Class 2 (17.01.2017)

- Hope you have written the C code for solving *m* simultaneous equations with *n* unknowns (*m* < *n*) to obtain basic solutions.
- You should have knowledge on Basic Solution, Basic Feasible Solution, Degenerate and Non-degenerate Basic Feasible Solution, Extreme Points.
- Assume non-negativity constraints for solving the following.
- **Write a menu-driven program** for obtaining Degenerate and Non-degenerate Basic Solution, Basic Feasible Solution, Extreme Points.
- **Using your code get the results of the following problems.**

1. Find all basic solutions for the system of simultaneous equations. Determine the degenerate and non-degenerate basic solutions and basic feasible solutions separately.;
    a. $2x_1 + 3x_2 + 4x_3 = 5, 3x_1 + 4x_2 + 5x_3 = 6.$
    b. $2x_1 + x_2 + 4x_3 = 11, 3x_1 + x_2 + 5x_3 = 14.$
    c. $3x_1 + x_2 + 5x_3 + x_4 = 12, 2x_1 + 4x_2 + x_3 + 2x_5 = 8.$
    d. $2x_1 + 6x_2 + 2x_3 + x_4 = 3, 6x_1 + 4x_2 + 4x_3 + 6x_4 = 2.$

2. Obtain all extreme points and the corresponding optimal solution of the following LPP.
    a. Maximize $5x_1 + 3x_2$ subject to $3x_1 + 5x_2 \leq 15, 5x_1 + 2x_2 \leq 10.$
    b. Maximize $2x_1 + x_2$ subject to $x_1 + 2x_2 \leq 10,\ x_1 + x_2 \leq 6,\ x_1 - x_2 \leq 2,\ x_1 - 2x_2 \leq 1.$

****

Class 1 (10.01.2017)

1. Write a C code for the following algorithm.
2. Use this algo as a function and solve *m* simultaneous equations with *n* unknowns (*m*<*n*) to obtain basic solutions.
3. Check your program for the following examples and count the number of basic solutions:
   (a) $x_1 + x_2 + S_1 = 40, 2x_1 + x_2 + S_2 = 60$
   (b) $2x_1 + x_2 + S_1 = 100, x_1 + x_2 + S_2 = 80, x_1 + S_3 = 40$

Gauss-Seidel Method Algorithm:

1. Start
2. Declare the variables and read the order of the matrix n
3. Read the stopping criteria er
4. Read the coefficients aim as

   Do for i=1 to n

   Do for j=1 to n

   Read a[i][j]

   Repeat for j

   Repeat for i
5. Read the coefficients b[i] for i=1 to n
6. Initialize x0[i] = 0 for i=1 to n
7. Set key=0
8. For i=1 to n

   Set sum = b[i]

For j=1 to n

If (j not equal to i)

Set sum = sum – a[i][j] * x0[j]

Repeat j

x[i] = sum/a[i][i]

If absolute value of ((x[i] – x0[i]) / x[i]) > er, then

Set key = 1

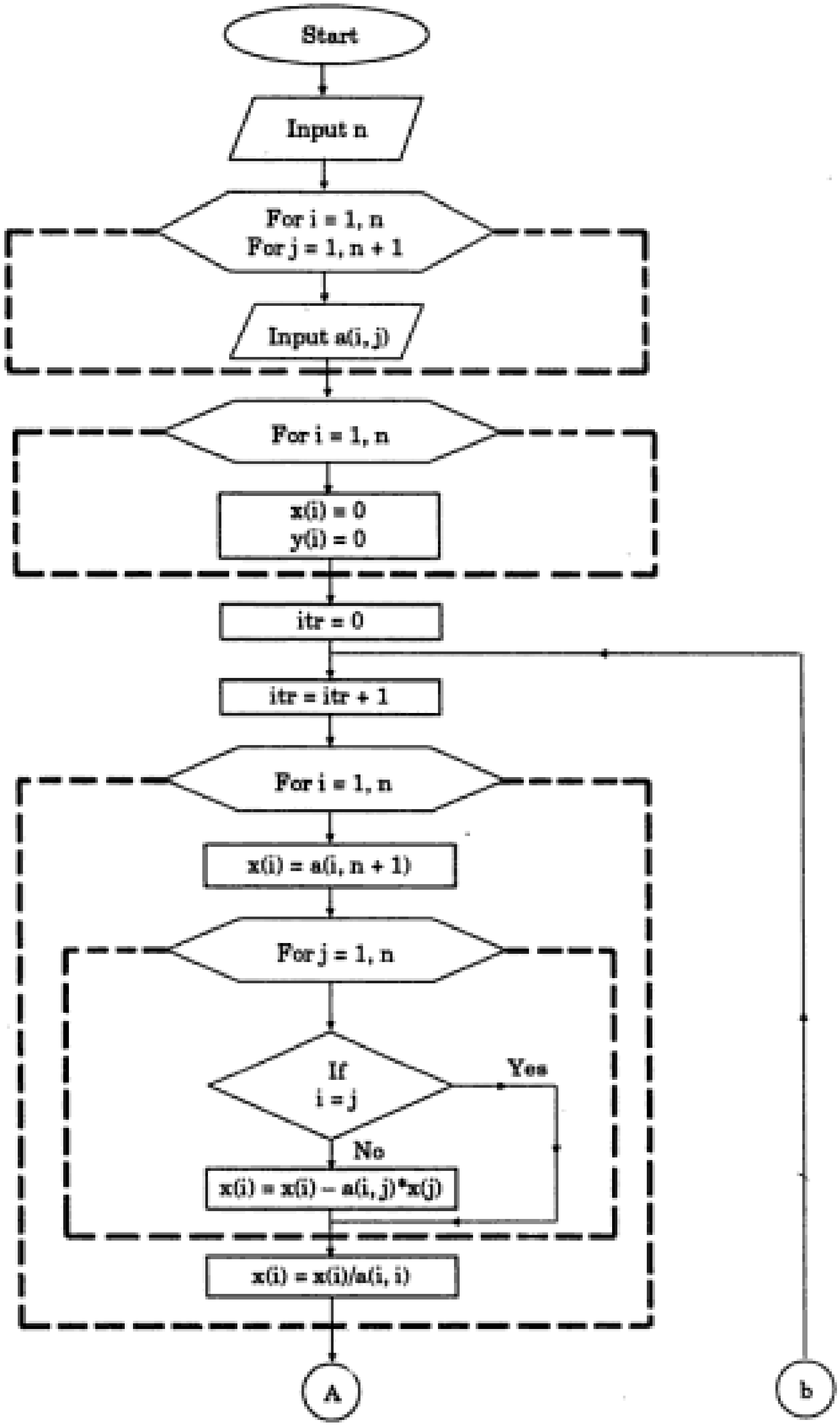
Set x0[i] = x[i]

Repeat i

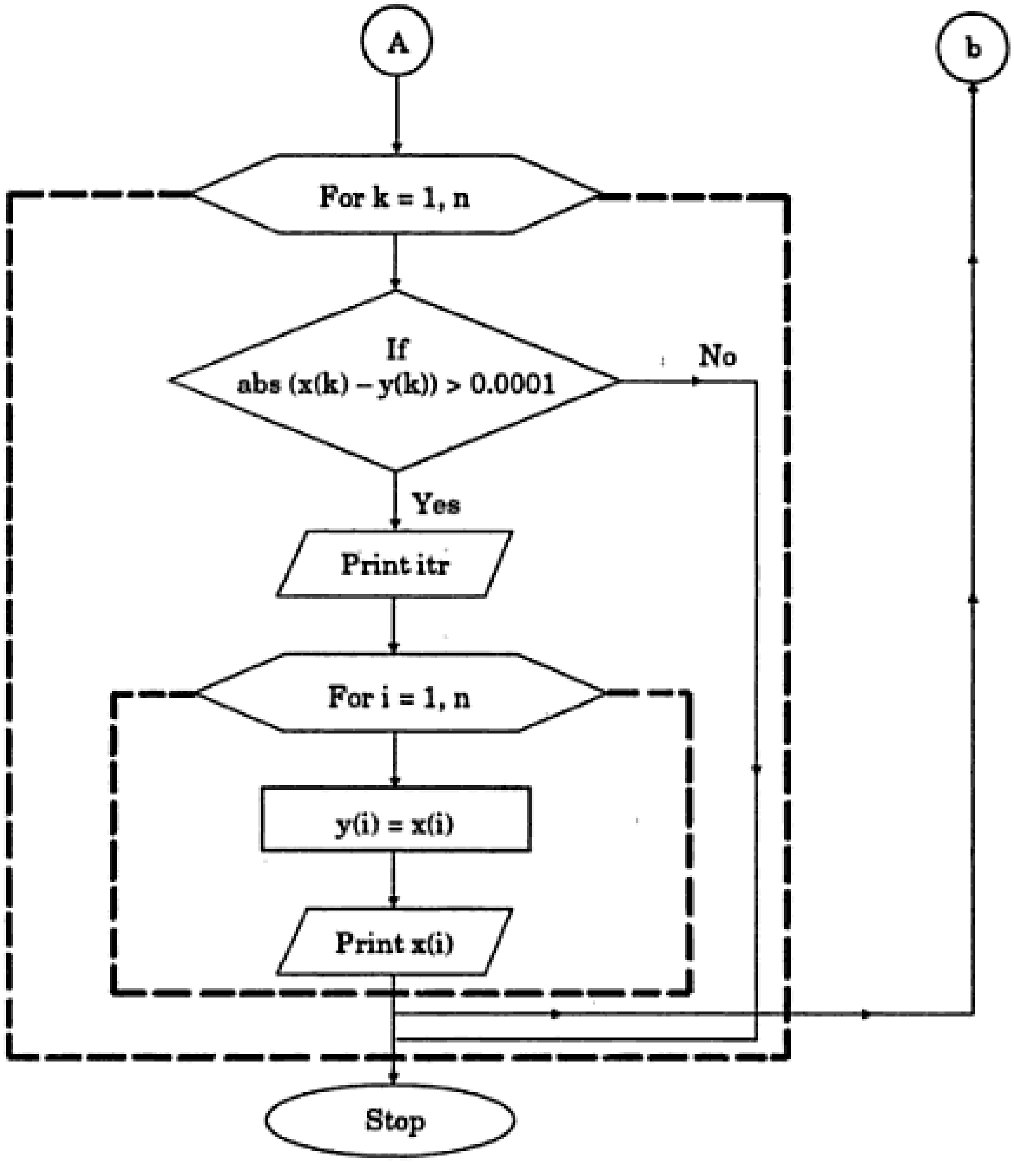
9. If key = 1, then

Goto step 6

Otherwise print results



Start
Input n
For i = 1, n
For j = 1, n + 1
Input a(i, j)
For i = 1, n
x(i) = 0
y(i) = 0
itr = 0
itr = itr + 1
For i = 1, n
x(i) = a(i, n + 1)
For j = 1, n
If
i = j
Yes
No
x(i) = x(i) – a(i, j)*x(j)
x(i) = x(i)/a(i, i)
A
b


A
b
For k = 1, n
If
abs (x(k) – y(k)) > 0.0001
No
Yes
Print itr
For i = 1, n
y(i) = x(i)
Print x(i)
Stop