Poprvé s MATLABem a Octavem

MATLAB je komerční software pro interaktivní řešení numerických problémů. MATLAB je akronymem pro Matrix Laboratory: prvotním cílem bylo pokrýt úlohy lineární algebry a zacházení s poli je jeho silnou stránkou dodnes. Programy v MATLABu zpracovává příkazový interpret, lze je i překládat. Množství dodávaných knihoven (toolboxů) je zdrcující, stejně jako cena systému. K dispozici jsou podobné zdarma dostupné systémy, které s MATLABem sdílejí syntaxi, méně pak knihovny; mezi nimi vyniká GNU Octave. MATLAB i Octave jsou vyvíjeny pro Windows i Linux, oba systémy nabízejí grafická rozhraní (GUI) i příkazový řádek (CLI).

Základ

help, doc cmd	nápověda (help zpřístupňuje komentář pod hlavičkou funkce)
%, %{ %}	řádkový a blokový komentář
line...	pokračování příkazu na dalším řádku
v(n), A(m,n), S.p	prvek vektoru, prvek matice, položka struktury
whos, who, clear	výpis proměnných, rušení proměnných
, ;	oddělovače příkazů na řádku, středník potlačí výpis př. 1 nebo 1, vypíše ans = 1, 1; mlčí, disp(1) i disp(1); vypíše 1
function(arg1,arg2)	funkční syntaxe, argumenty se přenesou hodnotou, př. disp(x) vypíše obsah proměnné x
function arg1 arg2	příkazová syntaxe, argumenty se přenesou jako řetězce, př. disp x vypíše text x
system('cmd');	příkaz pro operační systém, totéž: system cmd;
save, load, diary file	ukládání a načítání aktuálních proměnných, ukládání historie příkazů
quit	ukončení systému

Příkazy

Dostupné jsou běžné příkazy a konstrukce strukturovaného programování: přiřazení, vícecestný podmíněný příkaz včetně přepínače podle hodnoty výrazu, indexovaný cyklus a cyklus s podmínkou na začátku (v Octave i s podmínkou na konci) včetně skoků v cyklu. Odskok do funkce se provede voláním jména funkce, pro výpis hodnoty výrazu stačí jeho zápis nebo odeslání do funkce disp; výpisy se formátují příkazem format, případně v C-stylu funkcí sprintf.

výpis výrazu	samotný výraz (se středníkem bez výpisu), poslední výraz: ans; pro výpis i funkce disp
nastavení výpisu	format long, long e, (default) short, rat, hex; bez prázdných řádků: format compact
formátovaný výpis	sprintf(format,expressions), pro decimal %nd, float %nf, %n.nf, %n.ne, string %ns aj. př. pi, format long; 1., disp(sprintf('%s%9.6f','e =',e)) pro 3.1416 1 2.718282
přiřazení	variable=expression (pro skaláry i pole, na velikosti písmen záleží) př. x=1; X=2; disp(x+X); i=1; disp(i+j+I+J) pro 3, 1+3i
podmíněný příkaz	if expression, statements, elseif expression, statements, else, statements, end př. x=1; if x>0, s=1, elseif x<0, s=-1, else, s=0, end
přepínač	switch expression, case expression, statements, otherwise, statements, end výrazy v case větvích mohou být skaláry nebo seznamy hodnot v podobě cell arrays př. i=3; switch i, case 1, 'vetev 1', case {2,3}, 'vetev 2,3', otherwise, 'vetev else', end
indexovaný cyklus	for var=imin:stride:imax, statements, end (stride 1 lze vynechat) př. s=0; for i=1:10, s=s+i; end, s, nekonečný cyklus (jen MATLAB): for i=1:inf, i, end
cyklus s podmínkou	while expression, statements, end do, statements, until expression (jen Octave) př. s=0; i=0; while i<10, i=i+1; s=s+i; end, s, s=0; i=1; do, s=s+i; i=i+1; until i>10, s
skoky v cyklech	continue na další iteraci, break za cyklus př. for i=-2:2; if i==0, continue, end, 1/i, end
výjimky	try, body, catch, exception-handling, end př. for i=-1:1, try, a(i)=i, catch, 'out', end, end vypíše out out 1 (index musí být kladný) ale for i=-1:1, try, 1/i, catch, 'div zero', end, end vypíše -1, Inf, 1 (Inf a NaN se nezachytí)

Datové typy a výrazy

Matice jsou preferovanou datovou strukturou tvořenou prvky o stejném datovém typu (třídě). Jsou obdélníkové, indexované dvěma kladnými celočíselnými indexy (od 1 do end), mohou být řídké. Skalár je maticí 1x1, vektor může být sloupcový nx1 nebo řádkový 1xn, vícerozměrná pole jsou možná. Preferovaným datovým typem prvků jsou reálná čísla dvojnásobné přesnosti (8bytový double), dostupný je i 4bytový reálný typ (single), celá čísla jsou řešena v rámci reálného typu (flint hodnoty, vhodné pro indexování) i jako samostatné typy (int32, int64,

uint32 aj.), komplexní typ (complex, formálně atribut reálného typu) je samozřejmý, logický typ (logical) je často zastupitelný reálným, řetězce jsou řádkové vektory znaků (char). Cell arrays jsou pole s indexovanými prvky (obecně) různého typu, struktury jsou tvořeny pojmenovanými položkami (obecně) různého typu.

standardní datové typy	default double, single, complex obojí přesnosti, int8/uint8..int64/uint64, logical, char
literály	double: 1, 1., 1e0; complex: 0+1i, 1j, 1I, 1J; logical: false, true; char: '', 'A', 'bc'
konverzní funkce	double, single, complex, real, imag, logical, char, str2double, num2str, sprintf, cast př. complex(single(1)), imag(1i), logical(2), char(49), str2double('1'), cast(2^33,'int64')
operátory	aritmetické: +-*/, levé dělení \, umocnění ^; relační: ==, ~=, <, <=, >, >= logické: (negace) ~, (logický součin) &, (logický součet) |, zkrácené vyhodnocení: &&, || př. 1+2*3<9 & 2/3==3\2 & 0==false & 'A'<'a' pro 1, true pro 1 (operátory maticové: */\^' a prvkové: +, -, .*, ./, .\, .^, .' v dalším odstavci)
matice	pole s indexovanými prvky téhož typu, konstruktor: A=[...], přístup k prvku A(...) př. A=[1 2 3; 4 5 6] pro 2x3 pole, A(1,1) je prvek typu double
cell array	pole s indexovanými prvky různého typu, konstruktor: C={...}, přístup k prvku C{...} př. C={1,2+3i; false,'string'} pro 2x2 pole, C{1,1} typu double, C(1,1) 1prvkové cell array
struktura	konstruktor: S=struct('field',value,...), přístup k položce: S.field př. clear S T; S.f=1; S.ch='A'; T=struct('f',1,'ch','A'); whos S T; S.f, S.ch, T.f, T.ch struktura s polem: přiřazení pole do položky, př. S=struct('f',[1,1i]); S.f(1), S.f(2) pole struktur: přiřazení cell array do položky, př. S=struct('f',{1,1i}); S(1).f, S(2).f

Více o maticích

posloupnost s krokem	triplet xmin:stride:xmax, dublet xmin:xmax (stride 1), vše mohou být double výrazy př. 1:100 pro 100 prvků, 10:-1:0 11 prvků, 0:.1:1 11 prvků, ale 0:single(.1):1 10/11 prvků
n-prvková posloupnost	linspace(xmin,xmax,n) pro n prvků od xmin do xmax př. linspace(1,100), linspace(10,0,11), linspace(0,1,11) pro totéž jako výše
speciální matice	matice nul 1x1 zeros, nxn zeros(n), mxn zeros(m,n), a také ones, realmin, realmax, eps, nekonečno inf, nečíslo nan, e, pi, imaginární jednotky i, I, j, J, logické matice false, true jednotková eye, diagonální diag, trojúhelníkové horní triu a dolní tril, náhodné rand, randi př. eye(4), diag(1:4), triu(ones(4)), třídiagonální tril(triu(ones(4),-1),1) př. rand(m,n) pro <0,1), randi([imin imax],m,n) pro flint z <imin,imax>
konstruktor matice	[...]: seznam prvků, triplety, dublety, vektory, matice nebo nic, oddělovač řádků: ; př. prázdná matice [], řádkový vektor [0 1], sloupcový vektor [0; 1+j], řetězení ['ab' 'cde'] matice po řádcích [1 2,3; 2:4; 3:1:5], matice po blocích [ones(1) zeros(1,2)]
dotazovací funkce	size, př. A=ones(2,3); size(A) 2 3, prod(size(A)) 6, nulování matice A=zeros(size(A))
manipulační funkce	přetvarování reshape(A,m,n), zrcadlení flipud, fliplr, flipdim, rotace proti směru hodin rot90 př. reshape(1:6,2,3) pro 1 3 5;2 4 6, flipud([1;2]) 2;1, rot90([1 2]) 2;1, rot90([1 2],-1) 1;2
přístup k matici	A(...): v závorkách index (end pro poslední): v(1), v(end), A(end,end); nic A() pro všechno submatice: flint triplet, dublet, vektor, matice, př. triplet v(2:2:end), vektor v([2 4 6 8 10]) 1D indexování: pro A=ones(m,n) je A(:) totéž co reshape(A,m*n,1), A(i+(j-1)*m) je A(i,j)
prvkové funkce	působí prvek po prvku: abs, acos, asin, atan, atan2, ceil, conj, cos, exp, floor, imag, log, log10, log2, mod, real, rem, round, sign, sin, sqrt, tan, př. abs(-1:1) pro 1 0 1
vektorové funkce	působí na vektory nebo sloupce matic (nebo na dimenzi danou druhým argumentem): all, any, max, min, mode, mean, median, prod, sum, sort, std, var př. max(max(A)), průměry po sloupcích mean(A), po řádcích mean(A,2) nebo mean(A')'
maticové funkce	pro algebraické operace: det, rank, inv, norm, eig, svd př. det(diag(1:5)) pro 120, rank(diag(1:5)) pro 5, norm(rand(1,1000000),'inf') pro skoro 1
prvkové operace	prvek po prvku: +, -, .*, ./, .\, .^, př. [1 2]+[3 4] 4 6, [1 2].*[3 4] 3 8, transpozice .'
maticové operace	maticový součin *, řešiče soustav /\, mocnina ^, hermitovské sdružení ' př. skalární součin u*v' totéž co sum(u.*v), vnější součin u'*v, umocnění A^2 totéž co A*A řešení soustavy A x = b: A\b', zkouška A*ans (solver volen automaticky, jinak viz linsolve) hermit. sdružení a transpozice: C=complex(rand(5),rand(5)); all(all(C'==conj(C.'))) pro 1
maska	logické pole L vzniklé z relace s polem A, použitelné k výběru prvků z pole A př. A=reshape(-2:3,2,3); L=A>0; A(L) nebo jen A(A>0) pro 1;2;3 indexy nenulových prvků: find(A) pro seznam 1D indexů, [i,j]=find(A) seznam 2D indexů př. A=reshape(-2:3,2,3); A(find(A>0)) pro 1;2;3
indexovaný cyklus	for x=v prochází prvky řádkového vektoru, for x=A prochází sloupce matice př. v=[1 1 1]; for x=v, sum(x), end pro 1;1;1, A=ones(3); for x=A, sum(x), end pro 3;3;3
hříčky	magic(n) magický čtverec, gallery kolekce matic př. magic(3) pro 8 1 6; 3 5 7; 4 9 2, help gallery
vícerozměrná pole	př. A=zeros(2,3,4); A=reshape(1:24,2,3,4)

Funkce

Blokové funkce jsou samostatné programové jednotky (v MATLABu musí být každá v samostatném M-souboru, v Octavu téměř libovolně, ale před svým voláním a ne jako první v souboru). Není-li v hlavičce funkce naznačeno přiřazení návratové hodnoty, jde vlastně o proceduru. Anonymní a (zastarávající) inline funkce jsou tvořeny jediným definičním výrazem a mohou být (i v MATLABu) definovány nejen v M-souborech. Argumenty jsou funkcím předávány hodnotou, jsou tedy jen vstupní. Návratovou hodnotou může být pole i struktura, návratových hodnot může být více, čímž se simulují výstupní argumenty. Procedurální argumenty se řeší pomocí ukazatelů na funkce.

bloková funkce	function output=fname(arguments), statements, (kdekoliv return), end
	př. function r=iif(Expr,valT,valF), if Expr, r=valT; else, r=valF; end, end, iif(0,'A','bc')
pole návrat. hodnot	function array=fname(arguments), statements, end
	př. function r=f(x); r=[x, x.^2,x.^3]; end, f(2)
více návrat. hodnot	function [out1,out2,...]=fname(arguments), statements, end
	př. function [r1 r2 r3]=f(x); r1=x; r2=x.^2; r3=x.^3; end, f(2), [x y]=f(2), [x y z]=f(1:3)
	př. [status,output]=system('echo ahoj'); disp(output) pro ahoj
žádná návrat. hodnota	function fname(arguments), statements, end
	př. function pockej(n), 'waiting...', pause(n), end, pockej(2)
anonymní funkce	@(arguments) expression
	př. f=@(x) cos(x); f(0), f=@(x,y) x+y; f(1,2) pro 1 3
inline funkce	inline('string')
	př. f=inline('cos(x)'); f(0), f=inline('x+y'); f(1,2) pro 1 3
ukazatel na funkci	@function
	př. f=@sin; quad(f,0,pi), quad(@(x) x.^2,0,1) pro (určité integrály) 2 0.33333
M- a MEX-soubory	M: zdrojový soubor (pro skript, funkci, třídu; přípona .m), MEX: zdroj v C/C++/Fortranu
Octave skript s fcemi	př. 0; function r=f(x), r=x; end, f(1)
platnost proměnných	default: lokální proměnné, global: globální proměnné, persistent: trvalé lokální proměnné
přerušení programu	keyboard, pokračování return, prodleva pause(n), ladění dbstop

Numerické metody

linsolve(A,b,opts)	řešení soustavy algebraických rovnic A.x=b s volitelným popisem vlastností matice
polyval(p,x)	vyčíslení polynomu p (absolutní člen v posledním prvku) v x
polyfit(x,y,n)	polynom stupně n proložený body (x,y)
interp1(x,y,xi,method)	interpolace v xi mezi tabelovanými body (x,y), method např.: 'linear', 'spline'
fzero(f,x0)	nulový bod funkce f blízký hodnotě x0
quad(f,a,b)	určitý integrál funkce f na intervalu <a,b>
roots(p)	kořeny polynomu p
ode45(f,t,y0)	(MATLAB) řešení počáteční úlohy pro soustavu obyčejných diferenciálních rovnic
lsode(f,y0,t)	(Octave) obdoba matlabovské ode45 (Livermore Solver for ODEs)

Grafika

plot(x,y,string,...)	2D graf
hold on	podržet aktuální obrázek pro další příkazy, opak hold off
figure(n)	přepnutí na obrázek n
fplot(f,[x0 x1])	kreslení funkce f na <x0,x1>
title, xlabel, ylabel	nastavení popisů
text(x,y,string), gtext	text do obrázku
axis([..])	nastavení os, též axis off | equal | auto
plot(x,y1,x,y2), plot(x,[y1,y2])	více grafů
subplot(nx,ny,i)	více panelů
line types	- -- : -.
marker types	. o x + * s d v ^ < > p h
colors	y m c r g b w k
clf reset	clear current figure
shg	show graph window
bar, compass, feather, fill, hist, polar, quiver, rose, stairs	další typy obrázků
plot3(x,y,z,string,...)	3D graf
mesh(matice)	síť
surf(matice)	plocha v 3D
set(gcf,s-var,s-val,...)	nastavení parametrů obrázku (get current figure)

Hotové obrázky
load file, whos, image(var), colormap(var), axis image
Kreslení analytických funkcí pomocí Symbolic Math Toolboxu (MATLAB) viz níže.

Různé

tic, toc, t=toc	měření času

Symbolic Math Toolbox (MuPAD, součást MATLABu)

K provádění symbolických algebraických úprav (CAS, Computer Algebra System).

syms a, a=sym(a)	zavedení symbolů pro proměnné syms a b c d x, [a b]*[c d]' syms x, diff(x^2), int(x^2)
solve, dsolve string	řešení algebraických a diferenciálních rovnic symbolicky syms a b c x, solve a*x^2+b*x+c=0 syms a b c d p q x, solve('a*x+b*y=p','c*x+d*y=q'); [ans.x ans.y] syms x y, result=dsolve('Dx=y', 'Dy=-x'); result.x, result.y
vpa string precision	vyčíslení výrazu s volitelnou přesností (variable-precision arithmetic), př. vpa pi 100
ezplot(f,[x0 x1])	analytická funkce f(x) na <x0,x1> (easy plot), lze implicitní i parametrický zápis syms t x y; ezplot(@(x) x.^2,[0 1]); ezplot(x^2+y^2-1,[-1 1 -1 1]); ezplot(cos(t),sin(t))
ezpolar(f,[ph0 ph1])	pro funkci r=f(phi), př. ezpolar(1/t,[1 pi*6])
ezplot3(x,y,z,[t0 t1])	3D parametrická funkce s možností animace syms t x y z; x=cos(t); y=sin(t); z=t; ezplot3(x,y,z,[0 pi*6],'animate')
ezsurfc(f,[x0 x1 y0 y1])	plochy a izočáry f(x,y) (surface & contours), př. ezsurfc(x*y)

ezcontour, ezcontourf, ezmesh, ezmeshc, ezsurf

Octave

Balíčky: instalace a zavedení např. symbolic: pkg list; pkg install -forge symbolic; pkg load symbolic
(symbolic: balíček pro CAS, odvozeno od SymPy, solve/dsolve dosud neimplementováno)
Syntaktické přídavky (en.wikipedia.org/wiki/GNU_Octave#MATLAB_compatibility):
komentáře: #, řetězce v uvozovkách: "A", C-operátory: ++, --, += ad., rozšířené indexování: [1:10](end), linspace(1,10)(100), automatická expanze operandů a argumentů, mají-li jednotkovou velikost v jedné z dimenzí: ones(2,3)+ones(2,1), cyklus s podmínkou na konci: do until, kvalifikované end: endif, endfor, endwhile ad., definice funkčních jednotek na příkazovém řádku (nejen v M-souboru): function res=f(x), res=x; endfunction, f(1)

MATLAB vs. Fortran

s, m, n, imin, imax, istride, xmin, xmax, stride skaláry, u, v, b řádkové vektory, A, B matice téhož tvaru

	MATLAB	Fortran
speciální matice	A=zeros(m,n), B=ones(m,n)	real :: A(m,n)=0., B(m,n)=1.
konstruktor vektoru	[1 2], [1,2], [1:2]	[1,2], (/1,2/), [(i,i=1,2)]
konstruktor matice	[1 2; 3 4], reshape(1:4,2,2)'	transpose(reshape([1,2,3,4],[2,2]))
řetězení	[string1 string2]	string1//string2
indexový triplet	xmin:stride:xmax, xmin:xmax	imin:imax:istride, imin:imax
posloupnost	linspace(xmin,xmax,n)	[(xmin+(xmax-xmin)*(i-1)/(n-1),i=1,n)]
velikost, tvar	size(v), size(A), prod(size(A))	size(v), shape(A), size(A)
první a poslední prvek	v(1), v(end)	v(lbound(v,1)), v(ubound(v,1))
zrcadlení up-down	flipud(A)	A(ubound(A,1):lbound(A,1):-1,:)
zrcadlení left-right	fliplr(A)	A(:,ubound(A,2):lbound(A,2):-1)
rotace	rot90(A)	forall (i=1:n); A(n:1:-1,i)=A(i,:); endforall
prvkové funkce	abs(v), abs(A)	abs(v), abs(A)
součet prvků	sum(v), sum(sum(A)), sum(A)	sum(v), sum(A), sum(A,1)
počet kladných prvků	sum(sum(A>0))	count(A>0)
hodnota extrému	max(u), max(max(A))	maxval(u), maxval(A)
poloha extrému	[i,j]=find(u==max(u))	maxloc(u), maxloc(A)
nenulové prvky	find(u), find(A)	pack(u,u/=0), pack(A,A/=0)
prvkové operace	A+B, A-B, A.*B, A./B, A+s, ...	A+B, A-B, A*B, A/B, A+s, ...
skalární součin	u*v'	dot_product(u,v)
maticové násobení	A*B, A*v, v*A	matmul(A,B), matmul(A,v), matmul(v,A)
druhá mocnina	A^2, A*A	matmul(A,A)
transpozice	A.', A'	transpose(A), transpose(conjg(A))
řešení soustavy	A\b	LAPACK: call dgesv(n,nrhs,A,lda,ipiv,b,ldb,info)

Odkazy

MATLAB, www.mathworks.com, česká stránka www.humusoft.cz
GNU Octave, www.gnu.org/software/octave, online dokumentace www.gnu.org/software/octave/support.html
Knihy a PDF

Moler C., Numerical Computing with MATLAB, MathWorks, 2004, www.mathworks.com/moler (otec zakladatel)
Davis T. A., MATLAB Primer, CRC, 2011
Higham D. J., Higham N. J., MATLAB Guide, SIAM, 2005
Hanselman D. C., Littlefield B., Mastering MATLAB 7, Prentice Hall, 2005
Gerya T., Introduction to Numerical Geodynamic Modelling, Cambridge, 2010

L. H., 24. 11. 2015