```python
#  Let the three system of linear equations be:
# 1. 3x1 - x2 + x3 =2
# 2. -5x1 + 4x2 + 2x3 = 4
# 3. 3x1 - 2x2 + 6x3 = -7

# As per Gauss-Seidel iterative method, let's assume initial values of
[x1, x2, x3] = [0, 0, 0]
# As per the theorem, x^(k + 1) = inv(L)[b - Ux^k] for a given
equation Ax = b, L = strictly Lower triangular matrix U = strictly
upper triangular matrix

from numpy import array, dot
import numpy as np
from numpy.linalg import inv

# Number of iterations to be considered
n=8

A = array([[3.0,-1.0,1.0],[-5.0,4.0,2.0],[3.0,-2.0,6.0]])
B = array([2.0,4.0,-7.0])
initial = array([0.0,0.0,0.0])

U = np.triu(A,1)
L = np.tril(A,0)
inverse_L = inv(L)

#x^(k + 1) = inv(L)[b - Ux^k]

for i in range(n):
    Neg_Ux_plusB = -dot(U,initial)+B
    initial = dot(inverse_L, Neg_Ux_plusB)
    print(i+1,np.round(initial,7))
```

```
1 [ 0.6666667  1.8333333 -0.8888889]
2 [ 1.5740741  3.412037  -0.816358 ]
3 [ 2.0761317  4.0033436 -0.8702846]
4 [ 2.2912094  4.2991541 -0.87922  ]
5 [ 2.3927914  4.4305992 -0.8861959]
6 [ 2.4389317  4.4917626 -0.8888783]
7 [ 2.4602136  4.5197062 -0.8902048]
8 [ 2.4699703  4.5325653 -0.8907967]
```

```python
# Second method of solving the above set of equations using
numpy.linalg.solve

import numpy as np
from scipy.linalg import solve


def gauss_seidel(A, B, initial2, n2):
```

```python
    L = np.tril(A)
    U = A - L
    for i in range(n):
        initial2 = np.dot(np.linalg.inv(L), B - np.dot(U, initial2))
        print(str(i+1))
        print(initial2)
    return initial2


# Number of iterations to be considered
n2 = 8

A = np.array([[3.0,-1.0,1.0], [-5.0,4.0,2.0], [3.0,-2.0,6.0]])
B = [2.0,4.0,-7.0]
initial2 = [0.0, 0.0, 0.0]



print(gauss_seidel(A, B, initial2, n2))
print(solve(A, B))
```

```
1
[ 0.66666667  1.83333333 -0.88888889]
2
[ 1.57407407  3.41203704 -0.81635802]
3
[ 2.07613169  4.00334362 -0.87028464]
4
[ 2.29120942  4.29915409 -0.87922001]
5
[ 2.39279137  4.43059922 -0.88619595]
6
[ 2.43893172  4.49176262 -0.88887832]
7
[ 2.46021365  4.51970622 -0.89020475]
8
[ 2.46997032  4.53256528 -0.89079674]
[ 2.46997032  4.53256528 -0.89079674]
[ 2.47826087  4.54347826 -0.89130435]
```

By comparing the above two methods, we can conclude that the values of [x1, x2, x3] are close in both the cases.