UNIVERSITAT POLITÈCNICA DE CATALUNYA
BARCELONATECH
Centre de la Imatge i la Tecnologia Multimèdia

Lab 1

Implementing a solver for systems of linear equations

Julen Cayero, Cecilio Angulo

Bachelor's Degree in Video Game Design and Development

Outline

UNIVERSITAT POLITÈCNICA DE CATALUNYA
BARCELONATECH
Centre de la Imatge i la Tecnologia Multimèdia

Outline

UNIVERSITAT POLITÈCNICA DE CATALUNYA
BARCELONATECH
Centre de la Imatge i la Tecnologia Multimèdia

It wouldn't be too difficult to implement a naive solver using the Gauss method.
We will need 2 functions:

1. Triangulate a given matrix
2. Backtracking of a triangular matrix

However in practice there are some problems associated with this way of proceed related with:

1. Avoid handling with large numbers
2. Proceed efficiently
3. Identify critical cases (Indeterminate or incompatible system)

UNIVERSITAT POLITÈCNICA DE CATALUNYA
BARCELONATECH
Centre de la Imatge i la Tecnologia Multimèdia

Outline

UNIVERSITAT POLITÈCNICA DE CATALUNYA
BARCELONATECH
Centre de la Imatge i la Tecnologia Multimèdia

Naïve Pseudocode I
Class assignment

Simple MatLab code for triangulate a matrix:

```
function [At, bt]= ownTriangulation(A,b)

% TO DO
% Aa = concatenation of A and B

% TO DO
% L = number of unknowns

% TO DO
% for loop c = from 1 until # columns -1
    % for loop r = from c+1 until #rows
            %row_r = pivot*row_r - subpivot* row_p

% TO DO
% At = part of Aa
% bt = part of Aa
```

Test:

$$\mathbf{A} = \begin{pmatrix} 5 & 4 & 2 \\ 2 & 1 & 1 \\ 1 & 2 & -3 \end{pmatrix} \quad \boldsymbol{b} = \begin{pmatrix} 5 \\ 1 \\ 0 \end{pmatrix} \quad \rightarrow \quad \mathbf{A}_T = \begin{pmatrix} 5 & 4 & 2 \\ 0 & -3 & 1 \\ 0 & 0 & 45 \end{pmatrix} \quad \boldsymbol{b}_T = \begin{pmatrix} 5 \\ -5 \\ 45 \end{pmatrix}$$

UNIVERSITAT POLITÈCNICA DE CATALUNYA
BARCELONATECH
Centre de la Imatge i la Tecnologia Multimèdia

Naïve Pseudocode II

Class assignment

Simple MatLab code for backtracking:

```
function [x] = backSubs(A,b)

    % TO DO
    % L = length of b
    % prealocate the vector x in memory initialize x at 0

    % solve for the last element

    % loop for q = L-1 until 1
        % solve for x
        %x_r = 1/diagonalTerm_r * (RHS_r - offdiagonalTerms_{r+1:end} * ...
              x_{r+1 to end})
```

Test:

$$\mathbf{A}_T = \begin{pmatrix} 5 & 4 & 2 \\ 0 & -0.6 & 0.2 \\ 0 & 0 & -3 \end{pmatrix} \quad \boldsymbol{b}_T = \begin{pmatrix} 5 \\ -1 \\ -3 \end{pmatrix} \quad \rightarrow \quad x = \begin{pmatrix} -1 \\ 2 \\ 1 \end{pmatrix}$$

Outline

Handling large numbers

Computer does not like large numbers [1]. Try:

```
a = 1E99;
b = 1E99 + 1;
c = a-b
```

- Which value do you expect for c?
- What is the value of c that MatLab throws?

When triangulating, the operation

$$r_j = r_j Aa(p,p) - r_p Aa(j,p)$$

Tends to increase the value of the entries in A.

[1] In case you're interested, there are special ways of dealing with those numbers, however it will be complex and computational expensive

UNIVERSITAT POLITÈCNICA DE CATALUNYA
BARCELONATECH
Centre de la Imatge i la Tecnologia Multimèdia

Handling large numbers II

Things we can do:

1. Since we can multiply and divide every row by a non-zero number

$$r_j = r_j - r_p Aa(j,p)/Aa(p,p)$$

   However we must ensure that A(p,p) is far enough from 0.

2. Since we can choose any row for triangulate. Select the row with the biggest (absolut value) possible pivot.

$$r_j = r_j - r_p Aa(j,p)/Aa(p,c)$$

   or even better, calculate $s_j = \max_i |A(j,i)|$ once for every row and choose the pivots such that:

$$p = \operatorname{argmax}_j \frac{|Aa(j,c)|}{s_j}$$

   That is call scaled partial pivoting

UNIVERSITAT POLITÈCNICA DE CATALUNYA
BARCELONATECH
Centre de la Imatge i la Tecnologia Multimèdia

Exchange rows means wasting time

We can do that without actually make a real exchange of rows. For this purpose:

- Maintain a list of the visited rows (in order)
  - initialise as an empty vector `vr = []`
  - Whenever you visit a row `vr = [vr p]`
- Maintain a list of the unvisited rows
- initialise as an empty vector `uvr = [1:L]`
- Whenever you visit a row extract the visited row from the vector. Use `uvr = setdiff(uvr,p)`

Note: that, after make the Gaussian elimination Aa(vr,:) is a triangular matrix with 0's under the diagonal. It means that the backsubstiuttion algorithm must be adapted.
But is easy. Whenever you access an element of a row you must do it by using the pointer.
E.g. the last component of $\boldsymbol{x}$ was calculated before as

$$x_L = b(L)/x(L, L)$$

Now that must be substituted by;

$$x_L = b(rv(L))/x(rv(L), L)$$

UNIVERSITAT POLITÈCNICA DE CATALUNYA
BARCELONATECH
Centre de la Imatge i la Tecnologia Multimèdia

Identify critical cases

Problems

Indeterminate or Incompatible systems of equations?

By maintaining the strategy of selecting the row with the biggest pivot, we can check when the pivot ≈ 0, we are going to be in one of those critical cases.

Set a threshold value (something small $\approx 1E-8$) and compare the pivot value. If exceeded, return x = 'NaN' and a flag to the non-solved value.

Efficiency
Problems

- Try to use matrix multiplications
- Try to avoid loops
- Do not waste time putting 0's that later you are not going to use

UNIVERSITAT POLITÈCNICA DE CATALUNYA
BARCELONATECH
Centre de la Imatge i la Tecnologia Multimèdia

Outline

Modified MatLab code for triangulate a matrix:

```
function [At ,bt, vr, flag] = ownTriangulation_mod(A,b)

% flag = -1 -> indeterminate or incompatible
% flag = 0 -> solvable

% TODO: initialise Th

% TODO: Aa = concatenation of A and b

% TODO: L = number of unknowns

% TODO: Initialize vr and uvr vectors

% TODO: create the sj vector

% TO DO: for loop c = from 1 until # columns -1

    % TO DO: implement a function that Given Aa, sj and uvr returns
            % the abs value of the pivot and the row of the pivot

    % TO DO: Check if the returned value is under the threshold
```

continues on the next slide

UNIVERSITAT POLITÈCNICA DE CATALUNYA
BARCELONATECH
Centre de la Imatge i la Tecnologia Multimèdia

Modified MatLab code for (virtually) triangulate a matrix:

```
    % TO DO: Check if the returned value is under the threshold

    % TO DO: Update the lists to remember the row pivot as visited

    %  TO DO: for loop rs in uvr, modify the uvr

    % TO DO: Update the lists to remember the row pivot as visited

    %  TO DO: for loop rs in uvr, modify the uvr

% TO DO: check if the last unvisited row has a unvalid pivot nad introduce
% the last row in the lists as visited

% TO DO: At as part of Aa
% TO DO: bt as part of Aa
```

UNIVERSITAT POLITÈCNICA DE CATALUNYA
BARCELONATECH
Centre de la Imatge i la Tecnologia Multimèdia

Pseudocode
Lab Homework

Modified MatLab code for backward substitution:

```
function [x] = backSubs_mod(At,bt,vr)
% TO DO L = length of b

% prealocate the vector x in memory initialize x at 0

% solve for the last element taking into account the pointer

% loop for q = L-1 until 1

    %solve x
```

Principal function that calls the other two:

```
function [x, flag] = ownGauss_mod(A,b)
% flag = -1 -> indeterminate or incompatible
% flag = 0 -> solvable

[At ,bt, vr, flag] = ownTriangulation_mod(A,b)

%TO DO: check the flag and return x = NaN if not solvable

[x] = backSubs_mod(At,bt,vr)
```

Tests

Lab Homework

We give you an additional `tests.m` file with several calls to your main function. Use it to verify that your functions is properly implemented.