Warunki startowe Programu :

In [ ]:
```
using Plots , LinearAlgebra
max_i=10000
epsilon = 1e-10
matrix = [1 2 3; 2 4 5; 3 5 -1]
```

```
3×3 Matrix{Int64}:
 1  2   3
 2  4   5
 3  5  -1
```

Metoda **Potęgowa**

Opis Algorytmu

a. Inicjalizacja losowego wektora `v`.

b. Normalizacja wektora `v`.

c. Iteracje do maksymalnej liczby iteracji (`max_i`).

d. Obliczenie nowego wektora `v_new` przez pomnożenie macierzy `matrix_c` przez wektor `v`.

e. Obliczenie wartości własnej (`eigen`) jako iloczynu skalarnego wektorów `v` i `v_new`.

f. Normalizacja wektora `v_new`.

g. Sprawdzenie warunku zbieżności. Przerwanie iteracji, jeśli warunek spełniony.

h. Aktualizacja macierzy `matrix_c` odejmując wartość własną pomnożoną przez zewnętrzny iloczyn wektorów (`v * v'`).

Jest to implementacja metody **Potęgowej** w wersji pozwalającej obliczyć każdy wektor własny wraz z wartością, po znalezieniu jednej wartości własnej metoda odejmuje składnik dominujący `matrix_c -= eigen * (v * v')` następnie ponownie aplikuje metodę potęgową na zdeflowanej macierzy w celu znalezienia kolejnej wartości własnej. Ten proces jest powtarzany, aż do uzyskania wszystkich wartości własnych.

In [ ]:
```
matrix_c = copy(matrix)
function power(matrix_c,max_i, epsilon)
    n = size(matrix_c, 1)
    for _ in 1:n
        v = randn(n)
        v /= norm(v, 2)
        eigen = 0.0
        for i in 1:max_i
            v_new = matrix_c * v
            eigen = dot(v, v_new)
            v = v_new / norm(v_new, 2)
            if norm(matrix_c * v - eigen * v, 2) < epsilon
                println("Liczba iteracji po których osiągnięto dokładność ",i)
                break
```

```
            end
        end
        matrix_c -= eigen * (v * v')
        println("Wartość własna: ",eigen,"\nDla wektora ",v)
    end
end
power(matrix_c,max_i,epsilon)
```

```
Liczba iteracji po których osiągnięto dokładność 40
Wartość własna: 8.548512853222785
Dla wektora [0.4059533322650147, 0.7509847864456235, 0.520791457831397]
Liczba iteracji po których osiągnięto dokładność 5
Wartość własna: -4.574087225857106
Dla wektora [-0.30580263397330276, -0.4253864432242941, 0.8517811473471192]
Liczba iteracji po których osiągnięto dokładność 2
Wartość własna: 0.025574372634318235
Dla wektora [0.8612123090535558, -0.5050427945340213, 0.05696608134591154]
```

Metoda **Rayleigha**

Opis Algorytmu

a. Inicjalizacja wektora `v` jako wektora jednostkowego.

b. Iteracje do maksymalnej liczby iteracji (`max_i`).

c. Obliczenie wartości własnej (`eigen`) jako iloczynu skalarnego wektora `v` i macierzy `matrix` pomnożonej przez wektor `v` .

d. Aktualizacja wektora własnego (`v`) poprzez pomnożenie macierzy `matrix` przez wektor `v` .

e. Normalizacja wektora `v` .

f. Sprawdzenie warunku zbieżności. Przerwanie iteracji, jeśli warunek spełniony.

In [ ]:

```
matrix_c = copy(matrix)
function rayleigh(matrix, max_i, epsilon)
    n = size(matrix, 1)
    eigen = 0.0
    v = ones(n) / sqrt(n)
    for i in 1:max_i
        eigen = dot(v, matrix * v)
        v = matrix * v
        v /= norm(v)
        if norm(matrix * v - eigen * v) < epsilon
            println("Liczba iteracji po których osiągnięto dokładność ", i)
            break
        end
    end
    println("Wartość własna: ", eigen, "\nDla wektora ", v)
end
rayleigh(matrix_c, max_i, epsilon)
```

```
Liczba iteracji po których osiągnięto dokładność 37
Wartość własna: 8.548512853222787
Dla wektora [0.4059533322650253, 0.7509847864456384, 0.5207914578313675]
```

Metoda QR

a. Inicjalizacja macierzy ortogonalnej (V) jako macierzy jednostkowej.

b. Iteracje do maksymalnej liczby iteracji (`max_i`).

c. Obliczenie rozkładu QR macierzy `matrix_c` , gdzie (Q) to macierz ortogonalna, a (R) to macierz trójkątna górna.

d. Aktualizacja macierzy `matrix_c` jako iloczyn (R i Q).

e. Akumulacja macierzy ortogonalnej (V) przez mnożenie przez (Q).

f. Sprawdzenie warunku zbieżności. Przerwanie iteracji, jeśli warunek spełniony.

In [ ]:
```
matrix_c = copy(matrix)
function qr_method(matrix_c, max_i, epsilon)
    n = size(matrix_c, 1)
    V = I(n)
    for i in 1:max_i
        Q, R = qr(matrix_c)
        matrix_c = R * Q
        V *= Q
        if norm(matrix_c - diagm(diag(matrix_c))) < epsilon
            println("Liczba iteracji po których osiągnięto dokładność ",i)
            break
        end
    end
    eigenvalues = diag(matrix_c)
    eigenvectors = V
    return eigenvalues, eigenvectors
end
eigenvalues, eigenvectors = qr_method(matrix_c,max_i,epsilon)

for i in 1:size(matrix_c,1)
    println("Wartość własna: ",eigenvalues[i],"\nDla wektora ",eigenvectors[:,i]
end
```

```
Liczba iteracji po których osiągnięto dokładność 42
Wartość własna: 8.54851285322279
Dla wektora [0.40595333226396413, 0.7509847864441626, 0.5207914578343267]
Wartość własna: -4.574087225857102
Dla wektora [0.3058026339655925, 0.4253864432140819, -0.8517811473549874]
Wartość własna: 0.02557437263431785
Dla wektora [-0.8612123089479676, 0.5050427946808966, -0.056966081640040724]
```

W zależności od wywołania oraz metody wyniki mogą lekko się różnić natomiast są one równoznaczne ze sobą oraz mają postać:

$$v_1 = \begin{bmatrix} 0.40595333226396413 \\ 0.7509847864441626 \\ 0.5207914578343267 \end{bmatrix} \lambda_1 = 8.54851285322279$$

$$v_2 = \begin{bmatrix} 0.3058026339655925 \\ 0.4253864432140819 \\ -0.8517811473549874 \end{bmatrix} \lambda_2 = -4.574087225857102$$

$$v_3 = \begin{bmatrix} -0.8612123089479676 \\ 0.5050427946808966 \\ -0.056966081640040724] \end{bmatrix} \lambda_3 = 0.02557437263431785$$