000} \frac{n+3}{n^3+n}$, calculez les bornes supérieure et inférieure sur le reste, et écrivez un intervalle pour les valeurs possibles de S.
8. Utilisez le critère de l'intégrale pour montrer que $\sum_{2}^{\infty} \frac{1}{\sqrt{x}\ln x}$ diverge.
9. Utilisez le critère de comparaison avec calcul de la limite pour déterminer si la série $\sum_{n=1}^{\infty} \frac{\sqrt{3n^2-1}}{n^3+5n}$ converge ou non.
10. Utilisez le critère de d'Alembert pour montrer que $\sum_{n=1}^{\infty} \frac{n}{2^n}$ converge. Utilisez ensuite le critère de comparaison avec passage à la limite pour montrer que $\sum_{n=1}^{\infty} \frac{n}{2^n-1}$ converge.

11. Tentez d'utiliser le critère de l'intégrale pour montrer que $\sum_{n=1}^{\infty} \frac{\ln n}{2^n - 1}$ converge. Que se passe-t-il ? Utilisez une boucle `do` pour afficher les sommes partielles pour $n = 100, 200, \ldots, 2000$. La série semble-t-elle converger ? Utilisez les résultats de l'exercice précédent pour montrer que la série converge (vous devrez établir que $\ln n < n$).
12. Montrer que la série $\sum_{n=1}^{\infty} \frac{(-1)^n n^{10}}{e^n}$ converge absolument.
13. Montrer que la série $\sum_{n=1}^{\infty} (-1)^n \left(\frac{\pi}{2} - \tan^{-1} n \right)$ est semi-convergente.
14. Utilisez `taylor` pour calculer les dix premiers termes de la série de Taylor de $\ln x$ centrée en $x = 1$. Trouvez une formule explicite pour le $n^{\text{ième}}$ terme du développement, et calculez l'intervalle de convergence. Montrez que la série converge en $x = 2$, puis exprimez $\ln 2$ comme une série infinie. Quel est le nom usuel de cette série ?
15. Calculez les dix premiers termes de la série de Maclaurin pour $f(x) = (1 + x)^n$ en utilisant `taylor`. Quelle est l'erreur commise par l'approximation linéaire lorsque $x = 0.001$?
16. Utilisez `taylor` pour écrire au moins les dix premiers termes de la série de Maclaurin pour e^{ix} (le symbole de l'unité imaginaire est `%i` dans wxMaxima). Séparez (mentalement) les termes réels et imaginaires pour obtenir deux séries infinies distinctes. Reconnaissez-vous ces deux séries ? Enfin, écrivez e^{ix} comme une combinaison linéaire de $\cos x$ et $\sin x$.
17. Utilisez `taylor` pour écrire la série de Maclaurin pour $\cosh x$. Pouvez-vous trouver un argument de la fonction cosinus produisant cette même série ? Vérifiez votre réponse avec wxMaxima.
18. Exprimez $f(x) = \frac{1}{1+x^2}$ sous forme d'une série de Maclaurin (en gardant au moins cinq termes non nuls) à l'aide de la commande `taylor`. Trouvez ensuite une équation avec $f(x)$ à gauche et sa série de puissances à droite. Appliquez `integrate` à cette équation, puis évaluez le résultat en $x = 1$ pour obtenir une série infinie égale à π. Écrivez la formule explicite du $n^{\text{ième}}$ terme de cette série. Enfin, utilisez `sum` pour trouver une approximation de π avec 100 termes. Quelle est l'erreur commise par cette approximation ?
19. Pour un pendule simple, une analyse des moments de force conduit à l'équation différentielle du second ordre $\frac{\mathrm{d}^2\theta}{\mathrm{d}t^2} = -\frac{g}{L} \sin\theta$, où g est l'accélération de la gravité, L est la longueur du pendule et θ est l'angle du pendule mesuré par rapport à la « verticale ». Cette équation différentielle est non linéaire et n'admet pas de solution sous forme algébrique. Cependant, on peut faire une approximation pour les petits angles en utilisant la série de Taylor centrée en $c = 0$: si θ est « petit », on peut tronquer la série de $\sin\theta$ après le terme de degré 1 pour linéariser l'équation différentielle.
    (a) Quelle est l'approximation linéaire de $\sin\theta$ lorsque θ est proche de zéro ?
    (b) Tracez $\sin\theta$ et l'approximation linéaire dans la même fenêtre sur l'intervalle $[-.5, .5]$. Rappelez-vous que l'angle est mesuré ici en radians, ce qui correspond à un déplacement d'environ 30° par rapport à la position d'équilibre.
    (c) Calculez l'intervalle des valeurs de θ pour lesquelles l'erreur commise par l'approximation linéaire est inférieure à 5%.
    (d) Utilisez `ode2` pour calculer la solution générale de l'équation différentielle linéarisée. Quelle est la période de la solution ?

20. *Une *onde stationnaire* résulte d'une vibration sur une corde pour laquelle un nombre entier de demi-longueurs d'onde s'inscrit précisément sur la longueur de la corde. Pour une corde donnée, il n'existe qu'un ensemble discret de longueurs d'onde (et de fréquences) produisant des ondes stationnaires. Pour une longueur L, une amplitude A et une vitesse d'onde v, l'équation d'une onde stationnaire est $y(x,t) = A\sin\left(\frac{n\pi x}{L}\right)\cdot\cos\left(\frac{n\pi v}{L}t\right)$ pour $n = 1, 2, 3, \ldots$.

    Le code suivant utilise $L = 1$, $v = 1$, $n = 3$ et $A = 1$ pour créer une animation montrant une onde stationnaire avec des scènes calculées toutes les 0.02 s pendant 10 s :

```
TIMEDEP(x,t,n):=sin(n*%pi*x)*cos(n*%pi*t)$
TIMEDEP3:TIMEDEP(x,t,3)$
load(draw)$
wxanimate_autoplay:true;
wxanimate_framerate:20;
with_slider_draw(
 i, makelist(0.05*i, i, 0, 200),
 title=concat("t=",0.05*i),
 grid=true,
 xaxis=true,
 yaxis=true,
 nticks=120,
 xrange=[0,1],
 yrange=[-1,1],
 color=black,
 line_width=2,
 explicit(subst(0.05*i,t,TIMEDEP3),x,0,1)
)$
```

    (a) Exécutez l'animation pour $n = 3$.
    (b) Créez une deuxième animation pour le cas $n = 5$.
    (c) Créez une animation combinant les ondes stationnaires $n = 3$ et $n = 5$, en donnant au cas $n = 5$ un tiers de l'amplitude du cas $n = 3$.

21. *Pour modéliser une corde de guitare pincée, on utilise une série de Fourier en sinus pour exprimer l'état initial comme une superposition de nombreuses ondes sinusoïdales qui s'annulent aux extrémités. L'avantage de cette approche est que l'évolution temporelle de chaque onde sinusoïdale est simple : chaque fonction sinus est simplement une onde stationnaire, et elle évolue selon la formule $y(x,t) = A\sin\left(\frac{n\pi x}{L}\right)\cdot\cos\left(\frac{n\pi v}{L}t\right)$.

    Supposons qu'une corde de guitare soit contrainte à l'intervalle $[0, 2]$ et qu'elle soit pincée selon une forme initiale donnée par la fonction définie par morceaux :

$$f(x) = \begin{cases} x & 0 \le x \le 0.5 \\ 1 - x & 0.5 \le x \le 1 \\ 0 & 1 \le x \le 2 \end{cases}$$

    (a) Calculez une approximation par une série de Fourier en sinus à 50 termes de cette fonction sur $[0, 2]$, puis tracez l'approximation avec $f(x)$ pour vérifier qu'elle fonctionne.

Remarque : `foursin` ne fonctionnera pas dans ce cas car $f(x)$ est définie par morceaux. Vous devrez calculer les coefficients en utilisant `integrate` et en divisant l'intervalle d'intégration.

(b) En supposant $v = 5$, ajoutez la dépendance temporelle correcte à chaque terme du développement de Fourier et créez une animation montrant comment la forme de la corde évolue pendant vingt secondes après le pincement initial. Utilisez des pas de temps de 0.02 seconde pour calculer les scènes de l'animation.

(c) Dans des ondes réalistes, les fréquences plus élevées s'atténuent plus rapidement que les fréquences plus basses. Utilisez un facteur d'amortissement $e^{-(0.1n)t}$ attaché à chaque $n^{\text{ième}}$ terme et relancez l'animation.

Si votre animation fonctionne correctement, les vibrations se stabiliseront après un certain temps vers le mode « fondamental » (dans le mode fondamental, le centre de la corde monte et descend simplement).

Appendice du Traducteur

La traduction de ce manuel a été finalisée en août 2026.

— Il est à noter que la traduction s'est voulue fidèle au manuel original. Dans ce cadre, certaines expressions mathématiques ou approches des concepts sont marqués par la culture anglo-saxone de l'enseignement des mathématiques, et ne correspondent pas forcément à l'enseignement délivré en France. Par exemple, l'auteur parle indifféremment de f, de $f(x)$, du graphe de $f(x)$, alors que nous sommes plus exigents en évitant de confondre une fonction et son expression. De même, la généralisation à partir de l'observation d'un phénomène mathématique est souvent plus fréquente que dans notre enseignement, ou l'on attend une preuve pour tout hypothèse émise.
— Certains codes Maxima ont été mis à jour pour fonctionner avec les dernières versions de Maxima.
— La correction des exercices est proposée sur le site https://maxima-french-doc.fr/
— N'hésitez pas à envoyer vos remarques, commentaires, ou bugs trouvés à mon adresse michel.gosse@free.fr