```
import numpy as np

aug = np.array([[8, 3, -3, 14],
                [-2, -8, 5, 5],
                [3, 5, 10, -8]], dtype=float)
```

Gauss Seidel Method

```
def is_dominant(A):
  isDominant = True
  diag = []
  ndiag = []
  #cari elemen diagonal
  for i in range(len(A)):
    diag.append(abs(A[i][i]))
  #cari jumlah elemen non diagonal
  for i in range(len(A)):
    nsum = 0
    for k in range(len(A[i]) - 1):
      nsum += abs(A[i][k])
    nsum -= diag[i]
    ndiag.append(nsum)
  #membandingkan diagonal dengan non diagonal
  for i in range(len(diag)):
    if diag[i] <= ndiag[i]:
      isDominant = False
  return isDominant
```

```
def gauss_seidel(x, eps):
  erd = 999
  x_old = np.array(x)
  while erd > eps:
    x[0] = (14 - (3 * x[1]) + (3 * x[2])) / 8
    x[1] = (-5 - (2 * x[0]) + (5 * x[2])) / 8
    x[2] = (-8 - (3 * x[0]) - (5 * x[1])) / 10
    x_new = np.array([x[0], x[1], x[2]])
    erd = np.sqrt(np.dot(x_new-x_old, x_new-x_old))
    x_old = x_new
  return x
```

```
if is_dominant(aug):
  x = [1, 2, 3]
  eps = 0.01
  xr = gauss_seidel(x, eps)
  print("x1 = %.3f; x2 = %.3f; x3 = %.3f"%(xr[0], xr[1], xr[2]))
else:
  print("Cannot be solved using Gauss Seidel method.")
```

```
x1 = 2.087; x2 = -1.554; x3 = -0.649
```

Gauss Jordan Elimination

```
def forward_elim(A):
  for i in range(len(A) - 1):
    a = A[i][i]
    for k in range(len(A[i])):
      A[i][k] /= a
    a = A[i][i]
    for j in range(i+1, len(A)):
      lb = A[j][i] / a
      for k in range(len(A[i])):
        A[j][k] -= (lb * A[i][k])
  return A
```

```
def back_elim(A):
  for i in range(len(A) - 1, 0, -1):
    a = A[i][i]
    for k in range(len(A[i])):
      A[i][k] /= a
    a = A[i][i]
    for j in range(i-1, -1, -1):
      lb = A[j][i] / a
      for k in range(len(A[i])):
        A[j][k] -= (lb * A[i][k])
  return A
```

```
A = forward_elim(aug)
B = back_elim(A)
print("x1 = %.3f; x2 = %.3f; x3 = %.3f"%(B[0][3], B[1][3], B[2][3]))
```

```
x1 = 2.089; x2 = -1.553; x3 = -0.650
```