Politechnika Wrocławska

Design and implementation issues of a computer algebra system in an interpreted, dynamically typed programming language

Mateusz Paprocki <mattpap@gmail.com>
Wrocław University of Technology

June 24, 2010

Wprowadzenie

Design and implementation issues
of a computer algebra system
in an interpreted, dynamically typed
programming language

- promotor: dr inż. Krzysztof Juszczyszyn
- język realizacji pracy: angielski

Plan prezentacji

- Wprowadzenie do SymPy
- Cele pracy dyplomowej
- Zrealizowane zadania
- Plany na przyszłość

Co to jest SymPy?

SymPy jest to biblioteka pisana w Pythonie do wykonywania:

- obliczeń symbolicznych
  - np. wyznaczanie całek, sum, granic
- obliczeń algebraicznych
  - np. obliczanie baz Gröbnera
- obliczeń numerycznych
  - np. rozwiązywanie równań nieliniowych

Co to jest SymPy?

SymPy jest to biblioteka pisana w Pythonie do wykonywania:

- obliczeń symbolicznych
  - np. wyznaczanie całek, sum, granic
- obliczeń algebraicznych
  - np. obliczanie baz Gröbnera
- obliczeń numerycznych
  - np. rozwiązywanie równań nieliniowych

Co to jest SymPy?

SymPy jest to biblioteka pisana w Pythonie do wykonywania:

- obliczeń symbolicznych
  - np. wyznaczanie całek, sum, granic
- obliczeń algebraicznych
  - np. obliczanie baz Gröbnera
- obliczeń numerycznych
  - np. rozwiązywanie równań nieliniowych

Co to jest SymPy?

SymPy jest to biblioteka pisana w Pythonie do wykonywania:

- obliczeń symbolicznych
  - np. wyznaczanie całek, sum, granic
- obliczeń algebraicznych
  - np. obliczanie baz Gröbnera
- obliczeń numerycznych
  - np. rozwiązywanie równań nieliniowych

Co to jest SymPy?

SymPy jest to biblioteka pisana w Pythonie do wykonywania:

- obliczeń symbolicznych
  - np. wyznaczanie całek, sum, granic
- obliczeń algebraicznych
  - np. obliczanie baz Gröbnera
- obliczeń numerycznych
  - np. rozwiązywanie równań nieliniowych

Dlaczego SymPy?

Dostępnych jest przecież wiele systemów matematycznych:

- systemy zamknięte:
  - Mathematica, Maple, Magma, ...
- systemy otwarte:
  - Axiom, GiNaC, Maxima, PARI, Sage, Singular, Yacas, ...

Problemy:

- systemy wprowadzają własny język programowania
  - należy się taki język nauczyć od podstaw
  - często jest to trudne zadanie, marnujemy czas
  - wyjątki: GiNaC i Sage
- występuje podział na:
  - hermetyczne i niedostępne jądro systemu
  - biblioteki pisane w języku danego systemu

Dlaczego SymPy?

Dostępnych jest przecież wiele systemów matematycznych:

- systemy zamknięte:
  - Mathematica, Maple, Magma, ...
- systemy otwarte:
  - Axiom, GiNaC, Maxima, PARI, Sage, Singular, Yacas, ...

Problemy:

- systemy wprowadzają własny język programowania
  - należy się taki język nauczyć od podstaw
  - często jest to trudne zadanie, marnujemy czas
  - wyjątki: GiNaC i Sage
- występuje podział na:
  - hermetyczne i niedostępne jądro systemu
  - biblioteki pisane w języku danego systemu

Dlaczego SymPy?

Dostępnych jest przecież wiele systemów matematycznych:

- systemy zamknięte:
  - Mathematica, Maple, Magma, ...
- systemy otwarte:
  - Axiom, GiNaC, Maxima, PARI, Sage, Singular, Yacas, ...

Problemy:

- systemy wprowadzają własny język programowania
  - należy się taki język nauczyć od podstaw
  - często jest to trudne zadanie, marnujemy czas
  - wyjątki: GiNaC i Sage
- występuje podział na:
  - hermetyczne i niedostępne jądro systemu
  - biblioteki pisane w języku danego systemu

Dlaczego SymPy?

Dostępnych jest przecież wiele systemów matematycznych:

- systemy zamknięte:
  - Mathematica, Maple, Magma, ...
- systemy otwarte:
  - Axiom, GiNaC, Maxima, PARI, Sage, Singular, Yacas, ...

Problemy:

- systemy wprowadzają własny język programowania
  - należy się taki język nauczyć od podstaw
  - często jest to trudne zadanie, marnujemy czas
  - wyjątki: GiNaC i Sage
- występuje podział na:
  - hermetyczne i niedostępne jądro systemu
  - biblioteki pisane w języku danego systemu

Dlaczego SymPy?

Dostępnych jest przecież wiele systemów matematycznych:

- systemy zamknięte:
  - Mathematica, Maple, Magma, ...
- systemy otwarte:
  - Axiom, GiNaC, Maxima, PARI, Sage, Singular, Yacas, ...

Problemy:

- systemy wprowadzają własny język programowania
  - należy się taki język nauczyć od podstaw
  - często jest to trudne zadanie, marnujemy czas
  - wyjątki: GiNaC i Sage
- występuje podział na:
  - hermetyczne i niedostępne jądro systemu
  - biblioteki pisane w języku danego systemu

Co chcemy osiągnąć?

- biblioteka pisana w Pythonie
  - bez nowego środowiska, języka, ...
  - działa od razu na dowolnej platformie
  - moduły nie–Pythonowe mogą być opcjonalne
- prostota architektury
  - relatywnie mała baza kodu źródłowego
  - łatwość w rozbudowie na dowolnym poziomie
- szeroka funkcjonalność
  - obsługa najważniejszych działów matematyki
  - wspieranie zaawansowanych metod i algorytmów
- użycie Cythona do optymalizacja wydajności
  - opcjonalnie, jako dodatek do wersji interpretowanej
- liberalna licencja: BSD
  - duża swoboda w użytkowaniu SymPy

Co chcemy osiągnąć?

- biblioteka pisana w Pythonie
  - bez nowego środowiska, języka, ...
  - działa od razu na dowolnej platformie
  - moduły nie–Pythonowe mogą być opcjonalne
- prostota architektury
  - relatywnie mała baza kodu źródłowego
  - łatwość w rozbudowie na dowolnym poziomie
- szeroka funkcjonalność
  - obsługa najważniejszych działów matematyki
  - wspieranie zaawansowanych metod i algorytmów
- użycie Cythona do optymalizacja wydajności
  - opcjonalnie, jako dodatek do wersji interpretowanej
- liberalna licencja: BSD
  - duża swoboda w użytkowaniu SymPy

Co chcemy osiągnąć?

- biblioteka pisana w Pythonie
  - bez nowego środowiska, języka, ...
  - działa od razu na dowolnej platformie
  - moduły nie–Pythonowe mogą być opcjonalne
- prostota architektury
  - relatywnie mała baza kodu źródłowego
  - łatwość w rozbudowie na dowolnym poziomie
- szeroka funkcjonalność
  - obsługa najważniejszych działów matematyki
  - wspieranie zaawansowanych metod i algorytmów
- użycie Cythona do optymalizacja wydajności
  - opcjonalnie, jako dodatek do wersji interpretowanej
- liberalna licencja: BSD
  - duża swoboda w użytkowaniu SymPy

Co chcemy osiągnąć?

- biblioteka pisana w Pythonie
  - bez nowego środowiska, języka, ...
  - działa od razu na dowolnej platformie
  - moduły nie–Pythonowe mogą być opcjonalne
- prostota architektury
  - relatywnie mała baza kodu źródłowego
  - łatwość w rozbudowie na dowolnym poziomie
- szeroka funkcjonalność
  - obsługa najważniejszych działów matematyki
  - wspieranie zaawansowanych metod i algorytmów
- użycie Cythona do optymalizacja wydajności
  - opcjonalnie, jako dodatek do wersji interpretowanej
- liberalna licencja: BSD
  - duża swoboda w użytkowaniu SymPy

Co chcemy osiągnąć?

- biblioteka pisana w Pythonie
  - bez nowego środowiska, języka, ...
  - działa od razu na dowolnej platformie
  - moduły nie–Pythonowe mogą być opcjonalne
- prostota architektury
  - relatywnie mała baza kodu źródłowego
  - łatwość w rozbudowie na dowolnym poziomie
- szeroka funkcjonalność
  - obsługa najważniejszych działów matematyki
  - wspieranie zaawansowanych metod i algorytmów
- użycie Cythona do optymalizacja wydajności
  - opcjonalnie, jako dodatek do wersji interpretowanej
- liberalna licencja: BSD
  - duża swoboda w użytkowaniu SymPy

Dlaczego wybraliśmy Pythona?

- prosty, interpretowany język programowania
  - łatwa do nauczenia składnia i semantyka
  - czytelny i łatwy w zarządzaniu kod
- język wykorzystywany przez potentatów
  - Google, NASA, ...
- olbrzymia liczba bibliotek
  - obliczenia numeryczne: NumPy, SciPy
  - fizyka, symulacje, bioinformatyka
  - wizualizacje 2D i 3D, wykresy
  - bazy danych, narzędzia sieciowe, ...
- prostota łączenia ze światem zewnętrznym
  - C/C++ poprzez natywne API lub Cythona
  - Fortran poprzez bibliotekę f2py

Dlaczego wybraliśmy Pythona?

- prosty, interpretowany język programowania
  - łatwa do nauczenia składnia i semantyka
  - czytelny i łatwy w zarządzaniu kod
- język wykorzystywany przez potentatów
  - Google, NASA, ...
- olbrzymia liczba bibliotek
  - obliczenia numeryczne: NumPy, SciPy
  - fizyka, symulacje, bioinformatyka
  - wizualizacje 2D i 3D, wykresy
  - bazy danych, narzędzia sieciowe, ...
- prostota łączenia ze światem zewnętrznym
  - C/C++ poprzez natywne API lub Cythona
  - Fortran poprzez bibliotekę f2py

Dlaczego wybraliśmy Pythona?

- prosty, interpretowany język programowania
  - łatwa do nauczenia składnia i semantyka
  - czytelny i łatwy w zarządzaniu kod
- język wykorzystywany przez potentatów
  - Google, NASA, ...
- olbrzymia liczba bibliotek
  - obliczenia numeryczne: NumPy, SciPy
  - fizyka, symulacje, bioinformatyka
  - wizualizacje 2D i 3D, wykresy
  - bazy danych, narzędzia sieciowe, ...
- prostota łączenia ze światem zewnętrznym
  - C/C++ poprzez natywne API lub Cythona
  - Fortran poprzez bibliotekę f2py

Dlaczego wybraliśmy Pythona?

- prosty, interpretowany język programowania
  - łatwa do nauczenia składnia i semantyka
  - czytelny i łatwy w zarządzaniu kod
- język wykorzystywany przez potentatów
  - Google, NASA, ...
- olbrzymia liczba bibliotek
  - obliczenia numeryczne: NumPy, SciPy
  - fizyka, symulacje, bioinformatyka
  - wizualizacje 2D i 3D, wykresy
  - bazy danych, narzędzia sieciowe, ...
- prostota łączenia ze światem zewnętrznym
  - C/C++ poprzez natywne API lub Cythona
  - Fortran poprzez bibliotekę f2py

Moja rola w projekcie

Czyli odrobina historii z moim udziałem

- początek współpracy w marcu 2007 roku
  - kilka prostych poprawek i rozszerzeń
- następnie Google Summer of Code 2007
  - algorytmy rozwiązywania równań rekurencyjnych
  - algorytmy sumowania nieznaczonego i oznaczonego
- no i tak już zostało:
  - algorytmy całkowania symbolicznego
  - struktury algebraiczne, wielomiany
  - upraszczanie wyrażeń, ...
- poza tym:
  - mentor w Google Summer of Code 2009 oraz 2010
  - prezentacja i krótki tutorial na EuroSciPy '09 (Lipsk)
  - prezentacja w ramach cyklu Py4Science (UC Berkely, 2010)
  - planowana prezentacja na EuroSciPy '10 (Paryż)

Moja rola w projekcie

Czyli odrobina historii z moim udziałem

- początek współpracy w marcu 2007 roku
  - kilka prostych poprawek i rozszerzeń
- następnie Google Summer of Code 2007
  - algorytmy rozwiązywania równań rekurencyjnych
  - algorytmy sumowania nieznaczonego i oznaczonego
- no i tak już zostało:
  - algorytmy całkowania symbolicznego
  - struktury algebraiczne, wielomiany
  - upraszczanie wyrażeń, ...
- poza tym:
  - mentor w Google Summer of Code 2009 oraz 2010
  - prezentacja i krótki tutorial na EuroSciPy '09 (Lipsk)
  - prezentacja w ramach cyklu Py4Science (UC Berkely, 2010)
  - planowana prezentacja na EuroSciPy '10 (Paryż)

Moja rola w projekcie

Czyli odrobina historii z moim udziałem

- początek współpracy w marcu 2007 roku
  - kilka prostych poprawek i rozszerzeń
- następnie Google Summer of Code 2007
  - algorytmy rozwiązywania równań rekurencyjnych
  - algorytmy sumowania nieznaczonego i oznaczonego
- no i tak już zostało:
  - algorytmy całkowania symbolicznego
  - struktury algebraiczne, wielomiany
  - upraszczanie wyrażeń, ...
- poza tym:
  - mentor w Google Summer of Code 2009 oraz 2010
  - prezentacja i krótki tutorial na EuroSciPy '09 (Lipsk)
  - prezentacja w ramach cyklu Py4Science (UC Berkely, 2010)
  - planowana prezentacja na EuroSciPy '10 (Paryż)

Moja rola w projekcie

Czyli odrobina historii z moim udziałem

- początek współpracy w marcu 2007 roku
  - kilka prostych poprawek i rozszerzeń
- następnie Google Summer of Code 2007
  - algorytmy rozwiązywania równań rekurencyjnych
  - algorytmy sumowania nieznaczonego i oznaczonego
- no i tak już zostało:
  - algorytmy całkowania symbolicznego
  - struktury algebraiczne, wielomiany
  - upraszczanie wyrażeń, ...
- poza tym:
  - mentor w Google Summer of Code 2009 oraz 2010
  - prezentacja i krótki tutorial na EuroSciPy '09 (Lipsk)
  - prezentacja w ramach cyklu Py4Science (UC Berkely, 2010)
  - planowana prezentacja na EuroSciPy '10 (Paryż)

Moja rola w projekcie

Czyli odrobina historii z moim udziałem

- początek współpracy w marcu 2007 roku
  - kilka prostych poprawek i rozszerzeń
- następnie Google Summer of Code 2007
  - algorytmy rozwiązywania równań rekurencyjnych
  - algorytmy sumowania nieznaczonego i oznaczonego
- no i tak już zostało:
  - algorytmy całkowania symbolicznego
  - struktury algebraiczne, wielomiany
  - upraszczanie wyrażeń, ...
- poza tym:
  - mentor w Google Summer of Code 2009 oraz 2010
  - prezentacja i krótki tutorial na EuroSciPy '09 (Lipsk)
  - prezentacja w ramach cyklu Py4Science (UC Berkely, 2010)
  - planowana prezentacja na EuroSciPy '10 (Paryż)

Moja rola w projekcie

Czyli odrobina historii z moim udziałem

- początek współpracy w marcu 2007 roku
  - kilka prostych poprawek i rozszerzeń
- następnie Google Summer of Code 2007
  - algorytmy rozwiązywania równań rekurencyjnych
  - algorytmy sumowania nieznaczonego i oznaczonego
- no i tak już zostało:
  - algorytmy całkowania symbolicznego
  - struktury algebraiczne, wielomiany
  - upraszczanie wyrażeń, ...
- poza tym:
  - mentor w Google Summer of Code 2009 oraz 2010
  - prezentacja i krótki tutorial na EuroSciPy '09 (Lipsk)
  - prezentacja w ramach cyklu Py4Science (UC Berkely, 2010)
  - planowana prezentacja na EuroSciPy '10 (Paryż)

Moja rola w projekcie

Czyli odrobina historii z moim udziałem

- początek współpracy w marcu 2007 roku
  - kilka prostych poprawek i rozszerzeń
- następnie Google Summer of Code 2007
  - algorytmy rozwiązywania równań rekurencyjnych
  - algorytmy sumowania nieznaczonego i oznaczonego
- no i tak już zostało:
  - algorytmy całkowania symbolicznego
  - struktury algebraiczne, wielomiany
  - upraszczanie wyrażeń, ...
- poza tym:
  - mentor w Google Summer of Code 2009 oraz 2010
  - prezentacja i krótki tutorial na EuroSciPy '09 (Lipsk)
  - prezentacja w ramach cyklu Py4Science (UC Berkely, 2010)
  - planowana prezentacja na EuroSciPy '10 (Paryż)

Cele pracy dyplomowej

Zbudować moduł algebry komputerowej dla SymPy:

- zgodnie z wymienionymi postulatami SymPy
- szeroki wybór algorytmów i struktur danych
- wprowadzenie architektury wielowarstwowej
- wprowadzenie struktur algebraicznych
  - wykorzystanie różnych typów bazowych
- wykorzystanie pure mode Cython
- ...

Cele pracy dyplomowej

Zbudować moduł algebry komputerowej dla SymPy:

- zgodnie z wymienionymi postulatami SymPy
- szeroki wybór algorytmów i struktur danych
- wprowadzenie architektury wielowarstwowej
- wprowadzenie struktur algebraicznych
  - wykorzystanie różnych typów bazowych
- wykorzystanie pure mode Cython
- ...

Cele pracy dyplomowej

Zbudować moduł algebry komputerowej dla SymPy:

- zgodnie z wymienionymi postulatami SymPy
- szeroki wybór algorytmów i struktur danych
- wprowadzenie architektury wielowarstwowej
- wprowadzenie struktur algebraicznych
  - wykorzystanie różnych typów bazowych
- wykorzystanie pure mode Cython
- ...

Cele pracy dyplomowej

Zbudować moduł algebry komputerowej dla SymPy:

- zgodnie z wymienionymi postulatami SymPy
- szeroki wybór algorytmów i struktur danych
- wprowadzenie architektury wielowarstwowej
- wprowadzenie struktur algebraicznych
  - wykorzystanie różnych typów bazowych
- wykorzystanie pure mode Cython
- ...

Cele pracy dyplomowej

Zbudować moduł algebry komputerowej dla SymPy:

- zgodnie z wymienionymi postulatami SymPy
- szeroki wybór algorytmów i struktur danych
- wprowadzenie architektury wielowarstwowej
- wprowadzenie struktur algebraicznych
  - wykorzystanie różnych typów bazowych
- wykorzystanie pure mode Cython
- . . .

Cele pracy dyplomowej

Zbudować moduł algebry komputerowej dla SymPy:

- zgodnie z wymienionymi postulatami SymPy
- szeroki wybór algorytmów i struktur danych
- wprowadzenie architektury wielowarstwowej
- wprowadzenie struktur algebraicznych
  - wykorzystanie różnych typów bazowych
- wykorzystanie pure mode Cython
- ...

Zaimplementowane algorytmy

- dekompozycja bezkwadratowa
  - Yun
- rozkład na czynniki proste
  - nad ciałami skończonymi
    - Berlekamp, Shoup, Zassenhaus
  - nad innymi dziedzinami
    - Cantor–Zassenhaus, Wang
- dekompozycja funkcjonalna
  - Landau–Zippel
- bazy Gröbnera
  - Buchberger
- izolacja pierwiastków
  - ułamki łańcuchowe, Collins–Krandick

Architektura wielowarstwowa

Poziomy L3 oraz L2

- poziom L3

```
>>> f3 = x**10 - 1
>>> %timeit factor_list(f3)
100 loops, best of 3: 5.57 ms per loop
```

- poziom L2

```
>>> f2 = Poly(x**10 - 1, x, domain='ZZ')
>>> %timeit f2.factor_list()
100 loops, best of 3: 2.15 ms per loop
```

Architektura wielowarstwowa

Poziomy L3 oraz L2

- poziom L3

```
>>> f3 = x**10 - 1
>>> %timeit factor_list(f3)
100 loops, best of 3: 5.57 ms per loop
```

- poziom L2

```
>>> f2 = Poly(x**10 - 1, x, domain='ZZ')
>>> %timeit f2.factor_list()
100 loops, best of 3: 2.15 ms per loop
```

Architektura wielowarstwowa

Poziomy L1 oraz L0

- poziom L1

```
>>> f1 = DMP([1, 0, 0, 0, 0, 0, 0, 0, 0, 0, -1], ZZ)
>>> %timeit f1.factor_list()
100 loops, best of 3: 1.90 ms per loop
```

- poziom L0

```
>>> f0 = [1, 0, 0, 0, 0, 0, 0, 0, 0, 0, -1]
>>> %timeit dup_factor_list(f0, ZZ)
100 loops, best of 3: 1.88 ms per loop
```

Architektura wielowarstwowa

Poziomy L1 oraz L0

- poziom L1

```
>>> f1 = DMP([1, 0, 0, 0, 0, 0, 0, 0, 0, 0, -1], ZZ)
>>> %timeit f1.factor_list()
100 loops, best of 3: 1.90 ms per loop
```

- poziom L0

```
>>> f0 = [1, 0, 0, 0, 0, 0, 0, 0, 0, 0, -1]
>>> %timeit dup_factor_list(f0, ZZ)
100 loops, best of 3: 1.88 ms per loop
```

Różne typy bazowe współczynników

Wprowadzenie

- liczby całkowite $\mathbb{Z}$
  - Python — `int`
  - SymPy — `Integer`
  - GMPY — `mpz`
  - ...
- liczby wymierne $\mathbb{Q}$
  - Python — `Fraction`
  - SymPy — `Rational`
  - GMPY — `mpq`
  - ...
- inne dziedziny ...

Różne typy bazowe współczynników

Wprowadzenie

- liczby całkowite $\mathbb{Z}$
  - Python — `int`
  - SymPy — `Integer`
  - GMPY — `mpz`
  - ...
- liczby wymierne $\mathbb{Q}$
  - Python — `Fraction`
  - SymPy — `Rational`
  - GMPY — `mpq`
  - ...
- inne dziedziny ...

Różne typy bazowe współczynników

Wprowadzenie

- liczby całkowite $\mathbb{Z}$
  - Python — `int`
  - SymPy — `Integer`
  - GMPY — `mpz`
  - ...
- liczby wymierne $\mathbb{Q}$
  - Python — `Fraction`
  - SymPy — `Rational`
  - GMPY — `mpq`
  - ...
- inne dziedziny ...

Różne typy bazowe współczynników

Benchmark

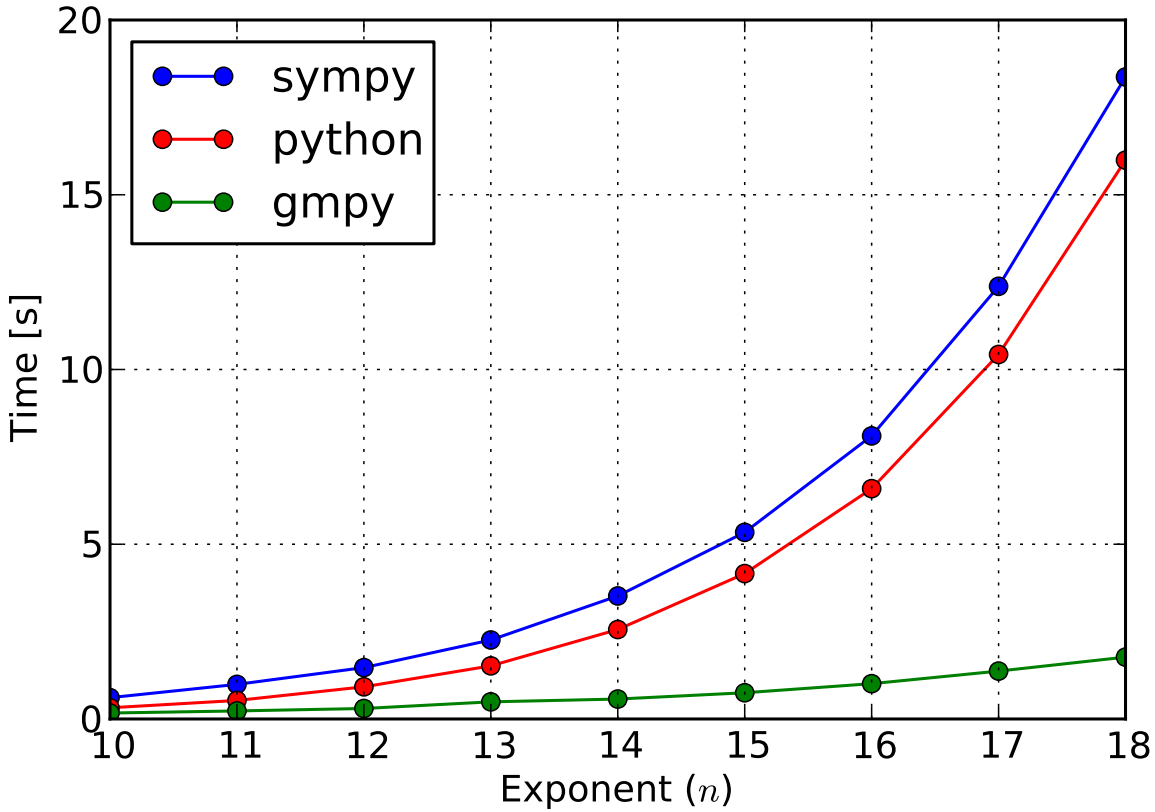

Figure: Rozkład na czynniki wielomianu $(1234x + 123y + 12z + 1)^n$

Plany na przyszłość

- zaangażowanie większej liczby osób w rozwój projektu
- więcej dokumentacji, przykładów, testów, benchmarków
- implementacja lepszych algorytmów
  - algorytmy dekompozycji, bazy Gröbnera, ...
- zastosowanie modułu w praktyce
  - projekt GSoC: Algorithms for Symbolic Integration
  - wykorzystanie SymPy w zdalnej edukacji matematyki

Dziękuję za uwagę!

Pytania, uwagi, dyskusja ...