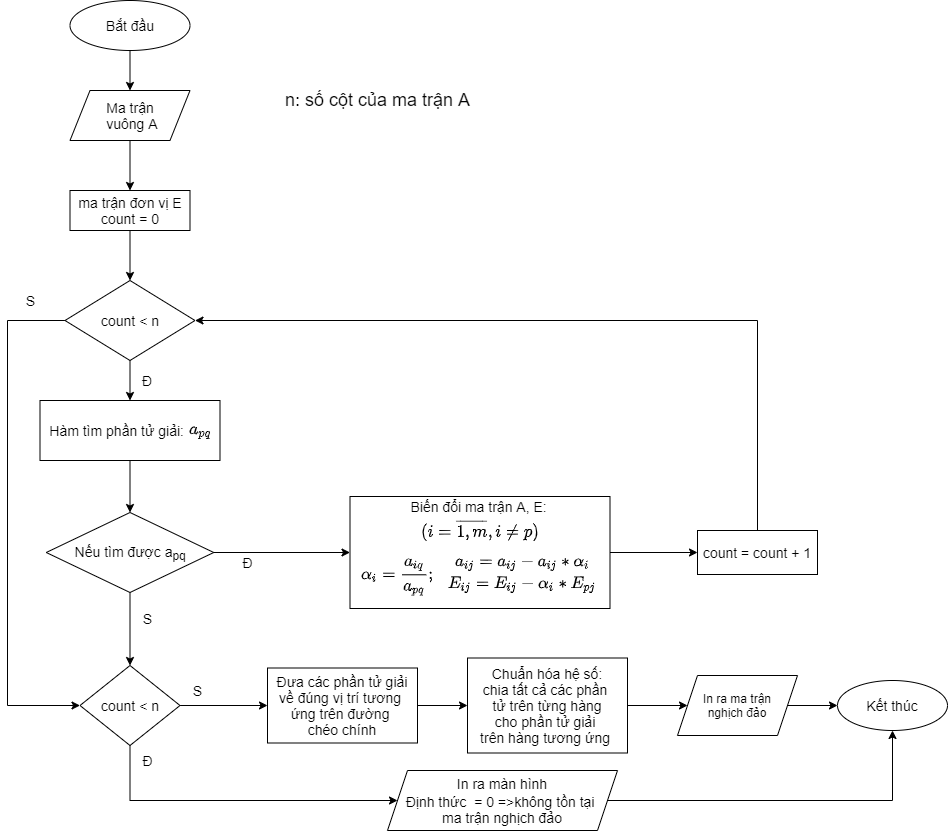
**Các phương pháp tìm đúng ma trận nghịch đảo: Gauss – Jordan; Cholevski; Viền quanh**

1. **Phương pháp Gauss – Jordan**
2. Thuật toán tổng thể



1. Thuật toán chi tiết. (Giả mã)

|  |
| --- |
| 1. Định nghĩa ma trận là một bản ghi gồm có:   + hàng: row  + cột: col  + dữ liệu: mảng 2 chiều a[][]   1. Hàm đổi chỗ hai hàng của ma trận   Input: ma trận A, vị trí hai hàng cần đổi chỗ: x, y  Output: ma trận A sau khi đã được đổi chỗ  Function swap:  m = số hàng ma trận A  for i = 0 to m – 1:  temp = Ax,i  Ax,i = Ay,i  Ay,i = temp   1. Hàm tìm vị trí phần tử giải và lưu vị trí đó vào mảng 1 chiều index[]   Input: ma trận A, mảng chứa vị trí của các phần tử giải: index[],  địa chỉ vị trí hàng giải: p, địa chỉ vị trí cột giải: q  Output: true nếu tìm được vị trí phần tử giải; false nếu không tìm được    // Chú ý: index[j] = i; vị trí phần tử giải ở hàng i cột j  Function index\_ele\_sol:  m = số hàng ma trận A  n = số cột ma trận A  max = 0  for i = 0 to m – 1:  check = 0  for k = 0 to n – 1:  if i = index[k]:  check = 1  End for  if check 1:  for j = 0 to n – 1:  if index[j] -1:  if |Aij| = 1:  p = i  q = j  index[q] = p  return true  else:  if |Aij| > max:  max = |Aij|  p = i  q = j  if max != 0:  index[q] = p  return true  else:  return false   1. Hàm ma trận nghịch đảo sử dụng phương pháp Gauss – Jordan   Input: Ma trận đầu vào A  Output: Kết luận và in ra ma trận nghịch đảo nếu có  Function gauss\_jordan\_method:  m = số hàng ma trận A  n = số cột ma trận A  count = 0  if m n:  print “ma trận không vuông => không tồn tại nghịch đảo”  else:  //Khởi tạo ma trận đơn vị E  for i = 0 to n – 1:  for j = 0 to n – 1:  if i = j:  Eij = 1  else:  Eij = 0  for i = 0 to n – 1:  index[i] = -1  for k = 0 to n - 1:  if index\_ele\_sol(A, index, p, q) = true:  count = count + 1  for i = 0 to n – 1:  if i p:  coeff = Aiq / Apq  for j = 0 to n – 1:  Aij = Aij – coeff \* Apj  Eij = Eij – coeff \* Epj  else:  End for  if count < n:  print “Định thức ma trận bằng 0 => Không tồn tại ma trận nghịch đảo”  // Chuẩn hóa hệ số ma trận  else:  for i = 0 to n – 1:  if Aii = 0:  for h = i + 1 to n – 1:  if Ah,i 0:  swap(A, h, i)  swap(E, h, i)  End for  for i = 0 to n – 1:  for j = 0 to n – 1:  Eij = Eij / Aii  print “ma trận nghịch đảo là: “  for i = 0 to n – 1:  for j = 0 to n – 1:  print Eij |

1. Ưu và nhược điểm của phương pháp Gauss – Jordan
2. Ưu điểm

* Tính toán ma trận nghịch đảo với độ chính xác cao do đã giảm được sai số khi chia cho số gần 0
* Dễ lập trình trên máy tính
* Có thể áp dụng để tính định thức của ma trận

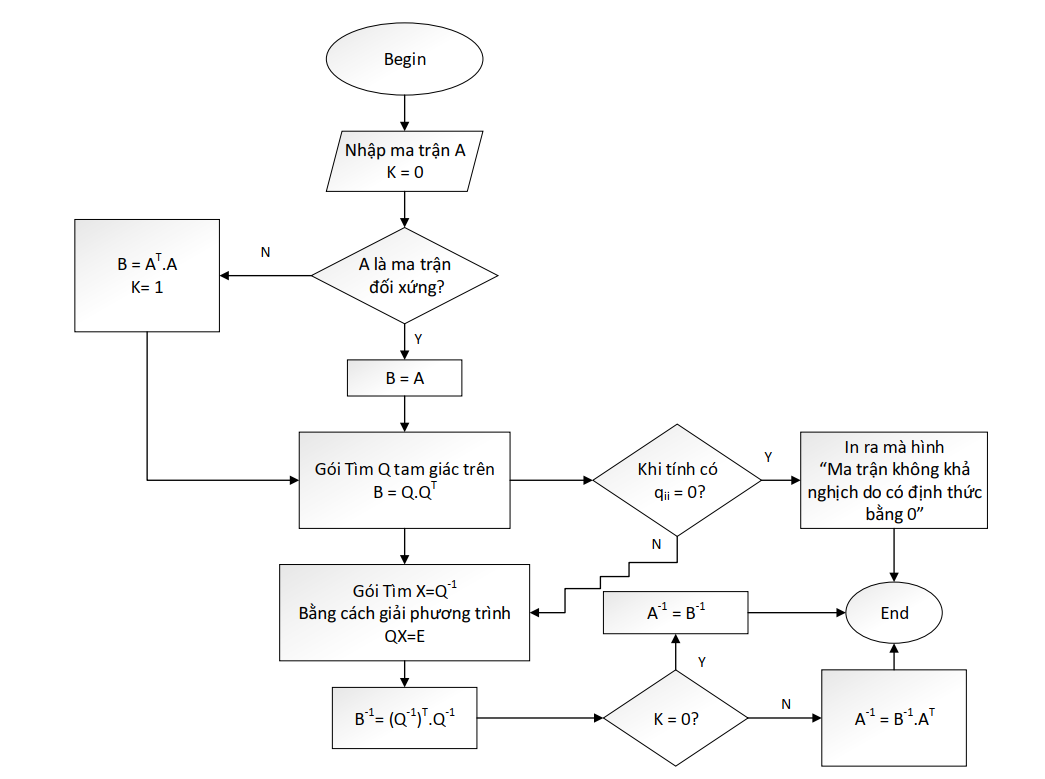
1. Nhược điểm.

* Độ phức tạp thuật toán lớn do mỗi lần lặp ta đều phải đi tìm phần tử giải sau đó mới biến đổi ma trận bổ sung A|E
* Đối với những ma trận cỡ lớn thì tìm ma trận nghịch đảo theo pp GJ sẽ lâu và nên dùng các pp tính gần đúng để tìm sẽ nhanh hơn.

1. **Tóm tắt phương pháp Gauss – Jordan tìm ma trận nghịch đảo**
2. Điều kiện của phương pháp

* Ma trận A phải là ma trận vuông

1. Ta lập ma trận mở rộng A|E, E là ma trận đơn vị
2. Ma trận A tồn tại ma trận nghịch đảo khi mà định thức của ma trận A là khác 0 ⬄ Ta tìm được đủ n phần tử giải
3. Phần tử giải apq: ưu tiên chọn phần tử có giá trị = 1 hoặc = - 1, nếu không có thì sẽ ưu tiên tìm phần tử có trị tuyệt đối ≠ 0 lớn nhất. Sau khi biến đổi ma trận A|E theo pp Gauss – Jordan thì trên mỗi hàng, mỗi cột của ma trận A sẽ chỉ còn 1 phần tử khác 0 duy nhất và đó chính là phần tử giải ở hàng giải và cột giải đó.
4. Các phần tử giải có thể không thuộc đường chéo chính, ta cần hoán vị hàng để đưa chúng về đúng vị trí đường chéo chính.
5. Sau đó, chuẩn hóa ma trận để đưa ma trận A về dạng đơn vị bằng cách chia tất cả các phần tử trên hàng với phần tử trên đường chéo chính ở hàng tương ứng.
6. Cuối cùng từ ma trận mở rộng A|E => ta thu được ma trận E|B và ma trận B chính là ma trận nghịch đảo của ma trận A
7. **Phương pháp Cholevski**
8. Thuật toán tổng thể



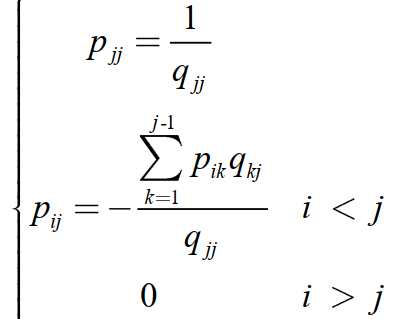
1. Thuật toán chi tiết
2. Ưu và nhược điểm của phương pháp
3. Ưu điểm

* Độ phức tạp thuật thấp, tốc độ tìm ra ma trận nghịch đảo nhanh
* Dễ lập trình tính toán trên máy tính

1. Nhược điểm

* Đầu vào của phương pháp: cần là ma trận A đối xứng hoặc nếu không đối xứng thì cần nhân AT với A để áp dụng pp với ma trận ATA
* Sai số trong tính toán có thể lớn do có thể sẽ xuất hiện chia cho số gần 0

1. Tóm tắt phương pháp
2. Điều kiện của pp là ma trận phải là ma trận đối xứng, nếu không đối xứng thì cần nhân thêm ma trận chuyển vị sau đó áp dụng pp với ma trận mới
3. Ta có: A = QTQ => tìm ma trận tam giác trên Q
4. Nếu trong quá trình tính toán xuất hiện qii = 0 thì kết luận định thức ma trận = 0 => không tồn tại ma trận nghịch đảo
5. Tìm ma trận nghịch đảo của Q: bằng cách giải pt QQ-1 = E. Theo công thức: Q-1 = [pij]



1. Sau đó tính ma trận nghịch đảo: B-1 = (Q-1)T \* Q-1
2. Nếu A là ma trận đối xứng thì ma trận nghịch đảo là: A-1 = B-1. Nếu A không là ma trận đối xứng thì ma trận nghịch đảo:

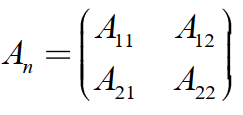
A-1 = B-1 \* AT

1. **Phương pháp Viền quanh**

Chú ý: Phương pháp viền quanh chỉ áp dụng được cho những ma trận vuông có các định thức con chính ≠ 0. Để chủ động trong phương pháp và giảm độ phức tạp => ta áp dụng phương pháp với ma trận B = ATA vì ma trận B là ma trận đối xứng xác định dương có tất cả các định thức con chính ≠ 0.

Sau khi tìm được ma trận nghịch đảo B-1 => ma trận nghịch đảo của ma trận A sẽ là: A-1 = B-1AT. Bên dưới trình bày ma trận A là ma trận đầu vào của phương pháp ⬄ A = B

1. Phương pháp viền quanh



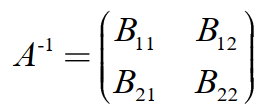
Với A11 = An-1, n-1;

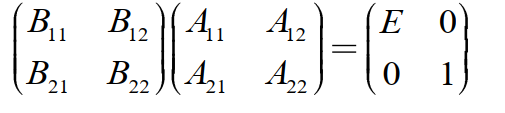
A12 = An-1,1

A21 = A1,n-1

A22 = Ann

* Ma trận nghịch đảo:



Giải hệ phương trình: A \* A-1 = E ⬄ 

Giải hệ ta được:







