A Tour Of Sage

Release 10.4

The Sage Development Team

23 jul. 2024

Sumário

Esta apresentação ao Sage segue de perto o "tour do Mathematica" que se encontra no começo do "manual do Mathematica".

CAPÍTULO 1

Sage como uma Calculadora

A linha de comando do Sage possui o prompt `sage:`; você não precisa digitar essa palavra. Se você usar o Sage Notebook, então você deve copiar todo o comando após o prompt `sage:` em uma célula, e pressionar shift-enter para calcular o resultado.

```
sage: 3 + 5
8
```

```
>>> from sage.all import *
>>> Integer(3) + Integer(5)
8
```

O acento circunflexo significa "elevar à potência".

```
sage: 57.1 ^ 100
4.60904368661396e175
```

```
>>> from sage.all import *
>>> RealNumber('57.1') ** Integer(100)
4.60904368661396e175
```

Pode-se calcular a inversa de uma matrix 2×2 com o Sage.

```
sage: matrix([[1,2], [3,4]])^(-1)
[  -2    1]
[ 3/2 -1/2]
```

```
>>> from sage.all import *
>>> matrix([[Integer(1),Integer(2)], [Integer(3),Integer(4)]])**(-Integer(1))
[  -2    1]
[ 3/2 -1/2]
```

A seguir, calculamos a integral de uma função simples.

```
sage: x = var('x')    # create a symbolic variable
sage: integrate(sqrt(x)*sqrt(1+x), x)
1/4*((x + 1)^(3/2)/x^(3/2) + sqrt(x + 1)/sqrt(x))/((x + 1)^2/x^2 - 2*(x + 1)/x + 1) -␣
↪1/8*log(sqrt(x + 1)/sqrt(x) + 1) + 1/8*log(sqrt(x + 1)/sqrt(x) - 1)
```

```
>>> from sage.all import *
>>> x = var('x')    # create a symbolic variable
>>> integrate(sqrt(x)*sqrt(Integer(1)+x), x)
1/4*((x + 1)^(3/2)/x^(3/2) + sqrt(x + 1)/sqrt(x))/((x + 1)^2/x^2 - 2*(x + 1)/x + 1) -␣
↪1/8*log(sqrt(x + 1)/sqrt(x) + 1) + 1/8*log(sqrt(x + 1)/sqrt(x) - 1)
```

Agora vamos resolver uma equação quadrática com o Sage. O símbolo == representa igualdade no Sage.

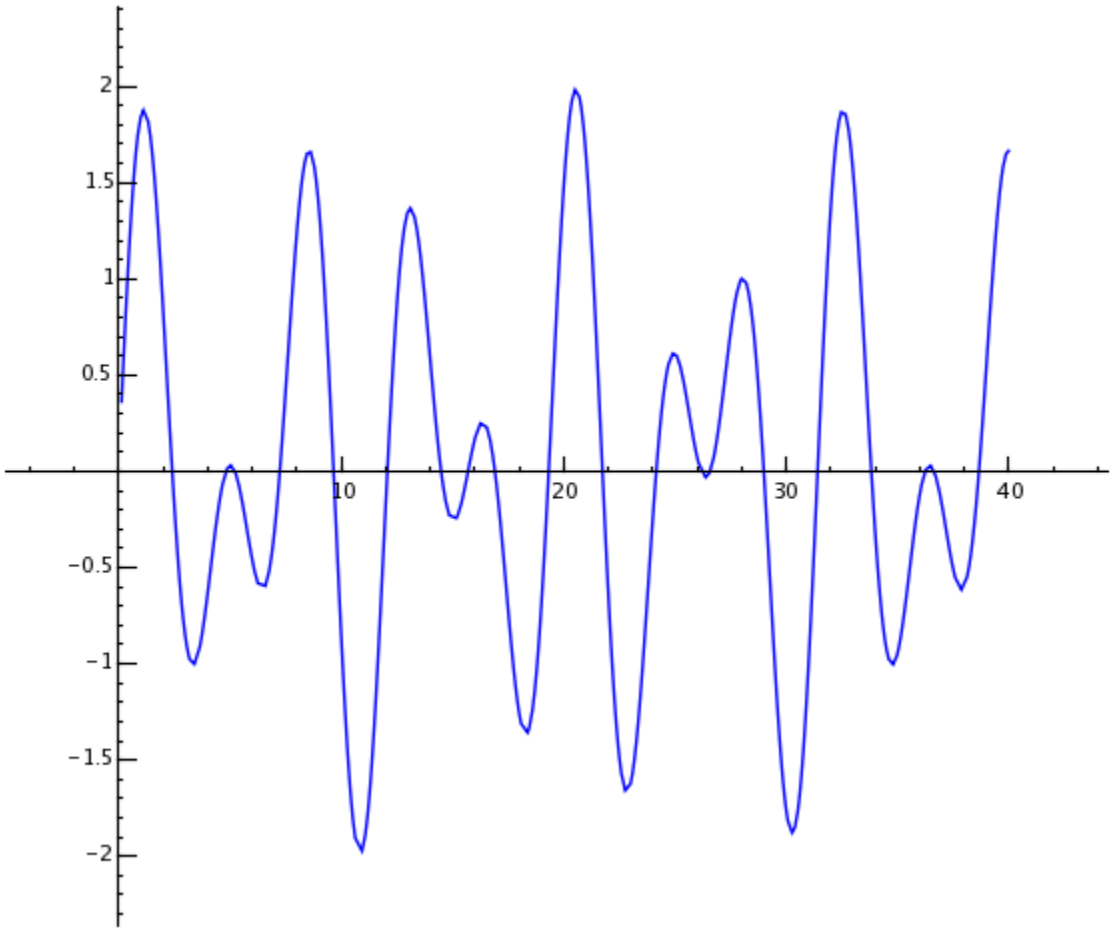
```
sage: a = var('a')
sage: S = solve(x^2 + x == a, x); S
[x == -1/2*sqrt(4*a + 1) - 1/2, x == 1/2*sqrt(4*a + 1) - 1/2]
```

```
>>> from sage.all import *
>>> a = var('a')
>>> S = solve(x**Integer(2) + x == a, x); S
[x == -1/2*sqrt(4*a + 1) - 1/2, x == 1/2*sqrt(4*a + 1) - 1/2]
```

O resultado é uma lista de igualdades.

```
sage: S[0].rhs()
-1/2*sqrt(4*a + 1) - 1/2
sage: show(plot(sin(x) + sin(1.6*x), 0, 40))
```

```
>>> from sage.all import *
>>> S[Integer(0)].rhs()
-1/2*sqrt(4*a + 1) - 1/2
>>> show(plot(sin(x) + sin(RealNumber('1.6')*x), Integer(0), Integer(40)))
```


2
1.5
1
0.5
−0.5
−1
−1.5
−2
10
20
30
40

CAPÍTULO 2

Cálculo Numérico com o Sage

Primeiro vamos criar uma matriz 500×500 de números aleatórios.

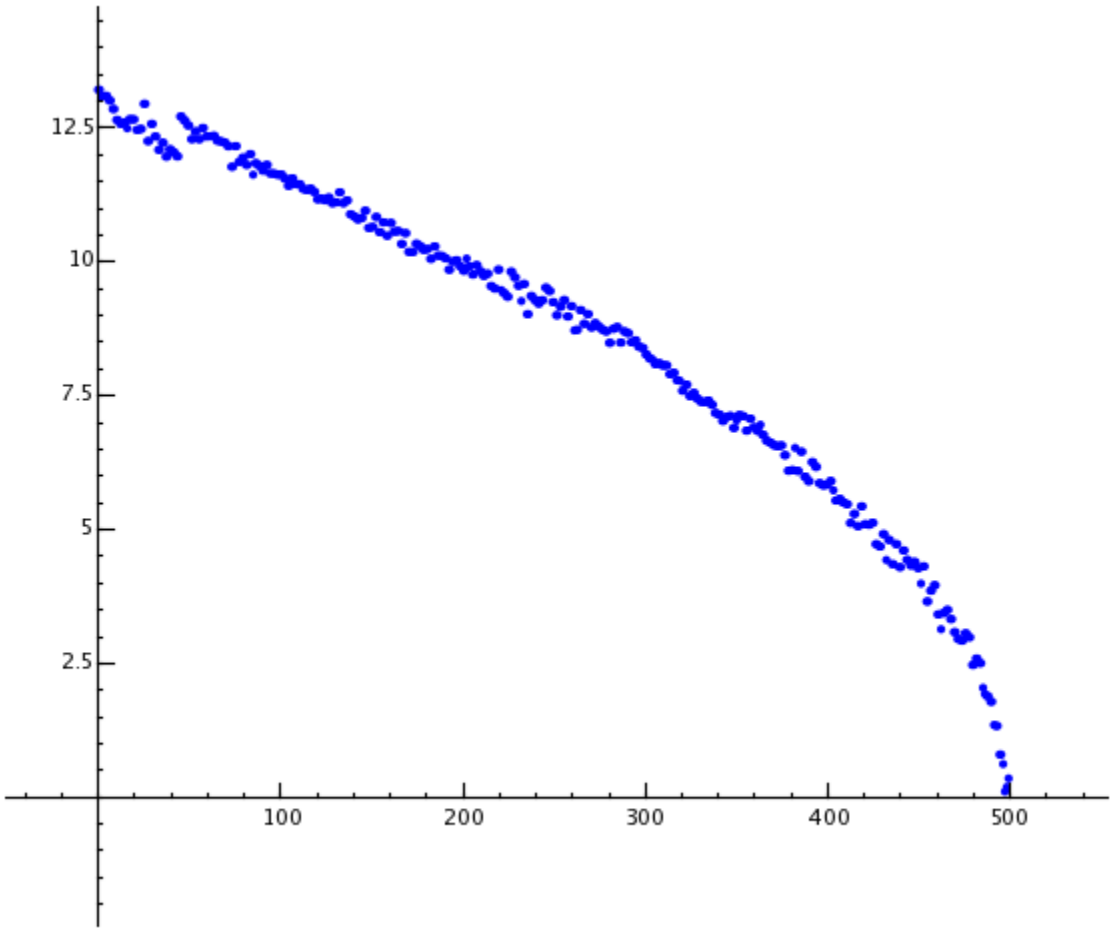
```
sage: m = random_matrix(RDF,500)
```

```
>>> from sage.all import *
>>> m = random_matrix(RDF,Integer(500))
```

Leva alguns segundos para calcular os autovalores dessa matriz e representá-los em um gráfico.

```
sage: e = m.eigenvalues()  #about 2 seconds
sage: w = [(i, abs(e[i])) for i in range(len(e))]
sage: show(points(w))
```

```
>>> from sage.all import *
>>> e = m.eigenvalues()  #about 2 seconds
>>> w = [(i, abs(e[i])) for i in range(len(e))]
>>> show(points(w))
```



Graças à biblioteca GMP (GNU Multiprecision Library), o Sage pode efetuar cálculos com números muito grandes, até mesmo com números com milhões de dígitos.

```
sage: factorial(100)
93326215443944152681699238856266700490715968264381621468592963895217599993229915608941463976156518286
sage: n = factorial(1000000)  #about 2.5 seconds
```

```
>>> from sage.all import *
>>> factorial(Integer(100))
93326215443944152681699238856266700490715968264381621468592963895217599993229915608941463976156518286
>>> n = factorial(Integer(1000000))  #about 2.5 seconds
```

Vamos calcular π com 100 algarismos decimais.

```
sage: N(pi, digits=100)
3.
↪1415926535897932384626433832795028841971693993751058209749445923078164062862089986280348253421170 68
```

```
>>> from sage.all import *
>>> N(pi, digits=Integer(100))
3.
↪1415926535897932384626433832795028841971693993751058209749445923078164062862089986280348253421170 68
```

Agora o Sage vai fatorar um polinômio em duas variáveis.

```
sage: R.<x,y> = QQ[]
sage: F = factor(x^99 + y^99)
sage: F
(x + y) * (x^2 - x*y + y^2) * (x^6 - x^3*y^3 + y^6) *
(x^10 - x^9*y + x^8*y^2 - x^7*y^3 + x^6*y^4 - x^5*y^5 +
 x^4*y^6 - x^3*y^7 + x^2*y^8 - x*y^9 + y^10) *
(x^20 + x^19*y - x^17*y^3 - x^16*y^4 + x^14*y^6 + x^13*y^7 -
 x^11*y^9 - x^10*y^10 - x^9*y^11 + x^7*y^13 + x^6*y^14 -
 x^4*y^16 - x^3*y^17 + x*y^19 + y^20) * (x^60 + x^57*y^3 -
 x^51*y^9 - x^48*y^12 + x^42*y^18 + x^39*y^21 - x^33*y^27 -
 x^30*y^30 - x^27*y^33 + x^21*y^39 + x^18*y^42 - x^12*y^48 -
 x^9*y^51 + x^3*y^57 + y^60)
sage: F.expand()
x^99 + y^99
```

```
>>> from sage.all import *
>>> R = QQ['x, y']; (x, y,) = R._first_ngens(2)
>>> F = factor(x**Integer(99) + y**Integer(99))
>>> F
(x + y) * (x^2 - x*y + y^2) * (x^6 - x^3*y^3 + y^6) *
(x^10 - x^9*y + x^8*y^2 - x^7*y^3 + x^6*y^4 - x^5*y^5 +
 x^4*y^6 - x^3*y^7 + x^2*y^8 - x*y^9 + y^10) *
(x^20 + x^19*y - x^17*y^3 - x^16*y^4 + x^14*y^6 + x^13*y^7 -
 x^11*y^9 - x^10*y^10 - x^9*y^11 + x^7*y^13 + x^6*y^14 -
 x^4*y^16 - x^3*y^17 + x*y^19 + y^20) * (x^60 + x^57*y^3 -
 x^51*y^9 - x^48*y^12 + x^42*y^18 + x^39*y^21 - x^33*y^27 -
 x^30*y^30 - x^27*y^33 + x^21*y^39 + x^18*y^42 - x^12*y^48 -
 x^9*y^51 + x^3*y^57 + y^60)
>>> F.expand()
x^99 + y^99
```

O Sage leva menos de 5 segundos para calcular de quantas maneiras pode-se particionar 10^8 como uma soma de inteiros positivos.

```
sage: z = Partitions(10^8).cardinality() #about 4.5 seconds
sage: str(z)[:40]
'1760517045946249141360373894679135204009'
```

```
>>> from sage.all import *
>>> z = Partitions(Integer(10)**Integer(8)).cardinality() #about 4.5 seconds
>>> str(z)[:Integer(40)]
'1760517045946249141360373894679135204009'
```

CAPÍTULO 3

Algoritmos incluídos no Sage

Ao usar o Sage, você acessa uma das maiores coleções disponíveis de algoritmos computacionais de código aberto.