SympyCalc [fr]

Des outils pour utiliser les capacités de sympy de , avec le package pyluatex.

Version 0.1.1 -- 8 mai 2023

Cédric Pierquet
c pierquet -- at -- outlook . fr
https://github.com/cpierquet/SympyCalc

SymPy est un module Python qui permet de faire du calcul symbolique.
https://www.sympy.org/

- Compilation en LuaLaTeX, avec un accès –shell-escape, en *partenariat* avec pyluatex.
- Du calcul *exact* avec des racines, de l'exponentielle, du logarithme, des complexes…
- Du calcul symbolique avec du développement, de la factorisation.
- Des résolutions d'équations.
- Du calcul différentiel, intégral.
- Tout ce que sympy peut faire !

$$1+\frac{7}{6}=\frac{13}{6} \text{ et } \left(\mathrm{e}^5+1\right)\times\left(\mathrm{e}^5-1\right)=-1+\mathrm{e}^{10}$$

$$3+\mathrm{i}\sqrt{3}=2\sqrt{3}\mathrm{e}^{\frac{\mathrm{i}\pi}{6}}$$

$$\mathrm{I}=\int_0^4 (x+3)\,\mathrm{e}^{2x}\,\mathrm{d}x=-\frac{5}{4}+\frac{13\mathrm{e}^8}{4}$$

$$\lim_{x\to+\infty}\sqrt{\frac{2x^2+1}{x^2-5}}=\sqrt{2}$$

$$10\,\mathrm{e}^{4x+5}=2:\mathscr{S}=\left\{-\frac{5}{4}-\frac{\ln(5)}{4}\right\}$$

$$f(t)=(t+3)\,\mathrm{e}^{2t}\implies f'(t)=(2t+7)\,\mathrm{e}^{2t}$$

LaTeX

pdfLaTeX

LuaLaTeX

TikZ

TeXLive

MiKTeX

Table des matières

Première partie
Introduction

1 Le package SympyCalc

1.1 Introduction et conventions

Le package *propose* des outils pour utiliser les capacités du module sympy de python et les *formater* en LaTeX :

- calcul exact (dans la mesure du possible) ;
- calcul symbolique (dans la mesure du possible) :
  - factorisation, développement, simplification ;
  - dérivation, intégration, limites ;
  - résolution d'équations.

À noter que les calculs – en interne – sont :

- effectués par le module sympy ;
- formatés par la fonction `latex()` du module sympy ;
- sortis en LaTeX avec quelques ajustements.

Les ajustements (qu'on pourrait qualifier de *mini-patchs*) permettent :

- de forcer les résultats sous forme de fraction (grâce – en interne – au recours à `S.One`) ;
- d'utiliser la notation `ln` (sympy utilise `log` par défaut) ;
- d'utiliser l'écriture du `e` et du `i` en romain ;
- de forcer l'écriture du « + » avant « ∞ ».

Hormis le *forçage* du `ln`, et du $+\infty$, conventions sur `e` et `i` peuvent être *désactivées*.

Le package propose également de quoi utiliser (et formater) toute commande exprimée en syntaxe sympy.
Le module sympy est chargé par le package, avec les variables symboliques `x`, `y`, `z` et `t`.

L'utilisation de pyluatex (en tant que passerelle entre python et LaTeX) nécessite une compilation adaptée, à savoir en LuaLaTeX et en activant le mode –shell-escape. Le module sympy doit donc être installé pour que l'exécutable python qui sera paramétré le *trouve* !.

1.2 Chargement du package

Le package (qui ne charge *que* xstring) se charge de manière classique, une fois pyluatex lui-même correctement chargé.
Compte-tenu de la spécificité de pyluatex et du paramétrage du chemin python, c'est à l'utilisateur de le charger en adéquation avec son installation !
Les packages adaptés aux formules mathématiques sont également nécessaires, mais compte-tenu de leurs diversités, l'utilisateur choisira celui (ceux) qu'il jugera nécessaire(s).

```
%compilation en lualatex + shell-escape !!
\usepackage{mathtools}                          %package pour les maths
\usepackage[executable=python.exe]{pyluatex}    %à adapter !
\usepackage{SympyCalc}
```

2 Philosophie du package

2.1 Fonctionnement global

Le package propose des commandes génériques, qui permettent de *traiter* tout commande sympy, ainsi que des commandes spécifiques qui sont en fait des *raccourcis* de commandes usuelles en sympy.

Autant que peut se faire, les commandes sont assez explicites sur leur fonctionnement, et elles reposent sur le même principe :

- `\sympy...` : commande qui simplifie (par défaut), avec règles typographiques `[fr]` romain ;
- `\dsympy...` : commande qui simplifie (par défaut), avec règles typographiques `[fr]` romain, et affichage en `\displaystyle` ;
- `\sympy...*` : commande qui simplifie (par défaut), avec règles typographiques `[fr]` italique ;
- `\dsympy...*` : commande qui simplifie (par défaut), avec règles typographiques `[fr]` italique, et affichage en `\displaystyle`.

Dans certains cas la *simplification* ne propose pas une sortie *classique*, et il est possible d'empêcher la simplification grâce à l'argument optionnel `<NoSimplif>`.

La variable est par défaut `x`, mais peut-être modifiée grâce à l'argument `[variable]`.

2.2 Commandes disponibles

```
%commandes génériques
\sympycalc(*)<NoSimplif>{commande sympy}
\dsympycalc(*)<NoSimplif>{commande sympy}

%commandes spécifiques, sortie standard, détaillées plus loin
\sympyimage(*)<NoSimplif>{fonction sympy}[variable]{point}
\sympydev(*)<NoSimplif>{expr sympy}[variable]
\sympyfact(*)<NoSimplif>{expr sympy}
\sympyderiv(*)<NoSimplif>{expr sympy}[variable]
\sympynbderiv(*)<NoSimplif>{expr sympy}[variable]{point}[côté]
\sympyprim(*)<NoSimplif>{expr sympy}[variable]
\sympyintegr(*)<NoSimplif>{expr sympy}[variable]{borne inf}{borne sup}
\sympylim(*)<NoSimplif>{expr sympy}[variable]{point}[côté]
\sympyfexpo(*)<NoSimplif>{complexe sympy}
\sympyresol(*){equation sympy}[variable]
\sympyresolC(*){equation sympy}[variable]

%commandes spécifiques, sortie displaystyle, détaillées plus loin
\dsympydev(*)<NoSimplif>{expr sympy}[variable]
\dsympyimage(*)<NoSimplif>{fonction sympy}[variable]{point}
\dsympyfact(*)<NoSimplif>{expr sympy}
\dsympyderiv(*)<NoSimplif>{expr sympy}[variable]
\dsympynbderiv(*)<NoSimplif>{expr sympy}[variable]{point}[côté]
\dsympyprim(*)<NoSimplif>{expr sympy}[variable]
\dsympyintegr(*)<NoSimplif>{expr sympy}[variable]{borne inf}{borne sup}
\dsympylim(*)<NoSimplif>{expr sympy}[variable]{point}[côté]
\dsympyfexpo(*)<NoSimplif>{complexe sympy}
\dsympyresol(*){equation sympy}[variable]
\dsympyresolC(*){equation sympy}[variable]
```

Dans certains cas, la sortie obtenue n'est pas celle attendue, que ce soit avec ou sans `<Simplif>`, et donc il s'agira de saisir les commandes sympy *manuellement*.

L'« infini » est disponible en sympy, grâce à `+oo` ou `-oo`.
Un petit *guide* pour sympy est disponible à : https://hashdork.com/fr/sympy-library-guide/.

3 Limitations

Pour le moment c'est le package pyluatex qui est privilégié, mais des tests sont en cours pour une compatibilité éventuelle avec d'autres packages pouvant travailler avec python.

Le formatage des flottants n'est pas géré par `\num` de sinuitx, donc l'écriture des éventuelles valeurs approchées ne seront pas forcément cohérentes avec les autres formatages dans le document…

Le remplacement éventuel du `e` en romain risque de ne pas fonctionner quand le résultat contient du « e tout seul »…

Des espacements *non voulus* peuvent apparaître lors de la transcription en LaTeX, donc le résultat formaté peut être légèrement différent de ce que peut attendre l'utilisateur…

La transformation automatique du `oo` en `+oo` ne détecte pas si le calcul est effectué avec des complexes, donc prudence sur les limites notamment…

Deuxième partie

Les commandes

4 La commande générique

4.1 Présentation

La commande générique pour formater une sortie sympy est \sympycalc ou \dsympycalc.
Elle permet de *parser* du code sympy et de formater la sortie avec les règles typographiques usuelles.

```
%commandes génériques
\sympycalc(*)<NoSimplif>{commande sympy}
\dsympycalc(*)<NoSimplif>{commande sympy}
```

4.2 Arguments

Concernant son fonctionnement :

- la version ⟨*⟩ permet de ne pas formater le `e` et le `i` en romain ;
- le premier argument, optionnel et pouvant valoir <NoSimplif> permet d'*annuler* la simplification (attention au comportement de la *simplification* par sympy…)
- le second argument, obligatoire et entre `{...}` est quant à lui la commande, en langage sympy à passer en python.

4.3 Exemples

```
On a $1+\frac76 = \sympycalc{1+7/6}$, ou $1+\dfrac76 = \dsympycalc{1+7/6}$
```

On a $1+\frac76 = \sympycalc{1+7/6}$, ou $1+\dfrac76 = \dsympycalc{1+7/6}$

```
$1+\frac14+\frac17 + \sqrt{45} + \frac{2}{\sqrt{12}} =
\sympycalc{1+1/4+1/7+sqrt(45)+2/sqrt(12)}$

$1+\dfrac14+\dfrac17 + \sqrt{45} + \dfrac{2}{\sqrt{12}} =
\dsympycalc{1+1/4+1/7+sqrt(45)+2/sqrt(12)}$
```

$1+\frac14+\frac17 + \sqrt{45} + \frac{2}{\sqrt{12}} = \sympycalc{1+1/4+1/7+sqrt(45)+2/sqrt(12)}$

$1+\dfrac14+\dfrac17 + \sqrt{45} + \dfrac{2}{\sqrt{12}} = \dsympycalc{1+1/4+1/7+sqrt(45)+2/sqrt(12)}$

```
On a $\left(\e^5 +1 \right) \times \left(\e^5-1 \right) =
    \sympycalc{(exp(5)+1)*(exp(5)-1)}$
```

On a $\left(\e^5 +1 \right) \times \left(\e^5-1 \right) = \sympycalc{(exp(5)+1)*(exp(5)-1)}$

```
On a $\dfrac{x}{x+1} - \dfrac{x+2}{x-3} = \dsympycalc{radsimp(factor(x/(x+1) -
    (x+2)/(x-3)))}$
```

On a $\dfrac{x}{x+1} - \dfrac{x+2}{x-3} = \dsympycalc{radsimp(factor(x/(x+1) - (x+2)/(x-3)))}$

```
Les solutions de $\dfrac13x^2+9x+4=0$ sont
$x_1 = \dsympycalc{solve(1/3*x**2+9*x+4,x)[0]}$ et
$x_2 = \dsympycalc{solve(1/3*x**2+9*x+4,x)[1]}$
```

Les solutions de $\frac{1}{3}x^2+9x+4=0$ sont $x_1=-\frac{27}{2}-\frac{\sqrt{681}}{2}$ et $x_2=-\frac{27}{2}+\frac{\sqrt{681}}{2}$

5 Quelques commandes spécifiques

5.1 Images

New 0.1.1 La commande pour calculer une image via sympy est \sympyimage ou \dsympyimage.
Il vaut mieux vérifier si le résultat formaté est conforme aux attentes, et sinon utiliser la commande générique.

```
%commandes pour cacluler une image
\sympyimage(*)<NoSimplif>{fonction sympy}[variable]{point}
\dsympyimage(*)<NoSimplif>{fonction sympy}[variable]{point}
```

Concernant son fonctionnement :

- la version ⟨*⟩ permet de ne pas formater le `e` et le `i` en romain ;
- le premier argument, optionnel et pouvant valoir <NoSimplif> permet d'*annuler* la simplification (attention au comportement de la *simplification* par sympy…)
- le deuxième argument, obligatoire et entre `{...}` est quant à lui la fonction, en langage sympy à passer en python ;
- les derniers arguments, optionnel entre `[...]` et obligatoire et entre `{...}` correspondent à la variable et le point.

```
$f(x)=\dfrac{2+\e^x}{3+4\e^{2x}}$ avec $a=5$ :
    $f(5)=\dsympyimage{(2+exp(x))/(3+4*exp(2*x))}{5}$
```

$f(x)=\frac{2+\mathrm{e}^x}{3+4\mathrm{e}^{2x}}$ avec $a=5$: $f(5)=\frac{2+\mathrm{e}^5}{3+4\mathrm{e}^{10}}$

```
$g(t)=\sqrt{1+\ln(t)}$ avec $a=13$ : $g(13)=\sympyimage{sqrt(1+log(t))}[t]{13}$
```

$g(t)=\sqrt{1+\ln(t)}$ avec $a=13$: $g(13)=\sqrt{1+\ln(13)}$

```
$h(x)=2x^2+2,5x+7$ en $a=\sqrt{2}$ :
$h\left(\sqrt{2}\right)=\dsympyimage{2*x**2+5/2*x+7}{sqrt(2)}$
```

$h(x)=2x^2+2,5x+7$ en $a=\sqrt{2}$: $h\left(\sqrt{2}\right)=\frac{5\sqrt{2}}{2}+11$

5.2 Développement

La commande pour développer une sortie sympy est \sympydev ou \dsympydev.
Il vaut mieux vérifier si le résultat formaté est conforme aux attentes, et sinon utiliser la commande générique.

```
%commandes pour développer
\sympydev(*)<NoSimplif>{commande sympy}[variable]
\dsympydev(*)<NoSimplif>{commande sympy}[variable]
```

Concernant son fonctionnement :

— la version ⟨*⟩ permet de ne pas formater le `e` et le `i` en romain ;

— le premier argument, optionnel et pouvant valoir <NoSimplif> permet d'*annuler* la simplification (attention au comportement de la *simplification* par sympy…)

— le deuxième argument, obligatoire et entre `{...}` est quant à lui la commande, en langage sympy à passer en python ;

— le dernier argument, optionnel et entre `[...]` est la variable éventuelle.

```
$(1+\text{i})^5 = \sympydev{(1+I)**5}$          %typo [fr]

$(1+i)^5 = \sympydev*{(1+I)**5}$

$\dfrac{1}{(1+\text{i})^5} = \dsympydev{1/(1+I)**5}$
```

$(1+\text{i})^5 = -4-4\text{i}$
$(1+i)^5 = -4-4i$
$\dfrac{1}{(1+\text{i})^5} = -\dfrac{1}{8}+\dfrac{\text{i}}{8}$

```
On a $(1+2x)^4 = \sympydev{(1+2*x)**4}$

On a $\left(2t+\dfrac13\right)^3 = \dsympydev{(2*t+1/3)**3}[t]$
```

On a $(1+2x)^4 = 16x^4+32x^3+24x^2+8x+1$
On a $\left(2t+\dfrac{1}{3}\right)^3 = 8t^3+4t^2+\dfrac{2t}{3}+\dfrac{1}{27}$

```
$\sin^2(x)+\cos^2(x) = \dsympydev{sin(x)**2+cos(x)**2}$

$\left(\cos(x)+\sin(x)\right)^2 = \dsympydev{(sin(x)+cos(x))**2}$
```

$\sin^2(x)+\cos^2(x) = 1$
$(\cos(x)+\sin(x))^2 = \sin(2x)+1$

5.3 Factorisation

La commande pour factoriser une sortie sympy est `\sympyfact` ou `\dsympyfact`.
Il vaut mieux vérifier si le résultat formaté est conforme aux attentes, et sinon utiliser la commande générique.

```
%commandes pour factoriser
\sympyfact(*)<NoSimplif>{commande sympy}
\dsympyfact(*)<NoSimplif>{commande sympy}
```

Concernant son fonctionnement :

— la version ⟨*⟩ permet de ne pas formater le `e` et le `i` en romain ;

— le premier argument, optionnel et pouvant valoir <NoSimplif> permet d'*annuler* la simplification (attention au comportement de la *simplification* par sympy…)

— le dernier argument, obligatoire et entre `{...}` est quant à lui la commande, en langage sympy à passer en python.

```
On a $x^3-x^2+x-1 = \sympyfact{x**3-x**2+x-1}$

On a $\e^x-3\e^{2x} = \sympyfact{exp(x)-3*exp(2*x)}$   %typo [fr]

On a $e^t-3e^{2t} = \sympyfact*{exp(t)-3*exp(2*t)}$
```

On a $x^3-x^2+x-1 = (x-1)\left(x^2+1\right)$
On a $\mathrm{e}^x-3\mathrm{e}^{2x} = (1-3\mathrm{e}^x)\,\mathrm{e}^x$
On a $e^t-3e^{2t} = \left(1-3e^t\right)e^t$

5.4 Résolution d'équations

La commande pour résoudre (dans les réels) une équation, grâce à sympy est \sympyresol ou \dsympyresol.
La commande pour résoudre (dans les complexes) une équation, grâce à sympy est \sympyresolC ou \dsympyresolC.

Pour cette commande, la simplification en sympy n'est pas *désactivable*.

Pour *forcer* les résultats sous forme exact (fractions), il vaut mieux écrire les nombres sous forme fractionnaire.

Il vaut mieux vérifier si le résultat formaté est conforme aux attentes, et sinon utiliser la commande générique.

```
%commandes pour résoudre dans R
\sympyresol(*){equation sympy}[variable]
\dsympyresol(*){equation sympy}[variable]

%commandes pour résoudre dans C
\sympyresolC(*){equation sympy}[variable]
\dsympyresolC(*){equation sympy}[variable]
```

Concernant son fonctionnement :

- la version ⟨*⟩ permet de ne pas formater le `e` et le `i` en romain ;
- le premier argument, obligatoire et entre `{...}` est quant à lui l'équation, en langage sympy à passer en python (avec le signe =) ;
- le second argument, optionnel et entre `[...]` est la variable.

```
Les solutions de $x^2-3=\num{0.5}$ sont $\mathcal{S}=\dsympyresol{x**2-3=0.5}$

%mieux vaut passer par les écritures en fraction ;-)
Les solutions de $x^2-3=\num{0.5}$ sont $\mathcal{S}=\dsympyresol{x**2-3=1/2}$
```

Les solutions de $x^2-3=0{,}5$ sont $\mathcal{S} = \{-1{,}87082869338697\,;\,1{,}87082869338697\}$
Les solutions de $x^2-3=0{,}5$ sont $\mathcal{S} = \left\{-\frac{\sqrt{14}}{2}\,;\,\frac{\sqrt{14}}{2}\right\}$

```
Les solutions réelles de $x^3-2,5x+16x=40$ sont
    $\mathcal{S}=\dsympyresol{x**3-5/2*x**2+16*x = 40}$

Les solutions de $x^3-2,5x+16x=40$ sont $\mathcal{S}=\dsympyresolC{x**3-5/2*x**2+16*x
    = 40}$
```

Les solutions réelles de $x^3 - 2{,}5x + 16x = 40$ sont $\mathcal{S} = \left\{\frac{5}{2}\right\}$

Les solutions de $x^3 - 2{,}5x + 16x = 40$ sont $\mathcal{S} = \left\{\frac{5}{2}\,;\,-4\mathrm{i}\,;\,4\mathrm{i}\right\}$

```
Les solutions de $10\,\e^{4x+5}=2$ sont $\mathcal{S}=\dsympyresol{10*exp(4*x+5)=2}$
```

Les solutions de $10\,\mathrm{e}^{4x+5} = 2$ sont $\mathcal{S} = \left\{-\frac{5}{4} - \frac{\ln(5)}{4}\right\}$

5.5 Dérivation locale

New 0.1.1 La commande pour dériver localement, grâce à sympy est \sympynbderiv ou \dsympynbderiv.
Il vaut mieux vérifier si le résultat formaté est conforme aux attentes, et sinon utiliser la commande générique.

```
%commandes pour dériver localement
\sympynbderiv(*)<NoSimplif>{expression sympy}[variable]{point}[côté]
\dsympynbderiv(*)<NoSimplif>{expression sympy}[variable]{point}[côté]
```

Concernant son fonctionnement :

- la version ⟨*⟩ permet de ne pas formater le `e` et le `i` en romain ;
- le premier argument, optionnel et pouvant valoir <NoSimplif> permet d'*annuler* la simplification (attention au comportement de la *simplification* par sympy…)
- le deuxième argument, obligatoire et entre `{...}` est quant à lui l'expression en langage sympy à passer en python ;
- le troisième argument, optionnel et entre `[...]` est la variable ;
- les derniers arguments, obligatoire et entre `{...}` et optionnel entre `[...]` correspondent à la valeur (et la position `[g]` ou `[d]`) en laquelle on souhaite travailler.

```
Si $f(x)=(x+3)\,\e^{2x}$, alors on peut montrer que $f'(2) =
    \sympynbderiv{(x+3)*exp(2*x)}{2}$
```

Si $f(x) = (x+3)\,\mathrm{e}^{2x}$, alors on peut montrer que $f'(2) = 11\mathrm{e}^4$

```
Le nombre dérivé de la fonction $x \mapsto \sqrt{x}$ en $a=2$ vaut
$\lim\limits_{x \to 2} \frac{\sqrt{x}-\sqrt{2}}{x-2} = \sympynbderiv{sqrt(x)}{2}$.

Le nombre dérivé de la fonction $x \mapsto \sqrt{x}$ en $a=2$ vaut
$\lim\limits_{x \to 2} \dfrac{\sqrt{x}-\sqrt{2}}{x-2} = \dsympynbderiv{sqrt(x)}{2}$.
```

Le nombre dérivé de la fonction $x \mapsto \sqrt{x}$ en $a = 2$ vaut $\lim\limits_{x \to 2} \frac{\sqrt{x}-\sqrt{2}}{x-2} = \frac{\sqrt{2}}{4}$.

Le nombre dérivé de la fonction $x \mapsto \sqrt{x}$ en $a = 2$ vaut $\lim\limits_{x \to 2} \dfrac{\sqrt{x}-\sqrt{2}}{x-2} = \dfrac{\sqrt{2}}{4}$.

```
Le nombre dérivé de $\varphi(x) = 1+\dfrac{1}{(x+2)^2}$ en $a=-5$ vaut
$\lim\limits_{x \to -3} \frac{\varphi(x)-\varphi(-5)}{x-(-5)} =
    \sympynbderiv{1+1/(x+2)**2}{-5}$.

Le nombre dérivé de $\varphi(x) = 1+\dfrac{1}{(x+2)^2}$ en $a=-5$ vaut
$\lim\limits_{x \to -3} \dfrac{\varphi(x)-\varphi(-5)}{x-(-5)} =
    \dsympynbderiv{1+1/(x+2)**2}{-5}$.
```

Le nombre dérivé de $\varphi(x) = 1+\dfrac{1}{(x+2)^2}$ en $a=-5$ vaut $\lim\limits_{x \to -3} \frac{\varphi(x)-\varphi(-5)}{x-(-5)} = \frac{2}{27}$.

Le nombre dérivé de $\varphi(x) = 1+\dfrac{1}{(x+2)^2}$ en $a=-5$ vaut $\lim\limits_{x \to -3} \dfrac{\varphi(x)-\varphi(-5)}{x-(-5)} = \dfrac{2}{27}$.

5.6 Dérivation globale

La commande pour dériver, grâce à sympy est \sympyderiv ou \dsympyderiv.
Il vaut mieux vérifier si le résultat formaté est conforme aux attentes, et sinon utiliser la commande générique.

```
%commandes pour dériver
\sympyderiv(*)<NoSimplif>{expression sympy}[variable]
\dsympyderiv(*)<NoSimplif>{expression sympy}[variable]
```

Concernant son fonctionnement :

- la version ⟨*⟩ permet de ne pas formater le e et le i en romain ;
- le premier argument, optionnel et pouvant valoir <NoSimplif> permet d'*annuler* la simplification (attention au comportement de la *simplification* par sympy…)
- le deuxième argument, obligatoire et entre {...} est quant à lui l'expression en langage sympy à passer en python ;
- le troisième argument, optionnel et entre [...] est la variable.

```
Si $f(x)=(x+3)\,\e^{2x}$, alors on peut montrer que $f'(x) =
    \sympyderiv{(x+3)*exp(2*x)}$
```

Si $f(x)=(x+3)\,\e^{2x}$, alors on peut montrer que $f'(x) = (2x+7)\,\e^{2x}$

```
La dérivée de $g(t)=\ln\left(\dfrac{1+x}{1-x}\right)$ sur
$\left]-1;1\right[$ est $g'(t) = \dsympyderiv<NoSimplif>{ln((1+t)/(1-t))}[t]
= \dsympyderiv{ln((1+t)/(1-t))}[t]$
```

La dérivée de $g(t)=\ln\left(\dfrac{1+x}{1-x}\right)$ sur $\left]-1;1\right[$ est $g'(t) = \dsympyderiv<NoSimplif>{ln((1+t)/(1-t))}[t] = \dsympyderiv{ln((1+t)/(1-t))}[t]$

```
La dérivée de $f(t)=t^2-8\ln(t)$ est
$f'(t)= \dsympyderiv{t**2-8*log(t)}[t] = \dsympycalc<NoSimplif>{cancel(2*t-8/t)}$
```

La dérivée de $f(t)=t^2-8\ln(t)$ est $f'(t)= \dsympyderiv{t**2-8*log(t)}[t] = \dsympycalc<NoSimplif>{cancel(2*t-8/t)}$

5.7 Intégration

Les commandes pour déterminer une primitive, grâce à sympy sont \sympyprim ou \dsympyprim.

Les commandes pour calculer une intégrale, grâce à sympy sont \sympyintegr ou \dsympysympyintegr.

Il vaut mieux vérifier si le résultat formaté est conforme aux attentes, et sinon utiliser la commande générique.

```
%commandes pour primitiver
\sympyprim(*)<NoSimplif>{expr sympy}[variable]
\dsympyprim(*)<NoSimplif>{expr sympy}[variable]

%commandes pour intégrer
\sympyintegr(*)<NoSimplif>{expr sympy}[variable]{borne inf}{borne sup}
\dsympyintegr(*)<NoSimplif>{expr sympy}[variable]{borne inf}{borne sup}
```

Concernant son fonctionnement :

- la version ⟨*⟩ permet de ne pas formater le `e` et le `i` en romain ;
- le premier argument, optionnel et pouvant valoir <NoSimplif> permet d'*annuler* la simplification (attention au comportement de la *simplification* par sympy…)
- le deuxième argument, obligatoire et entre `{...}` est quant à lui l'expression en langage sympy à passer en python ;
- le troisième argument, optionnel et entre `[...]` est la variable ;
- les derniers arguments, obligatoires et entre `{...}` correspondent aux bornes d'intégration (pour l'intégrale !).

```
Une primitive de la fonction $f(x)=(1+x)^2$ est $F(x) = \dsympyprim{(1+x)**2}$

Une primitive de la fonction $f(x)=(1+x)^2$ est $F(x) =
    \dsympyprim<NoSimplif>{(1+x)**2}$

Une primitive de la fonction $f(x)=(1+t)^2$ est $F(t) =
    \dsympyprim<NoSimplif>{(1+t)**2}[t]$
```

Une primitive de la fonction $f(x) = (1+x)^2$ est $F(x) = x\left(\frac{x^2}{3} + x + 1\right)$

Une primitive de la fonction $f(x) = (1+x)^2$ est $F(x) = \frac{x^3}{3} + x^2 + x$

Une primitive de la fonction $f(x) = (1+t)^2$ est $F(t) = \frac{t^3}{3} + t^2 + t$

```
On a $\displaystyle\int_1^3 (1+x)^2\,\text{d}x = \dsympyintegr{(1+x)**2}{1}{3}$

$I=\displaystyle\int_0^{4} (t+3)\,\e^{2t}\,\text{d}t =
    \dsympyintegr{(t+3)*exp(2*t)}[t]{0}{4}$
```

On a $\displaystyle\int_1^3 (1+x)^2\,\text{d}x = \frac{56}{3}$

$I = \displaystyle\int_0^4 (t+3)\,\mathrm{e}^{2t}\,\text{d}t = -\frac{5}{4} + \frac{13\mathrm{e}^8}{4}$

5.8 Limites

Les commandes pour déterminer une limite, grâce à sympy sont \sympylim ou \dsympylim.

```
%commandes pour détemriner une limite
\sympylim(*)<NoSimplif>{expr sympy}[variable]{point}[côté]
\dsympylim(*)<NoSimplif>{expr sympy}[variable]{point}[côté]
```

Concernant son fonctionnement :

- la version ⟨*⟩ permet de ne pas formater le `e` et le `i` en romain ;
- le premier argument, optionnel et pouvant valoir `<NoSimplif>` permet d'*annuler* la simplification (attention au comportement de la *simplification* par sympy…)
- le deuxième argument, obligatoire et entre `{...}` est quant à lui l'expression en langage sympy à passer en python ;
- le troisième argument, optionnel et entre `[...]` est la variable ;
- les derniers arguments, obligatoire et entre `{...}` et optionnel entre `[...]` correspondent à la valeur (et la position `[g]` ou `[d]`) en laquelle on souhaite travailler.

```
On a $\lim\limits_{x \to +\infty} \dfrac{1+\e^x}{x^2+1} =
    \sympylim{(1+exp(x))/(x**2+1)}{+oo}$
```

On a $\lim\limits_{x \to +\infty} \dfrac{1+\e^x}{x^2+1} = \sympylim{(1+exp(x))/(x**2+1)}{+oo}$

```
On a $\lim\limits_{x \to 1^+} \dfrac{x+2}{x^2-1} = \sympylim{(x+2)/(x**2-1)}{1}[d]$

On a $\lim\limits_{x \to 1^-} \dfrac{x+2}{x^2-1} = \sympylim{(x+2)/(x**2-1)}{1}[g]$
```

On a $\lim\limits_{x \to 1^+} \dfrac{x+2}{x^2-1} = \sympylim{(x+2)/(x**2-1)}{1}[d]$

On a $\lim\limits_{x \to 1^-} \dfrac{x+2}{x^2-1} = \sympylim{(x+2)/(x**2-1)}{1}[g]$

```
On a $\lim\limits_{x \to +\infty} \sqrt{\dfrac{2x^2+1}{x^2-5}} =
    \sympylim{sqrt((2*x**2+1)/(x**2-5))}{+oo}$
```

On a $\lim\limits_{x \to +\infty} \sqrt{\dfrac{2x^2+1}{x^2-5}} = \sympylim{sqrt((2*x**2+1)/(x**2-5))}{+oo}$

5.9 Forme exponentielle d'un complexe

La commande pour déterminer une forme exponentielle, grâce à sympy est `\sympyfexpo` ou `\dsympyfexpo`.
Il vaut mieux vérifier si le résultat formaté est conforme aux attentes, et sinon utiliser la commande générique.

```
%commandes pour détemriner une forme exponentielle
\sympyfexpo(*)<NoSimplif>{complexe sympy}
\dsympyfexpo(*)<NoSimplif>{complexe sympy}
```

Concernant son fonctionnement :

- la version ⟨*⟩ permet de ne pas formater le `e` et le `i` en romain ;
- le premier argument, optionnel et pouvant valoir `<NoSimplif>` permet d'*annuler* la simplification (attention au comportement de la *simplification* par sympy…)
- le second argument, obligatoire et entre `{...}` est quant à lui le complexe en version sympy à passer en python.

```
On a $1+\i = \dsympyfexpo{1+I}$

On a $3+\i\sqrt{3} = \dsympyfexpo{3+I*sqrt(3)}$

On a $\dfrac{-\sqrt{2}}{1+\i} = \dsympyfexpo{(-sqrt(2))/(1+I)}$
```

On a $1+\mathrm{i} = \sqrt{2}\mathrm{e}^{\frac{\mathrm{i}\pi}{4}}$
On a $3+\mathrm{i}\sqrt{3} = 2\sqrt{3}\mathrm{e}^{\frac{\mathrm{i}\pi}{6}}$
On a $\dfrac{-\sqrt{2}}{1+\mathrm{i}} = \mathrm{e}^{\frac{3\mathrm{i}\pi}{4}}$

Troisième partie

Historique et évolutions

6 Évolutions

L'utilisateur du package pourra utiliser ses propres fonctions ou commandes sympy, qui pourront être déclarés grâce à l'environnement `pythonq`.

```
\begin{pythonq}
def tangente_exacte(expr,var,a) :
    fdea = expr.subs(var,a)
    fprimedea = limit((expr-fdea)/(var-a),var,a)
    tgte = expand(fprimedea*(x-a) + fdea)
    return tgte

\end{pythonq}
```

```
La tangente à la courbe de la fonction $f(x)=\e^{2x}+3$ en $a=1$ a pour équation :\\
$y = \sympycalc{tangente_exacte(exp(2*x)+3,x,1)}$.
```

La tangente à la courbe de la fonction $f(x) = \mathrm{e}^{2x} + 3$ en $a = 1$ a pour équation : $y = 2x\mathrm{e}^{2} - \mathrm{e}^{2} + 3$.

```
La tangente à la courbe de la fonction $g(t)=\ln(x+3)$ en $a=4$ a pour équation :\\
$y = \dsympycalc{tangente_exacte(log(t+3),t,4)}$.
```

La tangente à la courbe de la fonction $g(t) = \ln(x + 3)$ en $a = 4$ a pour équation : $y = \frac{x}{7} - \frac{4}{7} + \ln(7)$.

7 Historique

`v0.1.1` : Commandes pour les nombres dérivés + images.
`v0.1.0` : Version initiale.