[509]:
```python
from sympy import *
import numpy as np
import matplotlib.pyplot as plt
import matplotlib.patches as mpatches
```

[510]:
```python
r = symbols('r')
```

[511]:
```python
def gauss_seidel(iter_, a, x, y, show_steps=False):

    def init_b(i, j) :
        if a[i, j] >= 0 :
            return a[i, j]
        return 0
    def init_c(i, j) :
        if a[i, j] < 0 :
            return -a[i, j]
        return 0
    b = Matrix(*shape(a), init_b)
    c = Matrix(*shape(a), init_c)

    l_b = b.lower_triangular()
    d_b = l_b.upper_triangular()
    l_b = l_b.lower_triangular(-1)
    u_b = b.upper_triangular(1)

    inv = (l_b + d_b).inv()

    M_gs = Matrix([[- inv * u_b, inv * c],
                   [inv * c, - inv * u_b]])

    C_gs = Matrix([inv * Matrix(y[:len(y) // 2]), inv * Matrix(y[len(y) // 2:
↪])])

    for i in range(iter_) :
        if show_steps :
            display(x)
            print(f'x ({i})')
        x = M_gs * x + C_gs
    return x
```

[515]:
```python
# gauss-seidel solving first example
# run this cell with show_steps=True to see each iteration
a = Matrix([[1, -1], [1, 3]])
x = zeros(4, 1)
y = Matrix([r, 4 + r, 2 - r,  7 - 2 * r])
gs_solution = gauss_seidel(5, a, x, y, show_steps=False)
```

[516]:
```python
def jacobian(iter_, a, x, y, show_steps=False):

    def init_b(i, j) :
        if a[i, j] >= 0 :
            return a[i, j]
        return 0
    def init_c(i, j) :
        if a[i, j] < 0 :
            return -a[i, j]
        return 0

    b = Matrix(*shape(a), init_b)
    c = Matrix(*shape(a), init_c)

    l_b = b.lower_triangular()
    d_b = l_b.upper_triangular()
    l_b = l_b.lower_triangular(-1)
    u_b = b.upper_triangular(1)

    inv = d_b.inv()

    M_gs = Matrix([[- inv* l_b, inv * c],
                   [inv * c, - inv * l_b]])

    C_gs = Matrix([inv * Matrix(y[:len(y) // 2]), inv * Matrix(y[len(y) // 2:
↪])])
    for i in range(iter_) :
        if show_steps :
            display(x)
            print(f'x ({i})')
        x = M_gs * x + C_gs
    return x
```

[517]:
```python
# jacobian solving first example,
# run this cell with show_steps=True to see each iteration
a = Matrix([[1, -1], [1, 3]])
x = zeros(4, 1)
y = Matrix([r, 4 + r, 2 - r,  7 - 2 * r])
j_solution = jacobian(6, a, x, y, show_steps=False)
```

[518]:
```python
def plot_fuzzy_number(x1, x2, *args) :
    def line_x_y(u1, u2) :
        x = np.linspace(0, 1, 30)
        return list(u1[0] + u1[1] * x) + list(u2[0] + u2[1] * x), list(x) +␣
↪list(x)
    ax = plt.gca()
    ax.plot(*line_x_y(x1, x2), *args)
```

```
def plot_fuzzy_solution(x, *args) :
    def get_coef(x) :
        b = x.subs(r, 0)
        a = x.subs(r, 1) - b
        return b, a

    n = len(x) // 2
    for i in range(n) :
        x1 = get_coef(x[i])
        x2 = get_coef(x[n + i])
        plot_fuzzy_number(x1, x2, *args)
```

[ ]:
```
# comparing results in the first example
real_solution = Matrix([[1.375 + 0.625 * r],
           [0.875 + 0.125 * r],
           [2.875 - 0.875 * r],
           [1.375 - 0.375 * r]])

plot_fuzzy_solution(j_solution, 'r+')
plot_fuzzy_solution(gs_solution, 'go')
plot_fuzzy_solution(real_solution, 'b')
handles = []
handles.append(mpatches.Patch(color='red', label='jacobian - 5 iterations'))
handles.append(mpatches.Patch(color='green', label='guass - 4 iterations'))
handles.append(mpatches.Patch(color='blue', label='solution'))
ax = plt.gca()
ax.legend(handles=[*handles], bbox_to_anchor=(1.05, 1),
          loc='best', borderaxespad=0.)
```

[ ]:
```
# plotting figure in example 2
real_solution = Matrix([[-2.31 + 3.62 * r],
           [-0.62 - 0.77 * r],
           [1.08 - 2.15 * r],
           [4.69 - 3.38  * r],
           [-1.62 + 0.23 * r],
           [-2.92 + 1.85 * r]])

plot_fuzzy_solution(real_solution, 'b')
```

[ ]:
```
# comparing iterative methods in the third example
real_solution = Matrix([[0.1399 * r - 0.4125],
            [0.2894 * r + 0.9125],
            [-0.1897 * r - 0.6969],
            [-0.3217 * r + 0.0351],
            [0.0970 * r + 1.1076],
            [- 0.1513 * r - 0.7353]])
```

```
a = Matrix([[4, 1, -1], [-1, 3, 1], [2, 1, 3]])
x = zeros(6, 1)
y = Matrix([r, 2 + r, -2, 2 - r, 3, - 1 - r])
j_solution = jacobian(10, a, x, y, show_steps=False)
gs_solution = gauss_seidel(4, a, x, y, show_steps=False)
plot_fuzzy_solution(j_solution, 'r+')
plot_fuzzy_solution(gs_solution, 'go')
plot_fuzzy_solution(real_solution, 'b')
handles = []
handles.append(mpatches.Patch(color='red', label='jacobian - 10 iterations'))
handles.append(mpatches.Patch(color='green', label='guass - 4 iterations'))
handles.append(mpatches.Patch(color='blue', label='solution'))
ax = plt.gca()
ax.legend(handles=[*handles], bbox_to_anchor=(1.05, 1),
          loc='best', borderaxespad=0.)
```