Mátrix Inverz Kiszámítása Klasszikus és LU Faktorizációval

Kovács Bálint-Hunor

2023. október 15.

1. Bevezetés

Ebben a tanulmányban azt a problémát vizsgáljuk, hogy hogyan lehet kiszámítani egy $n \times n$ méretű mátrix inverzét. A mátrixot generáljuk az egyszerűség kedvéért. Az inverz számítást két módszerrel végezzük: a klasszikus mátrix inverz számítás módszerrel és az LU faktorizációval. Összehasonlítjuk ezeknek a módszereknek az eredményeit, és elemzést végzünk, hogy melyik módszer hatékonyabb bizonyos esetekben.

2. Mátrix Inverziós Módszerek

A mátrix inverz számítás alapvető művelet a lineáris algebrában. Két közönséges módszer a mátrix inverz kiszámítására:

2.1. Klasszikus Mátrix Inverzió

A klasszikus módszer a mátrix inverz kiszámítására olyan műveleteket használ, mint a sorcsökkentés, elemi soroperációk és az adjugált mátrix számítása. Ez a módszer széles körben használt és könnyen megvalósítható.

2.2. LU Faktorizációs Módszer

Az LU (Alsó-Felső) faktorizációs módszer a kiindulási mátrixot két háromszög mátrix szorzataként bontja fel, L (alsó háromszög) és U (felső háromszög). $L \times U =$ A.
A faktorizáció a Gauss-eliminációs folyamat során történik. Amint a mátrixot felbontottuk L és U értékekre, könnyebb lineáris egyenletrendszerek megoldása és az inverz számítása.

Az LU faktorizáció módszer előnyös lehet nagyobb méretű mátrixok esetén.

3. Mátrix Inverzió Implementációk MATLAB-ban

3.1. Klasszikus Mátrix Inverzió Implementáció

Az alábbi MATLAB kód bemutatja a klasszikus mátrix inverziót.

```
function A_inv = my_traditional_inverse(A)
[n, ~] = size(A);
A_inv = eye(n);

for k = 1:n
    % Megkeressuk a pivot elemet
    pivot = A(k, k);
    if abs(pivot) < 1e-10
        error('Matrix is singular.');
    end

    % Skalazzuk a sorokat, hogy a pivot elem 1 legyen
```

```
        A(k, :) = A(k, :) / pivot;
        A_inv(k, :) = A_inv(k, :) / pivot;

        % Nullazzuk a tobbi elemet a k. oszlopban
        for i = 1:n
            if i ~= k
                factor = A(i, k);
                A(i, :) = A(i, :) - factor * A(k, :);
                A_inv(i, :) = A_inv(i, :) - factor * A_inv(k, :);
            end
        end
    end
    end
```

A kód magyarázata a következő:
A kimeneti mátrixot inicializáljuk az egységmátrix-al. A ciklusban megkeressük a pivot elemet, és skálázzuk a sorokat, hogy a pivot elem 1 legyen. Ezután nullázzuk a többi elemet a k. oszlopban. A ciklus végén megkapjuk az inverz mátrixot.

3.2. LU Faktorizációs Módszer Implementáció

Az alábbi MATLAB kód bemutatja az LU faktorizációs módszert.

```
    function A_inv = my_lu_inverse(A)
    [n, ~] = size(A);
    L = eye(n);
    U = zeros(n);

    for k = 1:n
        U(k, k:n) = A(k, k:n) - L(k, 1:k-1) * U(1:k-1, k:n);
        L(k+1:n, k) = (A(k+1:n, k) - L(k+1:n, 1:k-1) * U(1:k-1, k)) / U(k,
            k);
    end

    I = eye(n);
    Y = L \ I;
    A_inv = U \ Y;
    end
```

A kód magyarázata a következő:
A kimeneti mátrixot inicializáljuk az egységmátrix-al. A ciklusban megkeressük a pivot elemet, és skálázzuk a sorokat, hogy a pivot elem 1 legyen. Ezután nullázzuk a többi elemet a k. oszlopban. A ciklus végén megkapjuk az inverz mátrixot.

4. Mátrix Inverzió Tesztelése

A teszteléshez egy $n \times n$ méretű mátrixot generálunk, ahol $n = 4$.

```
clear all; %#ok<*CLALL>
close all;
clc;
N = 4;
A = rand(N, N);
disp('Matrix:');
disp(A);
```

```matlab

% Hagyomanyos inverz kiszamitasa
tic;
A_inv_traditional = my_traditional_inverse(A);
time_traditional = toc;

% LU felbontas Crout modszerrel
tic;
A_inv_crout = my_lu_inverse(A);
time_crout = toc;

% Ellenorzes
tolerance = 1e-6;
if norm(A_inv_traditional - A_inv_crout) < tolerance
    disp('Results match within tolerance.');
    disp('Inverse (custom - traditional):');
    disp(A_inv_traditional);
else
    disp('Results do not match.');
    disp('Traditional inverse:');
    disp(A_inv_traditional);
    disp('Inverse with LU decomposition and Crout''s method (custom):');
    disp(A_inv_crout);
end

% Idok kiirasa
fprintf('Time taken for custom traditional matrix inversion: %.6f seconds\n', time_traditional);
fprintf('Time taken for custom LU decomposition with Crout''s method: %.6f seconds\n', time_crout);
```

A kimenet a következő:

```
Matrix:
0.2725    0.4931    0.1349    0.3069
0.3438    0.9576    0.8482    0.9434
0.6118    0.2305    0.9267    0.6790
0.1700    0.4979    0.7514    0.6095

Results match within tolerance.
Inverse (custom - traditional):
3.5065    -2.3519    1.2650    0.4652
5.7605    -4.9132    -1.6112    6.4988
4.3193    -6.6971    -0.6330    8.8958
-11.0092    12.9265    1.7438    -14.7655

Time taken for custom traditional matrix inversion: 0.004478 seconds
Time taken for custom LU decomposition with Crout's method: 0.030248 seconds
```

5. Összefoglalás

A tesztelés eredménye alapján látható, hogy mindkét módszer helyes eredményt adott. A klasszikus módszer gyorsabb volt, mint az LU faktorizációs módszer ebben az esetben, viszont nagyobb méretű mátrixok esetén az LU faktorizációs módszer lényegesen hatékonyabb.

Példa egy 1000×1000 méretű mátrixra (a mátrixot és az inverz mátrixot ezúttal nem íratom ki, mert túl hosszú lenne):

```
Time taken for custom traditional matrix inversion: 16.874812 seconds
Time taken for custom LU decomposition with Crout's method: 0.889298
    seconds
```