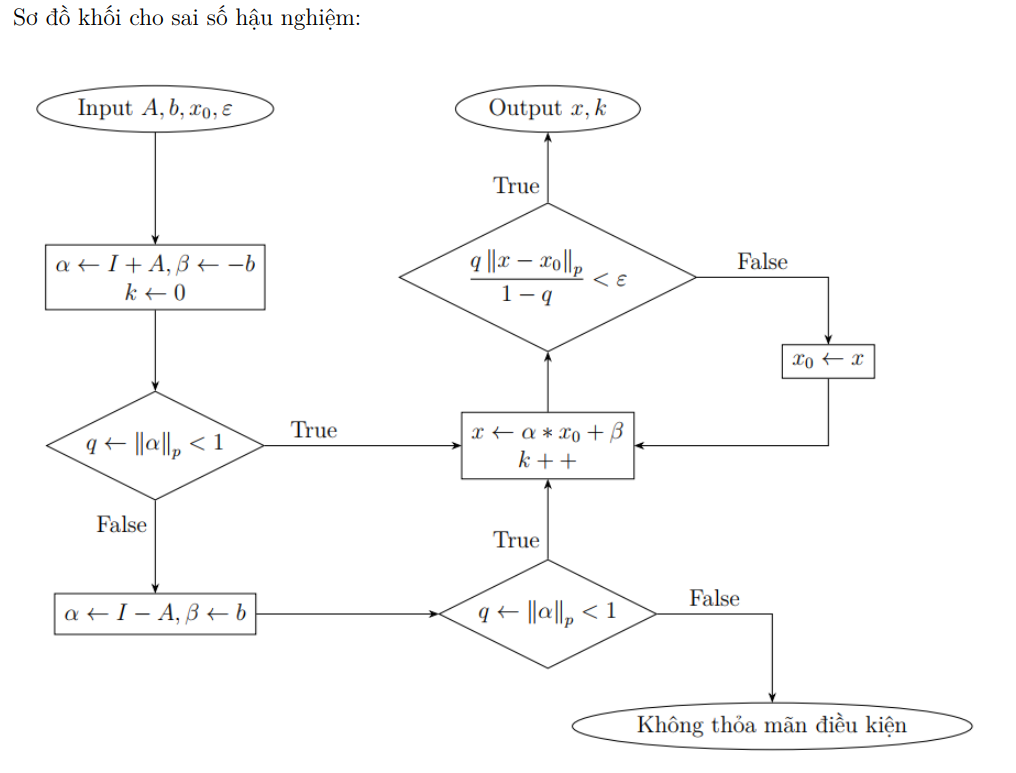
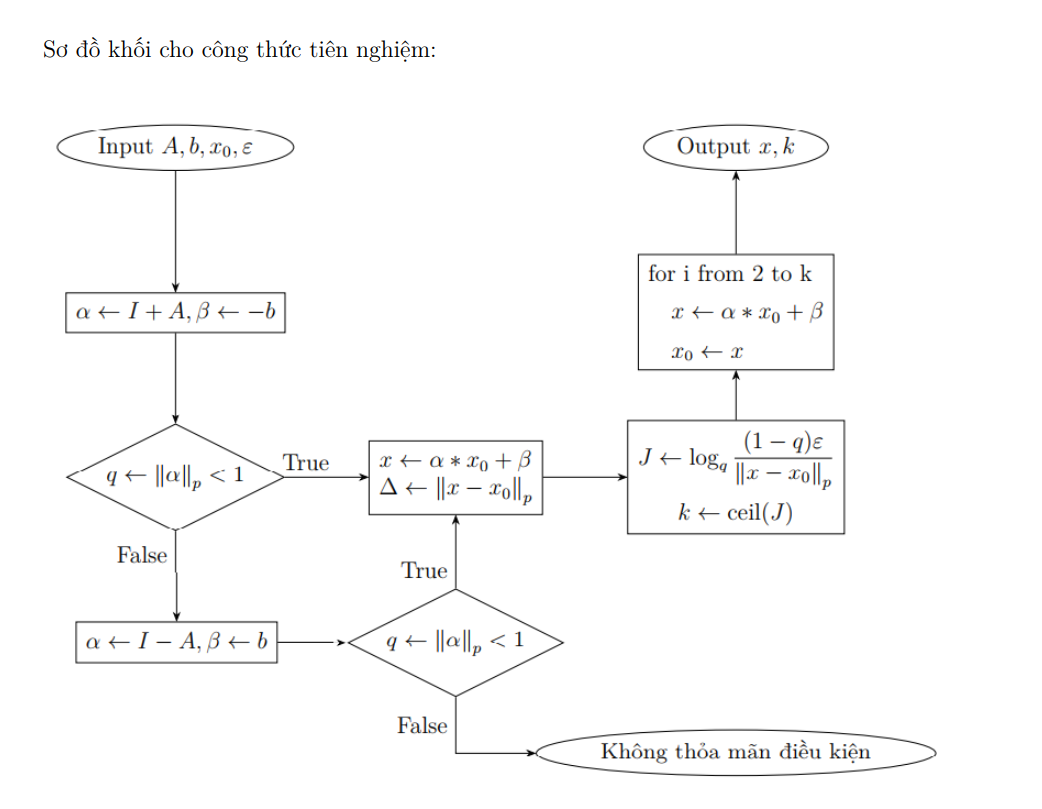
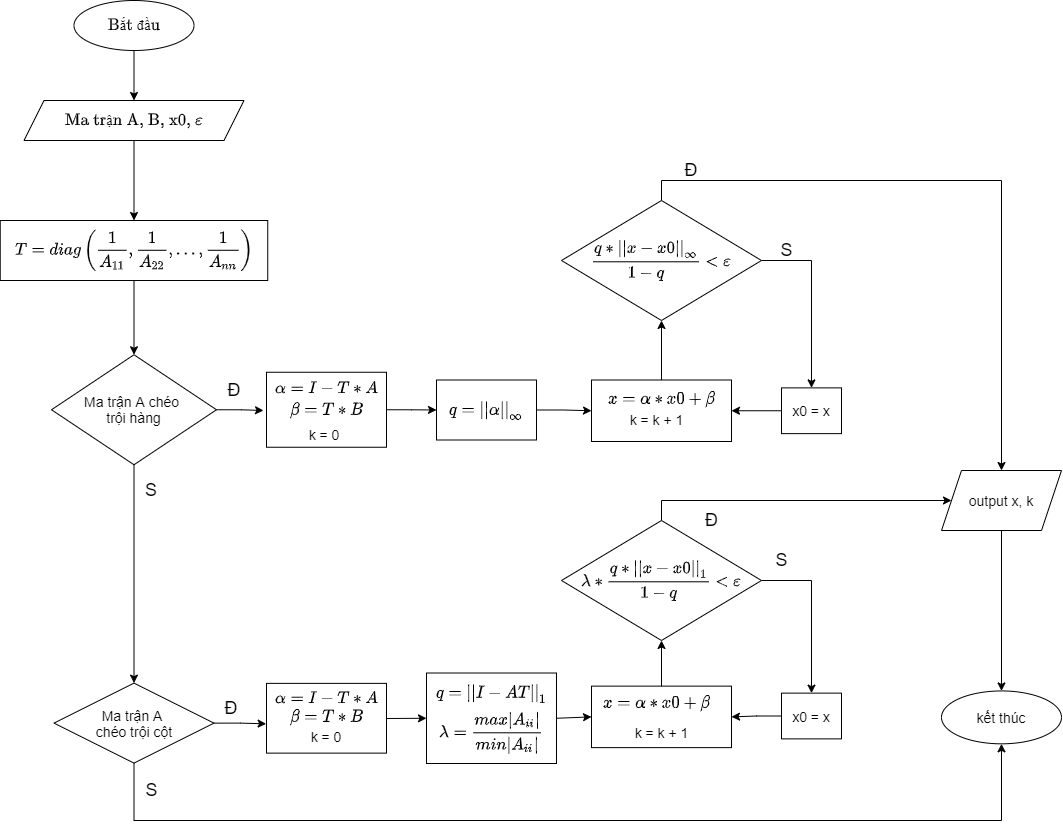
1. Thuật toán tổng quát
2. Lặp đơn hậu nghiệm



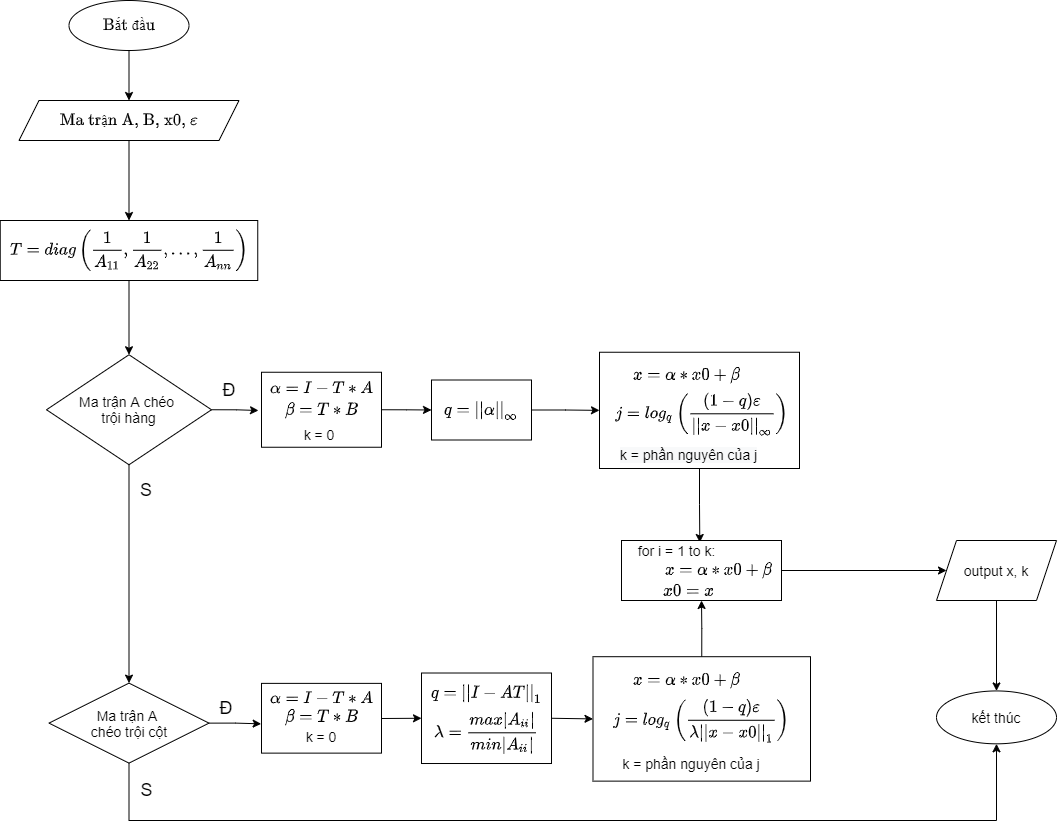
1. Lặp đơn tiên nghiệm



1. Lặp Jacobi hậu nghiệm.



1. Lặp Jacobi tiên nghiệm



1. Thuật toán chi tiết. Giả mã

|  |
| --- |
| 1. Hàm tính chuẩn của ma trận   input: ma trận A, loại chuẩn lựa chọn: type (với type = 1: chuẩn hàng; type = 2: chuẩn cột; type = 3: chuẩn Euclid; type = 4: chuẩn trị riêng  output: giá trị ||A||  function getNorm:  norm = 0  if type = 1:  for i = 1 to A.row:  norm1 = 0  for j = 1 to A.col:  norm1 = norm1 + |Aij|  if norm1 > norm:  norm = norm1  return norm  if type = 2:  for i = 1 to A.col:  norm1 = 0  for j = 1 to A.row:  norm1 = norm1 + |Aij|  if norm1 > norm:  norm = norm1  return norm  if type = 3:  for i = 1 to A.row:  norm1 = 0  for j = 1 to A.col:  norm1 = norm1 + Aij \* Aij  return  if type = 4:  ma trận B = AT \* A  for i = 1 to A.row:  X[i][1] = 1  t0 = 0  t1 = 0  do:  t0 = t1  X = B \* X  s = X[1][1]  for i = 2 to A.row – 1:  if |Xi, 1| > s:  j = i  s = |Xi, 1|  t1 = X[j][0]  for i = 0 to A.row – 1:  X[i][0] = X[i][0] / t1  while (|t1 – t0| > 10-5)  return |

1. **Lặp đơn.**

|  |
| --- |
| 1. Hàm lặp theo phương pháp lặp đơn   input: ma trận A, B, X0, epsi  output: nghiệm X và số lần lặp k hoặc thông báo không chạy được lặp đơn   * 1. Lặp đơn hậu nghiệm   Function singLoopHN:  check = 0  alpha = I – A //I là ma trận đơn vị cùng cỡ với ma trận A  for i = 1 to 4:  if getNorm(alpha, i) < 1:  check = i  beta = B  thoát khỏi vòng lặp for  if check = 0:  alpha = E + A  for i = 1 to 4:  if getNorm(alpha, i) < 1:  check = i  beta = - B  thoát khỏi vòng lặp for  if check = 0:  print “Không thể lặp đơn”  kết thúc chương trình  q = getNorm(alpha, check)  k = 0  X = X0  do:  X0 = X  X = alpha \* X0 + beta  k ++  while q \* getNorm(X – X0, check) > epsi \* (1 – q)  print “nghiệm là: “ X  print “số lần lặp: “ k  kết thúc chương trình   * 1. Lặp đơn tiên nghiệm   Function singLoopTN:  check = 0  alpha = I – A //I là ma trận đơn vị cùng cỡ với ma trận A  for i = 1 to 4:  if getNorm(alpha, i) < 1:  check = i  beta = B  thoát khỏi vòng lặp for  if check = 0:  alpha = E + A  for i = 1 to 4:  if getNorm(alpha, i) < 1:  check = i  beta = - B  thoát khỏi vòng lặp for  if check = 0:  print “Không thể lặp đơn”  kết thúc chương trình  q = getNorm(alpha, check)  X = alpha \* X0 + beta  j = log(epsi \* (1 – q) / getNorm(X – X0, check)) / log(q)  k = phần nguyên của j  for i = 1 to k:  X = alpha \* X0 + beta  X0 = X  print “Nghiệm là: “ X  print “Số lần lặp là: “ k  kết thúc chương trình |

1. **Lặp Jacobi.**

|  |
| --- |
| 1. Hàm kiểm tra tính chéo trội của ma trận A    1. Kiểm tra tính chéo trội hàng   input: Ma trận A  output: true nếu A chéo trội hàng, false nếu A không chéo trội hàng  Function check\_row:  for i = 1 to A.row:  max = A[i][i]  for j = 1 to A.row:  if i j:  max = max - |A[i][j]|  if max <= 0:  return false  return true   * 1. Kiểm tra tính chéo trội cột   input: ma trận A  output: true nếu A chéo trội cột, false nếu A không chéo trội cột  function check\_col:  for i = 1 to A.row:  max = A[i][i]  for j = 1 to A.row:  if i j:  max = max - |A[j][i]|  if max <= 0:  return false  return true   1. Hàm lặp Jacobi   input: ma trận A, B, X0, epsi  output: nghiệm X, số lần lặp k   * 1. Hàm lặp Jacobi hậu nghiệm   Function jacobiLoopHN:  for i = 1 to A.row:  for j = 1 to A.row:  if i = j:  T[i][j] = 1 / A[i][i]  else:  T[i][j] = 0  if check\_row = true:  alpha = I – T \* A  beta = T \* B  k = 0  q = getNorm(alpha, 1)  X = X0  do:  X0 = X  X = alpha \* X0 + beta  k = k + 1  while q \* getNorm(X – X0, 1) < epsi \* (1 – q)  print “nghiệm là: “ X  print “số lần lặp: “ k  kết thúc chương trình  if check\_col = true:  max = |A[1][1]|  min = |A[1][1]|  alpha = I – T\*A  beta = T \* B  k = 0  q = getNorm(I – A\*T, 2)  for i = 2 to A.row:  if |A[i][i]| > max:  max = |A[i][i]|  if |A[i][i]| < min:  min = |A[i][i]|  = max / min  X = X0  do:  X0 = X  X = alpha \* X0 + beta  k = k + 1  while \*q\*getNorm(X – X0, 2) < epsi \* (1 – q)  print “nghiệm là: “ X  print “số lần lặp: “ k  kết thúc chương trình  else:  print “ma trận A không chéo trội, không lặp Jacobi được”  kết thúc chương trình   * 1. Lặp Jacobi theo công thức tiên nghiệm   Function jacobiLoopTN:  for i = 1 to A.row:  for j = 1 to A.row:  if i = j:  T[i][j] = 1 / A[i][i]  else:  T[i][j] = 0  if check\_row = true:  alpha = I – T \* A  beta = T \* B  q = getNorm(alpha, 1)  X = alpha \* X0 + beta  j = log((1 – q) \* epsi / getNorm(X – X0, 1)) / log (q)  k = phần nguyên của j  for i = 1 to k:  X = alpha \* X0 + beta  X0 = X  print “nghiệm là: “ X  print “số lần lặp: “ k  kết thúc chương trình  if check\_col = true:  max = |A[1][1]|  min = |A[1][1]|  alpha = I – T\*A  beta = T \* B  q = getNorm(I – A\*T, 2)  for i = 2 to A.row:  if |A[i][i]| > max:  max = |A[i][i]|  if |A[i][i]| < min:  min = |A[i][i]|  = max / min  X = alpha \* X0 + beta  j = log((1 – q)\*epsi / ( \* getNorm(X – X0, 2)) / log (q)  k = phần nguyên của j  for i = 1 to k:  X = alpha \* X0 + beta  X0 = X  print “nghiệm là: “ X  print “số lần lặp: “ k  kết thúc chương trình  else:  print “ma trận A không chéo trội, không lặp Jacobi được”  kết thúc chương trình |

1. Ưu và nhược điểm của phương pháp
2. Ưu điểm

* Phương pháp lặp đơn giải quyết được sự bất ổn định của nghiệm khi giải hệ bằng phương pháp đúng (Gauss, Gauss-Jordan, Choleski)
* Tối ưu được bộ nhớ
* Dễ cài đặt trên máy tính
* Chi phí xử lí và tốc độ hội tụ nhanh hơn phương pháp tính toán trực tiếp

1. Nhược điểm

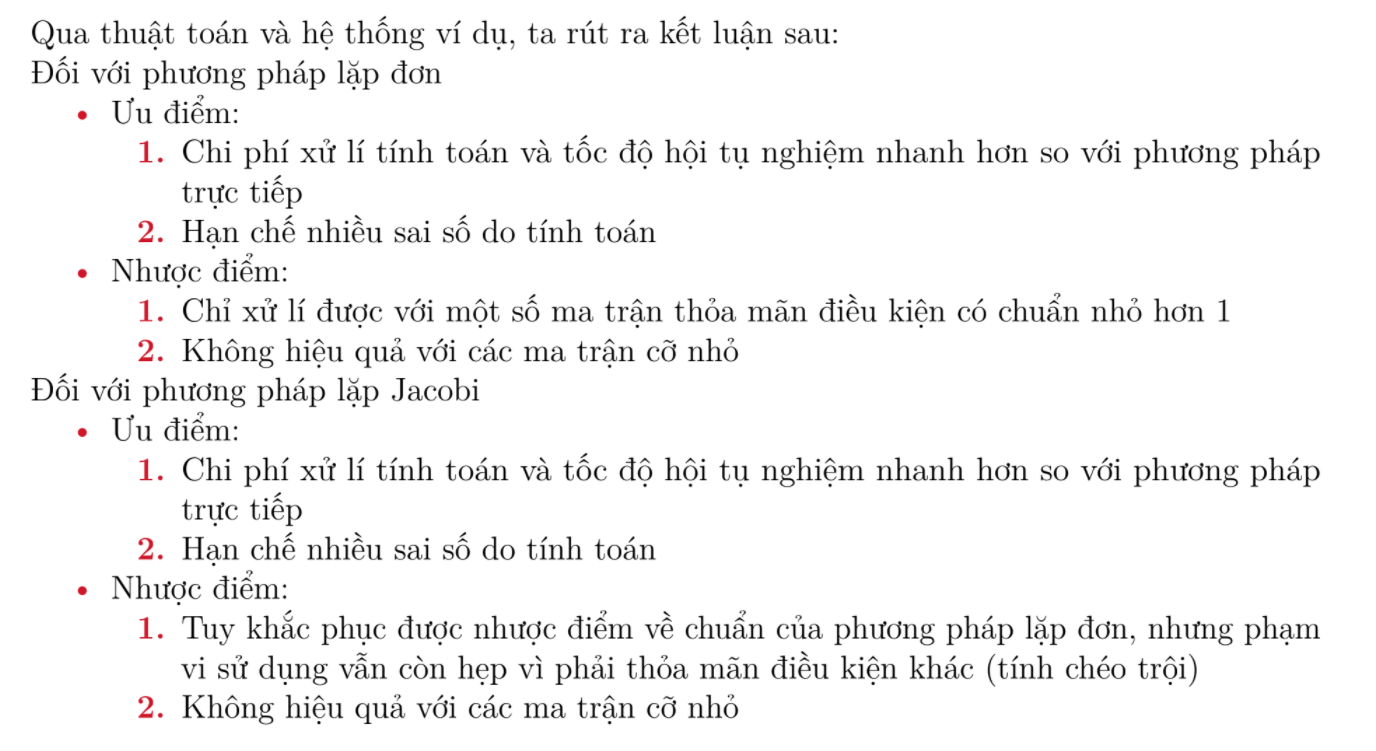
* Phương pháp lặp đơn: chỉ lặp được khi ||I – A|| < 1 hoặc

||I + A|| < 1

* Phương pháp lặp Jacobi: chỉ lặp được khi ma trận A là chéo trội hàng hoặc chéo trội cột
* Lớp phương trình đại số tuyến tính giải được bằng phương pháp lặp đơn và lặp Jacobi là tương đối hẹp

\*Chú ý: Tốc độ hội tụ của phương pháp lặp đơn và lặp Jacobi thì chậm hơn nhiều so với phương pháp lặp Seidel và Gauss Seidel

\* Phương pháp lặp Jacobi chỉ giải được với điều kiện ma trận A là ma trận vuông



1. **Tóm tắt phương pháp.**

Phương pháp chỉ áp dụng được với những ma trận vuông

1. Phương pháp lặp đơn.

Đưa phương trình Ax = B về dạng X = qua 2 cách:

+ Cách 1: = A + I, = - B

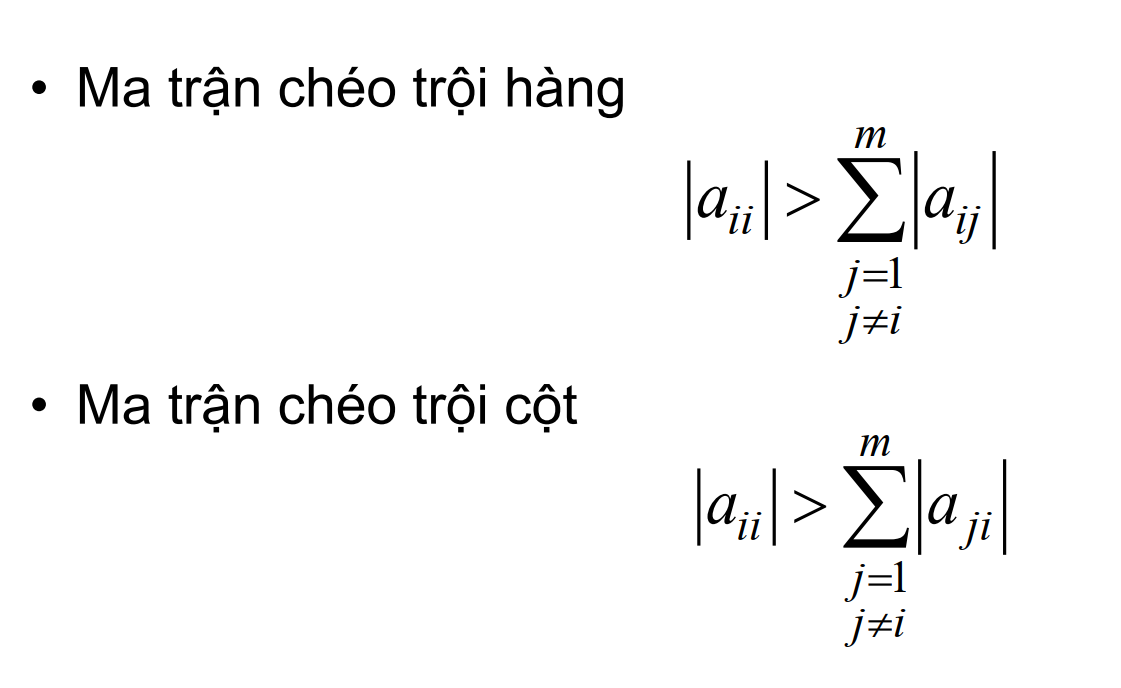
+ Cách 2: = I - A, = B

1. Điều kiện của phương pháp: q = |||| < 1
2. Dãy lặp của phương pháp: xn = xn-1\* +
3. Công thức sai số (điều kiện dừng lặp)

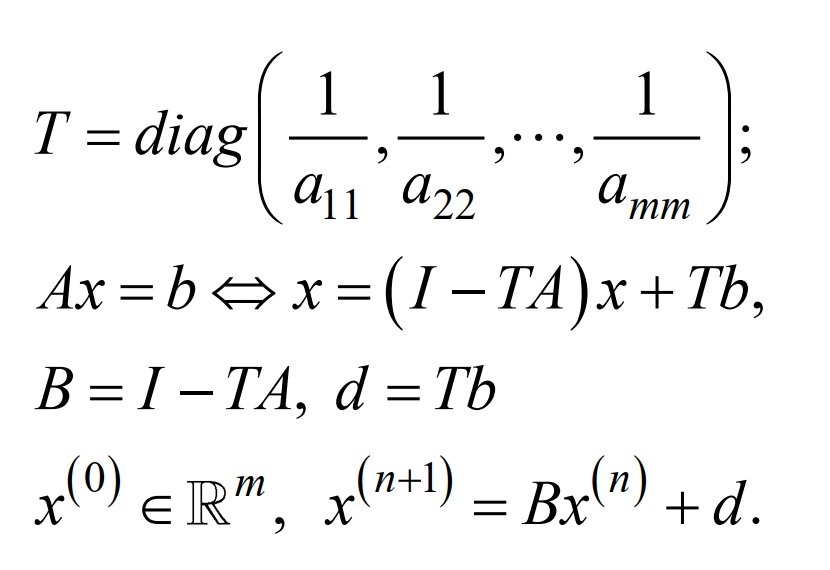
* Công thức hậu nghiệm: ||xn – x\*|| <= \* ||xn – xn-1||
* Công thức tiên nghiệm: ||xn – x\*|| <= \* ||x1 – x0||

1. Phương pháp lặp Jacobi. Dựa trên ý tưởng phương pháp lặp đơn giải quyết các bài toán mà |||| >= 1, và ma trận A là ma trận chéo trội hàng hoặc ma trận chéo trội cột.
2. Điều kiện của phương pháp:

* Ma trận A là ma trận chéo trội hàng hoặc ma trận chéo trội cột



+ Ma trận A là ma trận chéo trội hàng: các chuẩn dùng ở đây là chuẩn hàng

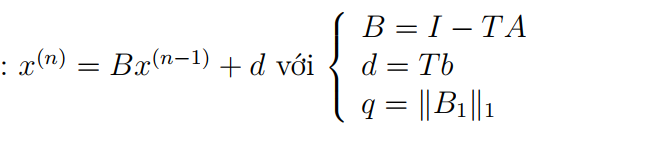


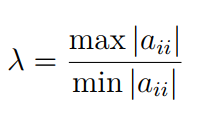
* Công thức sai số: q = ||B||

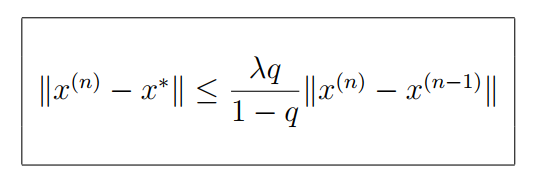
+ Công thức hậu nghiệm: ||xn – x\*|| <= \* ||xn – xn-1||

+ Công thức tiên nghiệm: ||xn – x\*|| <= \* ||x1 – x0||

+ Ma trận A là ma trận chéo trội cột: Các chuẩn dùng ở đây là chuẩn cột



* Công thức sai số:
* 

+ Hậu nghiệm: 

+ Tiên nghiệm:

