Matlab

Algebra feladatok megoldása Matlab-ban

1. További mátrix és vektor műveletek

Mátrix megadása:
$>> A = [1, 2, 5; 11\ 86\ 2];$
Ezt kibővíthetjük a következőképpen:
$>> A = [A; 2, 4, 6]; C = [A, ones(3); zeros(3), eye(3)]$
Diagonális tömb mátrix szerkesztése a *blkdiag* utasítással:
$>> D = blkdiag(2 * eye(2), ones(2))$
Altömb ismétlésével megszerkesztett mátrix, repmat(A,m,n) utasítással:
$>> repmat(eye(2), 2)$
Mátrix dimenziójának módósítása a *reshape* utasítással:
$>> A = [1, 2, 3; 4, 1, 0]; B = reshape(A, 3, 2), 3 \times 2$-es mátrixot szerkeszt ahol az elemeket az A oszlopai szerint választja ki.
Diagonális mátrixok szerkesztése a *diag*(x,k) utasítással, amelynek hatśára az x vektor kerül a k átlóra, k=0 a főátló.
$>> diag([1\ 2\ 3])$ vagy $diag([-1\ 1], 1)$ vagy $diag([6\ 8], -3)$
Ha A egy mátrix; akkor
$>> A = [1, 3, 5; 5, 7, 1; 8, 9, 1]; diag(A), diag(diag(A))$
Az $A(:)$ utasítás hatására egy vektor jelenik meg amely az mátrix oszlopaiból áll.
Ennek segítségével például a következőképpen lehet egy négyzetes prímszámokból álló mátrixot megszerkeszteni:
$>> A = zeros(3); A(:) = primes(23); A = A'$
A transzponáltra azért van szükség, hogy a mátrix elemei ne az oszlopok, hanem a sorok szerint legyenek rendezve.
A [] jelölés az üres vagy 0×0 mátrixot jelenti. Ennek segítségével, ha törölni akarjuk az előző mátrix egy sorát, akkor
$>> A(2, :) = []$
Ha adott két x és y vektor, akkor ezek skaláris szorzatát a $dot(x, y)$ vektorszorzatát a $cross(x, y)$ utasítással lehet kiszámítani.
Egy x vektor esetén a $sort(x)$ növekvő a $sort(x, 'descend')$ csökkenő sorrendben rendezi az elemeket. Ha x mátrix, a rendezést az oszlopok szerint végzi el. A min és max utasítások hasonlóképpen működnek.
$>> A = [3, 1, 1; 4, 5, 1; 2, 5, 9]; min(A)$
eredménye egy vektor, amely minden oszlop minimális eleméből áll.
$>> [m, i] = min(A)$
parancs hatására az m a minimális elemekből álló mátrix i pedig megadja a minimális elem helyét.
$>> min(min(A))$

az A mátrix összes eleme közül adja meg a minimális elemet. Vagy másképpen:
$>> min(A(:))$
A $diff$ parancs segítségével a következő alakú különbségeket kapjuk: ha x egy n komponensű vektor akkor a $diff(x)$ egy n-1 komponensű vektor $[x_2 - x_1, x_3 - x_2, ..., x_n - x_{n-1}]$.
$switch$ struktúra használatának szintaxisa:
switch kifejezés case utasítások (otherwise utasítások) end
A következő példa egy x vektor p-normáját adja meg különböző p értékekre.
$>> switch(p)$
$case\ \ 1$
$y = sum(abs(x));$
$caes\ \ 2$
$y = sqrt(x' * x);$
$case\ \ inf$
$y = max(abs(x));$
$otherwise$
$error('p\ csak\ 1\ 2\ vagy\ inf')$
end
Függvény paraméterlistájában függvénynév is lehet ezen függvény kiértékelésére használható az $eval$ és a $feval$ utasítás. Más lehetőség egy függvény paraméterként való átadására a *function handle* használata ez a @ szimbólummal lehetséges.
<u>Például</u>
$>> ezplot('sin')$ helyett $>> ezplot(@sin)$
A $function\ \ handle$ a MatLab 6-os verziójában jelent meg, viszont vannak esetek, amikor ez nem működik csak a karakterlánccal történő alkalmazás. A két módszer közötti konverzió a $func2str$ és a $str2func$ utasításokkal lehetséges (lásd help).
Más mód az $ezplot$-nak átadni a függvényt:
$>> ezplot('x^2 - 1')$ vagy inline obijektumként
$>> ezplot(inline('exp(x) - 1'))$
Egy inline obijektum egy karakterlánccal megadott függvény, amely átadható egy változónak, kiértḱelhető stb.:
$>> f = inline('exp(x) - 1')$
$>> f(2)$
A MatLab automatikusan meghatározza és sorrendbe állítja a függvény argumentumait, ezen változtathatunk, ha nem tetszik:
$>> f = inline('log(a * x + 1)/(2 + y^2)') \rightarrow f(a, x, y)$
$>> f = inline('log(a * x + 1)/(2 + y^2)', 'x', 'y', 'a') \rightarrow f(x, y, a)$.

Rekurzív függvényhívás

Egy függvényen belül meghívhatjuk magát a függvényt is, például az n-dik fibonacci szám $f1 = 2, f2 = 2, f_k = f_{k-1} + f_{k-2}$ kiszámítására az m-file a következő lesz:

$function \quad f = fibnum(n)$
% $fibnum(n)$ az n-dik fibonacci számot generálja
$if \quad n <= 1$
$f = 1$
$else$
$f = fibnum(n-1) + fibnum(n-2);$
end

Adat beolvasása és kiiratása a képernyőre

z=input('kerem a matrixot:');
z=input('kerem a nevet','s');
disp('szoveg') vagy disp(v);
$fprintf('azelozomatrix\%12.5f\backslash n', z);$
$fprintf('e\_format: \%12.5e\backslash n', 12345.6) \rightarrow$ e_format: 1.23456e+04
fprintf('e_format:%12.5e',12345.2);
fprintf('f_format %12.3f$\backslash n', 7.23462) \rightarrow$ egy sorba írja ki: $e\_format : 1.23452e+04 \quad f\_format : 7.235$
$sprintf('\%0.5g', (1+sqrt(5))/2)1.618$
$sprintf('\%0.5g', 1/eps)4.5036e+15$
$sprintf('\%15.5f', 1/eps)4503599627370496.00000$
$sprintf('\%d', round(pi))3$
$sprintf('\%s', 'hello')hello$
$sprintf('The \quad array \quad is \quad \%dx\%d.', 2, 3)Thearrayis2x3.$
$sprintf('\backslash n')$ is the line termination character on all platforms.

2. Lineáris egyenletrendszer megoldása

Legyen m az egyenletek, n pedig az ismeretlenek száma. Az $Ax = B$ egyenletrendszer megoldására a MatLab különböző numerikus módszereket használ attól függően, hogy az együtthatómátirx milyen tulajdonságokkal rendelkezik. Ha B is egy mátrix melynek p oszlopa van, akkor a MatLab az $Ax(:,j) = B(:,j), j = 1, ..., p$ egyenletrendszereket oldja meg.

Ha $m = n$ vagyis a mátrix négyzetes, akkor az $x = A\backslash b$ utasítással a MatLab Gauss eliminációt alkalmaz részleges főelemkiválasztással. A módszer alkalmazása során ellenőrzi a mátrix úgynevezett kondíciószámát, ha ez túl nagy, hibaüzenetet ír ki:

$>> x = hilb(15)\backslash ones(15,1)$

Ha az A mátrix szimmetrikus és pozitív definit, akkor Cholesky felbontást alkalmaz az $A\backslash b$ utasítás hatására.

Ha $m > n$ akkor a rendszer túlhatározott, ilyenkor az $A\backslash b$ MatLab utasítás a megoldást a legkisebb négyzetek eleve alapján oldja meg, vagyis minimalizálja a $norm(A * x - b)$ kifejezést x szerint. Ha A rangja m, akkor van egyértelmű megoldás, ha a rangja kisebb, akkor k nem nulla elemű megoldást ad QR faktorizációt alkalmazva. A meghatározatlan ismeretleneket lenullázza, de erről egy hibaüzenettel értesíti a felhasználót.

Ha $m < n$ vagyis több oszloppal mint sorral rendelkezünk, szintén QR-felbontást alkalmaz, a meghatározatlan ismeretleneket lenullázza, de ekkor nem ad ki hibaüzenetet. A több megoldásból csak egyet határoz meg a MatLAb.

Ha a jobb oldalon egy mátrix van, tehát B egy mátrix akkor az egyenletrendszerek megoldására az LU felbontást alkalmazza. Ekkor ugyanaz az együtthatómátrix, erre egyszer elvégzi az LU felbontást az

$[LU] = lu(A)$

utasítással, majd a megoldásokat:

$for \quad j = 1 : p$

$y = L\backslash b; x = U\backslash y;$ % az $Ly = b$ és a $Ux = y$ rendszer megoldása.

end

Az LU felbontásban szereplő összes mátrixot az

$[L, U, P] = lu(A)$

utasítással kapjuk meg, ahol P egy permutációs matrix.

Cholesky felbontás esetén

$U = chol(A)$; %U egy felső háromszög mátrix, amelyre $A = U'U$

$for \quad j = 1 : p$

$y = U'\backslash b; x = U\backslash y;$ % az $Ly = b$ és a $Ux = y$ rendszer megoldása.
end
QR-faktorizáció esetén
$[Q, R] = qr(A)$ % Q egy ortogonális és R egy felső háromszög mátrix, A=QR
Példa: Adjatok meg egy mátrixot számítsátok ki ennek LU, QR és Cholesky féle felbontását.
Az egyenletrendszer megoldása az $inv(A) * b$ utasítással nem ajánlott az inverz kiszámításához szükséges műveletszám miatt.
Használhatjuk a linsolve utasítást: linsolve(A,b,opts), amely az $A * x = b$ egyenletrendszert oldja meg a opts által megadott mátrixra. Lehetséges opciók: LT(alsóháromszög), UT(felsőháromszög), SYM-szimmetrikus, POSDEF-pozitív definit, RECT-ortogonális, TRANSA-transzponált.
Példa:
$>> A = triu(rand(5, 3)); x = [1, 1, 1, 0, 0]'; b = A' * x;$
$>> y1 = A'\backslash b;$
$>> opts.UT = true; opts.TRANSA = true;$
$>> y2 = linsolve(A, b, opts)$

3. Adatokhoz polinom hozzárendelése

A MatLab egy polinomot

$$p(x) = p_1x^n + p_2x^{n-1} + ... + p_{n+1}$$

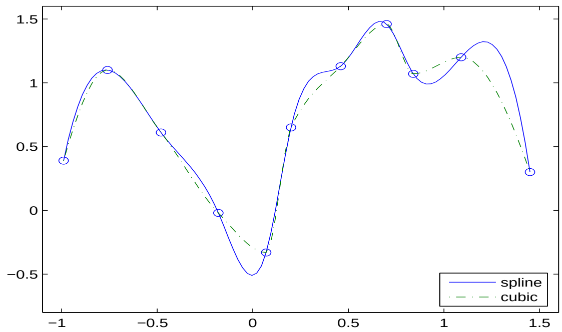
az együtthatók vektorával ad meg:
$>> p = [p_1, p_2, ..., p_{n+1}]$
Ennek kiértékelése a $y = polyval(p, x)$ utasítással, ahol x egy mátrix is lehet.
Gyökeit a $roots(p)$ utasítással, a gyökök ismeretében a polinomot a $poly$ függvény segítségével kaphatjuk meg.
A $polyfit(x, y, n)$ utasítással egy n-edfokú polinomot kapunk a legkisebb négyzetek elve alapján az x, y adatok felhasználásával.
Sajátérték illetve sajátvektor meghatározására:
$>> eig(A)$ %sajátértékeket kapjuk ($Ax = \lambda x$)
$>> [V, D] = eig(A)$ %V a sajátvektorokból áll, D a sajátértékekből

4. Ritka mátrixok

Egy létező mátrix átvihető ritka mátrixos alakba a $sparse$ utasítás-sal. A MatLab a ritka mátrixokat egy listával ábrázolja (i, j) $a(i, j)$ elrendezésben; $A = [1, 0, 3, 4; 4, 3, 0, 1; 1, 0, 3, -4; 0, 0, -2, 0]$;
$A = sparse(A)$
fordítva, a $full(A)$ utasítással.
$A = sparse(ia, ja, a, m, n)$ egy $m \times n$-es a-ban megadott elemekkel, ia, ja, a méretei megegyeznek.
A ritka mátrix alkotóelemeit a $find$ utasítással lehet megszerezni.
$>> [ia, ja, a] = find$- hatására három oszlopvektor jelenik meg.
$>> [ia, ja] = find(A)$ -megadja a nemnulla elemek indexeit, az $s = nonzeros(A)$ pedig a nemnulla elemeket.
Sávos mátixok szekesztśeére használható a $spdiags$ utasítás.
$A = spdiags(B, d, m, n)$-összeállítja az $m \times n$-méretű ritka A mátrix átlóit a B mátrix oszlpaiból, a d vektor komponensei szerinti helyeken.
$A = spdiags(B, d, A)$-kicseréli az A azon átlóit a B mátrix oszlopaira, amelyeknek sorszámát a d vektor megadja
$[B, d] = spdiags(A)$-az A összes nemzérus átlóit keresi és azokból állítja össze a B oszlopait, az információt az átlók elhelyezéséről a d-ben adja meg.
<u>Példa</u>
$>> n = 4;$
$>> B = [1 : n]';$
$>> B = [B, B, B];$
$>> A = spdiags(B, [-1, 0, 2], n, n);$
$A = full(A)$
$speye(n)$-adja az n-ed rendű ritkamátrixú egységmátrixot.
Lineáris rendszerek megoldása ritka mátrixok esetén is történhet az $x = A\backslash$ utasítással. Hasonlóképpen LU, $Cholesky$ felbontásokat is.

5. Interpoláció

Az (x,y) adtokhoz szerkeszti meg az interpolációs polinomot és adja meg az értékét egy xi pontban:
$yi = polyval(x, y, xi, \texttt{módszer})$
ahol a módszer paraméter a következő lehet:



1. ábra.

-'nearest'-a legközelebbi szomszéd szerinti interpoláció
-'lineáris'-lineáris interpoláció
-'spline'-köbös spline interpoláció
-'cubic'vagy 'pchip'-szakaszonként köbös Hermite interpoláció
<u>Példa</u>
$>> x = [-0.99, -0.76, -0.48, -0.18, 0.07, 0.2, ...0.46, 0.7, 0.84, 1.09, 1.45];$
$>> y = y = [0.39, 1.1, 0.61, -0.02, -0.33, 0.65, ...1.13, 1.46, 1.07, 1.2, 0.3];$
$>> plot(x, y, o); holdon$
$xi = linspace(min(x), max(x), 100);$
$ys = interp1(x, y, xi, spline);$
$yc = interp1(x, y, xi, cubic);$
$h = plot(xi, ys, -, xi, yc, -.);$
$legend(h, spline, cubic, 4)$
$axis([-1.1, 1.6, -0.8, 1.6])$
A $pp = spline(x, y)$ megadja az együtthatókat egy pp struktúrában amely tartalmazza az alappontokat, részintervallumokat, rendjét és dimenzióját. A $ppval$ utasítással kiértékelhető egy ilyen struktúrával megadott spline függvény.
Beolvasuk a billentyűzetről több pontot ezeknek megfelelő spline polinomot szerkesztünk.
$>> axis([0, 1, 0, 1]);$
$>> hold\ on$
$>> [x, y] = ginput;$
$>> data = [x; y];$
$>> t = linspace(0, 1, length(x));$
$>> tt1 = linspace(0, 1, 20);$
$>> tt2 = linspace(0, 1, 150);$

$>> pp = spline(t, data);$
$>> yy1 = ppval(pp, tt1);$
$>> yy2 = ppval(pp, tt2);$
$>> plot(x, y, o, yy1(1, :), yy1(2, :), 'r', yy2(1, :), yy2(2, :), 'g');$
$>> hold\ off$
Többváltozós függvény approximációja az interp2 utasítással.
$Z = interp2(x, y, z, X, Y, modszer)$
x,y az alappontok, z a függvény értéke a pontokon, X,Y a pont amelyben közelíteni szeretném a függvény értékét, a módszer a következő lehet:
-'nearest'-a legközelebbi szomszéd szerinti interpoláció
-'lineáris'-bilineáris interpoláció
-'spline'-köbös spline interpoláció
-'cubic'-kétváltozós köbös interpoláció
Példa
$>> [X, Y] = meshgrid(-3 : 1 : 3); Z = peaks(X, Y); surf(X, Y, Z)$
$>> [XI, YI] = meshgrid(-3 : 0.25 : 3); ZI1 = interp2(X, Y, Z, XI, YI, nearest);$
$>> ZI2 = interp2(X, Y, Z, XI, YI, linear);$
$>> ZI3 = interp2(X, Y, Z, XI, YI, cubic);$
$>> ZI4 = interp2(X, Y, Z, XI, YI, spline); subplot(2, 2, 1);\ surf(XI, YI, ZI1)$
$>> title(nearest)subplot(2, 2, 2);\ surf(XI, YI, ZI2);\ title(linear)$
$>> subplot(2, 2, 3);\ surf(XI, YI, ZI3)\ title(cubic)\ subplot(2, 2, 4),$
$>> surf(XI, YI, ZI4)title(spline)$

2. ábra.