Metody numeryczne I

Programy wspomagające obliczenia – Maxima

Janusz Szwabiński

szwabin@ift.uni.wroc.pl

Maxima

- Pierwsze kroki
- Przekształcenia wyrażeń algebraicznych
- Trygonometria
- Pochodne i całki
- Macierze
- Równania i układy równań
- Równania różniczkowe zwyczajne
- Operacje wejścia/wyjścia
- Elementy programowania

Gdzie szukać?

- `http://maxima.sourceforge.net`
- Linux+, nr 10/2002

Pierwsze kroki

- `quit();` - zakończenie programu
- każde polecenie kończymy znakiem `;` lub `$`
- `(C1)`, `(C2)`, ..., oraz `(D1)`, `(D2)`, ... to odpowiednio etykiety poleceń użytkownika i odpowiedzi programu (zmiana w najnowszej wersji programu!)
- `describe(polecenie);` - opis polecenia
- `example(polecenie);` - przykład zastosowania polecenia
- znak `%` reprezentuje ostatni wynik
- `%TH(i)` to i-ty wynik od końca

- `ev(expr,flag)` - modyfikuje wyrażenie na podstawie użytej flagi (np. `numer`)
- `ALT+P` - poprzednie polecenie

Operacje arytmetyczne

`+`	dodawanie
`−`	odejmowanie
`*`	mnożenie skalarne
`/`	dzielenie
`^` lub `**`	potęgowanie
`.`	mnożenie macierzowe

Operatory przypisania

`:`	przypisanie wartości
`:=`	definiowanie funkcji
`=`	definiowanie równań

Wybrane funkcje matematyczne

- `sin`, `cos`, `tan`, `cot`
  `asin`, `acos`, `atan`, `acot`
  `sinh`, `cosh`, `tanh`, `coth`
  `asinh`, `acosh`, `atanh`, `acoth`
- `log`, `exp`, `sqrt`, `abs/cabs`
- `max`, `min`, `signum`
- `airy`, `bessel`, `gamma`

Przekształcanie wyrażeń algebraicznych

`expand`	rozwinięcie wyrażenia algebraicznego
`ratsimp`	uproszczenie wyrażenia
`factor`	rozkład wyrażenia na czynniki
`scsimp`	uproszczenie wyrażenia przy spełnieniu pewnych tożsamości

Trygonometria

`trigexpand`	rozwinięcie wyrażenia tryg.
`trigreduce`	uproszczenie wyrażenia tryg.

Uwaga!

Polecenie `halfangles:true;` spowoduje, że Maxima będzie wykorzystywać związki między funkcjami kątów połówkowych.

Pochodne i całki

```
diff(exp,v1,n1,v2,n2,...)
```

- `diff(sin(x)*cos(x),x,2);`
- `diff(sin(x)*exp(y^2),x,1,y,2);`


```
integrate(exp,var)
```

- `integrate(sin(x)**3,x);`
- `integrate(1/(x^2 + 1),x,0,inf);`

`residue(exp,var,val)`

- `residue(1/(x-%i),x,%i);`
- `residue(sin(a*x)/x^4,x,0);`

Inne przydatne polecenia

- `assume`
- `forget`
- `depends`
- `changevar`

Macierze

Definiowanie macierzy

- `matrix`
- `setelmx`
- `genmatrix`

- `matrix([1,2],[3,4]);`
- `setelmx(5,2,2,A);`
- `h[i,j]:=1/(i+j-1);`
  `genmatrix(h,3,3);`

Elementy macierzy

- `A[i,j]`

Mnożenie macierzowe (uwaga na spacje!)

- `A . B`

Wyznacznik i macierz odwrotna

- `determinant`
- `invert`

Transpozycja macierzy

- `transpose`

Wartości i wektory własne

- `eigenvalues`
- `eigenvectors`

Równania i układy równań

- `algsys([exp1,exp2,...],[var1,var2,...])`
- `linsolve([exp1,exp2,...],[var1,var2,...])`
- `solve(exp,var)`

Równania różniczkowe zwyczajne (RRZ)

Definiowanie równań różniczkowych zwyczajnych

- `x^2 'diff(y,x)+3*x*y=sin(x)/x;`
- `depends(y,x);`
  `x^2 diff(y,x)+3*x*y=sin(x)/x;`
- `x^2 diff(y(x),x)+3*x*y(x)=sin(x)/x;`

Rozwiązywanie RRZ

- `ode2(eqn,depvar,indvar)`
- `desolve([eq1,...,eqn],[dv1,...,dvn])`

Wartości początkowe i brzegowe

- `ic1(soln,iv=a,dv=b)`
- `ic2(soln,iv=a,dv=b,diff(dv,iv)=c)`
- `bc2(soln,iv=a,dv=b,iv=c,dv=d)`
- `atvalue` (tylko w połączeniu z `desolve`)

Wizualizacja danych

- `plot2D(expr,range,options)`
- `plot3D(expr,xrange,yrange,options)`
- `xgraph_curves(list)`

- `plot2D(sin(x),[x,-5,5]);`
- `plot2D(sec(x),[x,-2,2],[y,-20,20],[nticks,200]);`
- `plot2D([parametric,cos(t),sin(t),[t,-%pi*2,%pi*2]]);`
- `plot3D(x^2-y^2,[x,-2,2],[y,-2,2],[grid,12,12])`

Operacje wejścia/wyjścia

Zapisanie sesji do pliku

- `writefile("/home/szwabin/sesja.txt");`
- `closefile();`

Przekształcenie sesji do skryptu

- `stringout("plik.mac");`

Załadowanie skryptu

- `batch("plik.mac");`
- `demo("plik.dem");`

Eksport wyników do TEX–a

- `tex(expr,plik)`

Zapisanie wybranych wyników

- `save(plik,arg1,arg2,...)`

Elementy programowania

Zmienne lokalne

- `block([var:val],stmt)`

Instrukcja warunkowa

- `IF cond THEN stmt1 ELSE stmt2`

Pętle

- `FOR var:init STEP incr THRU limit DO body`
- `FOR var:init STEP incr WHILE cond DO body`
- `FOR var:init STEP incr UNLESS cond DO body`

- `block([fpprec:50],bfloat(%pi));`
- `h(x) := if x>=0 then 1 else 0;`
- `for a:-3 thru 26 step 7 do ldisplay(A)$`