Algebra Symboliczna

Wykład II

Andrzej Odrzywolek

Instytut Fizyki, Zakład Teorii Względności i Astrofizyki

10.10.2007, środa, 13:15

Dane kontaktowe

dr Andrzej Odrzywołek

pokój 447, IV piętro

E-mail: `odrzywolek@th.if.uj.edu.pl`

Wykład: środy 13.15-15.00 s. 128

Ćwiczenia: piątki 10.30-12.00

Konsultacje: środy ∼11-13, czwartki 10-12

WWW: http://ribes.if.uj.edu.pl/alsymb/

Praca interaktywna (1)

Obliczenie wartości numerycznej wyrażenia

N[wyrazenie,precyzja]

Przykłady

- Wartość numeryczna pierwiastka z dwóch:
  N[Sqrt[2]
- Rozwinięcie dziesiętne π do 100 miejsc po przecinku:
  N[Pi,100]

Uwagi

1. Precyzja obliczeń jest teoretycznie dowolna
2. Domyślnie obliczenia numeryczne wykonywane są przez (ko)procesor; ich dokładność wynosi **$MachinePrecision**

Praca interaktywna (2)

Operator odwołujacy się do poprzednich wyników

%, %%, %%%, %%...%, %n

Operator ten pozwala na ponowne użycie wyniku uzyskanego wcześniej w tej samej sesji z programem MATHEMATICA

Przykłady

- Poprzedni wynik: **%**; Pierwszy uzyskany wynik **%1**
- Dwa poprzednie wyniki pomnożone przez siebie: **% %%**

Uwagi

1. W praktyce najczęściej używamy operatorów **%** i **%%**
2. Operator ten dotyczy operacji wykonywanych przez jądro; jeżeli zostanie zrestartowane zaczyna od nowa.
3. Kolejność wyświetlania (GUI) *nie jest równoważna z kolejnością wykonywania operacji przez jądro!.*

Praca interaktywna (3)

Operator nakazujący zastosowanie polecenia do wyrażenia obliczonego przed nim //

Przykłady

- Przykład: ile wynosi wartość numeryczna wyrażenia $e^{\pi} - \pi^{e}$?: **Exp[Pi]-Pi^E // N**
- Sprowadź poprzednio uzyskany wynik do prostej postaci: **% // Simplify**
- Rozwiń wyrażenie: **(x^2-x-1)^7 // Expand**
- Zwiń do poprzedniej postaci: **% // Factor**

Uwagi

Operator ten mozna stosować łańcuchowo np:
-1 // Sqrt // N co jest równoważne **N[Sqrt[-1]]**

Zmienne i wyrażenia

Dowolne wyrażenie (wielomian, funkcję, równanie, wykres, macierz) można przypisać do pewnej zmiennej operatorem =

Przykłady

- **f = Sin[Sin[x]]**
- **w = (x+1)^5**
- **rownanie = x^2==-1**
- **napis = "algebra symboliczna"**
- **n=100**
- **t=12.7**

Uwaga!

Znak równości (w równaniach) to *podwójne* == ! Użycie w tym momencie operatora przypisania = to najczęściej popełniany błąd.

Funkcje

Najprostsza definicja funkcji 2^x

```
f[x_] := 2^x
```

1. **f** — nazwa nowej funkcji
2. **x_** — argument funkcji (lokalny) zakończony podkreśleniem
3. **:=** — operator „oblicz po wywołaniu f.”

Przykład:

- Funkcja dwóch zmiennych: **g[x_,y_] := Sin[x y]**
- Wywołanie funkcji: **g[t,Pi]**

Funkcje (2)

Operator „podkreślenia”

Podkreślenie służy do testowania „wzorców”; w tym przypadku jest to cokolwiek. Defnicja funkcji **f[x_] := ...** oznacza więc, że niezależnie od tego co wstawimy za **x**, wyrażenie po lewej stronie zostanie obliczone.

Przykłady

- Rozważmy 3 definicje funkcji:
  f[t_]:=N[t]
  g[t_?NumberQ]:=N[t]
  h[t_?NumericQ]:=N[t]
  oraz ich wywołania: **f[Pi]** (lub **g[Pi]**, **h[Pi]**). Funkcja **f** zawsze będzie próbowała dać jakiś wynik, funkcja **g** tylko gdy argument będzie liczbą całkowitą lub zmiennoprzecinkową, funkcja **h** dla argumentów posiadających wartość numeryczną.

Listy

Lista ma postać:
lista = {x,1,Pi,Sin[x],{0,1}, "abc", x+y }

Przykłady

- Do elementów listy odwołujemy się *podwójnymi nawiasami kwadratowymi* np:
  pierwszy element: **lista[[1]]** czwarty: **lista[[4]]** ostatni: **lista[[-1]]**
- lista może byc argumentem funkcji, np: **N[lista]** wyprodukuje listę, zawierającą wartości numeryczne tam gdzie to ma sens.
- wybierz liczby całkowite z listy: **Select[lista,IntegerQ]**

Uwaga

1. Elementy listy numerujemy od 1 !

Podstawy rysowania funkcji

Rysowanie funkcji w postaci jawnej: **Plot[Exp[x], {x,-3,3}]**
Rysowanie funkcji w postaci parametrycznej:
ParametricPlot[{2 Sin[t],3 Cos[t] }, {t,0, 2 Pi }]

Podstawowe opcje (przykłady)

- Oś pozioma od 0 do 1, na pionowej wszystkie możliwe wartości: **PlotRange → {{0,1},All}**
- Pierszy wykres czerwony, drugi niebieski: **PlotStyle → {Red, Blue}**

Uwagi

Aby narysować kilka funkcji na jednym wykresie należy podać *listę funkcji* jako pierwszy argument np: **Plot[{Sin[x],Cos[x]},{x,0,Pi}]**

Animacje

Animacja jest po prostu listą (tablicą) wykresów

Przykłady

- Seria wykresów funkcji $\sin(x+t)$ dla t zmieniajacego się od 0 do 1 co 0.05:
  Table[Plot[Sin[x+t],{x,-2 Pi, 2 Pi}],{t,0,1,0.05}]
- To samo na jednym wykresie:
  Plot[Evaluate[Table[Sin[x+t],{t,0,1,0.05}]],{x,-2 Pi, 2 Pi}]

Uwaga

1. Przy tworzeniu animacji prawie zawsze należy „zafiksować" rozmiar osi za pomocą odpowiedniej opcji **PlotRange**
2. Proszę zwrócić uwagę na polecenie **Evaluate** które wykonuje instrukcję **Table** *przed* instrukcją **Plot**

Najczęściej wykonywane zadania (symbolicznie!)

- Całka oznaczona, np: $\int_0^\pi \sin x dx$ **Integrate[Sin[x], {x,0,Pi}]**
- Rozwiązanie równania, np: $x^4 = 1$ **Solve[x^4==1,x]** lub **Reduce[x^4==1,x]**
- Upraszczanie/przekształcanie wyrażeń: **Simplify, Expand, Factor, FullSimplify**
- Pochodna funkcji np: $\frac{d}{dx} \ln x$ **D[Log[x],x**
- Granice, np: $\lim_{x\to 0} \sin x/x$: **Limit[Sin[x]/x,x→0]**

Uwaga

Bardziej szczegółowe omówienie najwazniejszych zagadnień na kolejnych wykładach

Najczęściej wykonywane zadania (numerycznie!)

- Całka oznaczona, np: $\int_0^\pi \sin x dx$ **NIntegrate[Sin[x], {x,0,Pi}]**
- Rozwiązanie równania, np: $x^4 = 1$ **FindRoot[x^4==1, {x,1}]**
- Interpolacja, np: Przybliżenie f. sinus w przedziale $(0, \pi/2)$:
  sin = { {0,0}, {Pi/6,1/2}, {Pi/4, Sqrt[2]/2}, {Pi/3,Sqrt[3]/2}, {Pi/2,1} } // Interpolation

Uwaga

1. **FindRoot** daje wynik w postaci *reguły transformacyjnej*; aby wydobyć rozwiązanie stosujemy **x /. %** (Wykład I)
2. **Interpolation** daje w wyniku *nową definicję funkcji*; aby ją narysować lub obliczyć wartość należy podać argument, np: **sin[Pi/4]**, **Integrate[sin[t],{t,0,Pi/2}]**, **Plot[sin[x],{x,0,Pi/2}]**

Obliczenia symboliczne *versus* numeryczne

Uwaga!

Obliczenia numeryczne prowadzą do nieuchronnej utraty informacji

Rozważmy trzy liczby: **p1=3.141592654** ,
p2=3141592654/1000000000 oraz **Pi**
Wynik mnożenia *numerycznego* **p1^2** to 9.869604403666765
Wynik mnozenia *symbolicznego* **p2^2** to 9.8696044036663716(0)
Wynik mnozenia *symbolicznego* **Pi^2** to 9.869604401089358618834490999

Działania wykonywane symbolicznie (np. zgodnie z regułami mnożenia ułamków) *nie prowadzą do utraty informacji* — mogą być prowadzone poprzez nieograniczoną liczbę kroków.
Użycie specjalnych stałych symbolicznych jak **Pi** pozwala na przechowywanie „nieskończenie dokładnej" (w sensie numerycznym) informacji, oraz operowanie nią wg. znanych (zaimplementowanych!) reguł.

Implementacja metod numerycznych w CAS

Powody używania numeryki

- Obliczenia numeryczne często prowadzą do wyniku *szybciej*, np. obliczenie całki oznaczonej z funkcji o „banalnym' przebiegu
- Większość problemów nie posiada rozwiązania analitycznego
- Zwykle ostatecznym celem jest uzyskanie wyniku liczbowego
- Algorytmy „niepewne" nie są używane
- Dysponujemy metodą pozwalającą na relatywnie łatwe testowanie wyników pod kątem poprawności
- Gigantyczna i ciągle rosnąca moc obliczeniowa komputerów pozwalająca na obliczenia numeryczne w czasie rzeczywistym

Implementacja metod numerycznych w CAS

Powody używania numeryki

- Obliczenia numeryczne często prowadzą do wyniku *szybciej*, np. obliczenie całki oznaczonej z funkcji o „banalnym' przebiegu
- Większość problemów nie posiada rozwiązania analitycznego
- Zwykle ostatecznym celem jest uzyskanie wyniku liczbowego
- Algorytmy „niepewne" nie są używane
- Dysponujemy metodą pozwalającą na relatywnie łatwe testowanie wyników pod kątem poprawności
- Gigantyczna i ciągle rosnąca moc obliczeniowa komputerów pozwalająca na obliczenia numeryczne w czasie rzeczywistym

Implementacja metod numerycznych w CAS

Powody używania numeryki

- Obliczenia numeryczne często prowadzą do wyniku *szybciej*, np. obliczenie całki oznaczonej z funkcji o „banalnym' przebiegu
- Większość problemów nie posiada rozwiązania analitycznego
- Zwykle ostatecznym celem jest uzyskanie wyniku liczbowego
- Algorytmy „niepewne" nie są używane
- Dysponujemy metodą pozwalającą na relatywnie łatwe testowanie wyników pod kątem poprawności
- Gigantyczna i ciągle rosnąca moc obliczeniowa komputerów pozwalająca na obliczenia numeryczne w czasie rzeczywistym

Implementacja metod numerycznych w CAS

Powody używania numeryki

- Obliczenia numeryczne często prowadzą do wyniku *szybciej*, np. obliczenie całki oznaczonej z funkcji o „banalnym' przebiegu
- Większość problemów nie posiada rozwiązania analitycznego
- Zwykle ostatecznym celem jest uzyskanie wyniku liczbowego
- Algorytmy „niepewne" nie są używane
- Dysponujemy metodą pozwalającą na relatywnie łatwe testowanie wyników pod kątem poprawności
- Gigantyczna i ciągle rosnąca moc obliczeniowa komputerów pozwalająca na obliczenia numeryczne w czasie rzeczywistym

Implementacja metod numerycznych w CAS

Powody używania numeryki

- Obliczenia numeryczne często prowadzą do wyniku *szybciej*, np. obliczenie całki oznaczonej z funkcji o „banalnym' przebiegu
- Większość problemów nie posiada rozwiązania analitycznego
- Zwykle ostatecznym celem jest uzyskanie wyniku liczbowego
- Algorytmy „niepewne" nie są używane
- Dysponujemy metodą pozwalającą na relatywnie łatwe testowanie wyników pod kątem poprawności
- Gigantyczna i ciągle rosnąca moc obliczeniowa komputerów pozwalająca na obliczenia numeryczne w czasie rzeczywistym

Implementacja metod numerycznych w CAS

Powody używania numeryki

- Obliczenia numeryczne często prowadzą do wyniku *szybciej*, np. obliczenie całki oznaczonej z funkcji o „banalnym' przebiegu
- Większość problemów nie posiada rozwiązania analitycznego
- Zwykle ostatecznym celem jest uzyskanie wyniku liczbowego
- Algorytmy „niepewne" nie są używane
- Dysponujemy metodą pozwalającą na relatywnie łatwe testowanie wyników pod kątem poprawności
- Gigantyczna i ciągle rosnąca moc obliczeniowa komputerów pozwalająca na obliczenia numeryczne w czasie rzeczywistym

Ładowanie pakietów

Sposób pierwszy:

«Graphics‘ ładuje wszystkie pakiety z danej grupy. Od tego momentu możemy ich używać w naszej sesji z MATHEMATICA.

Sposób drugi:

«Miscellaneous‘RealOnly‘ ładuje pojedynczą funkcję ***RealOnly*** dostępną w pakiecie ***Miscellaneous***. Od tego momentu możemy jej używać w naszej sesji z MATHEMATICA.

Przykład

Sekwencja poleceń ładująca i uruchamiająca benchmark (test szybkości) systemu:

«Utilities‘Benchmark‘

BenchmarkReport[]

Użyteczne pakiety

Przykłady

- **Graphics** $\longrightarrow$ liczne typy wykresów 2 i 3D
- **Miscellaneous‘RealOnly‘** $\longrightarrow$ obliczenia **bez** liczb zespolonych.
- **NumericalMath** $\longrightarrow$ wersje numeryczne licznych funkcji sybolicznych
- **Statistics‘NonlinearFit‘** $\longrightarrow$ dopasowywanie krzywych np. do wyników pomiarów
- **Calculus‘VectorAnalysis‘** $\longrightarrow$ rachunek wektorowy
- **Utilities‘Benchmark‘** $\longrightarrow$ sprawdza skuteczność komputera z punktu widzenia obliczeń symbolicznych.
- . . . oraz wiele innych przydatnych pakietów.