Cassini Ovals and Their Applications

0. Motivation

Cassini's ovals are an area that contains eigenvalues. Cassini ovals are an important tool to estimate the spectrum of the matrix. With the help of spectrum estimation it is possible to accelerate methods such as the shifted inverse power method and the QR algorithm.

I. Introduction

Consider the matrix:

$$\begin{bmatrix} 8 & -1+i & 4 & -2 \\ -3+i & 2i & 0 & 3 \\ 0 & 1 & -2 & 6i \\ -1 & -2 & 1 & 5 \end{bmatrix}$$

How can we localize it's eigenvalues?

1. Gershgorin Circles

$$A = (a_{ij})_{i,j=1}^{n}$$

$$R_i'(A) = \sum_{j=1, j\neq i}^{n} |a_{ii}|$$

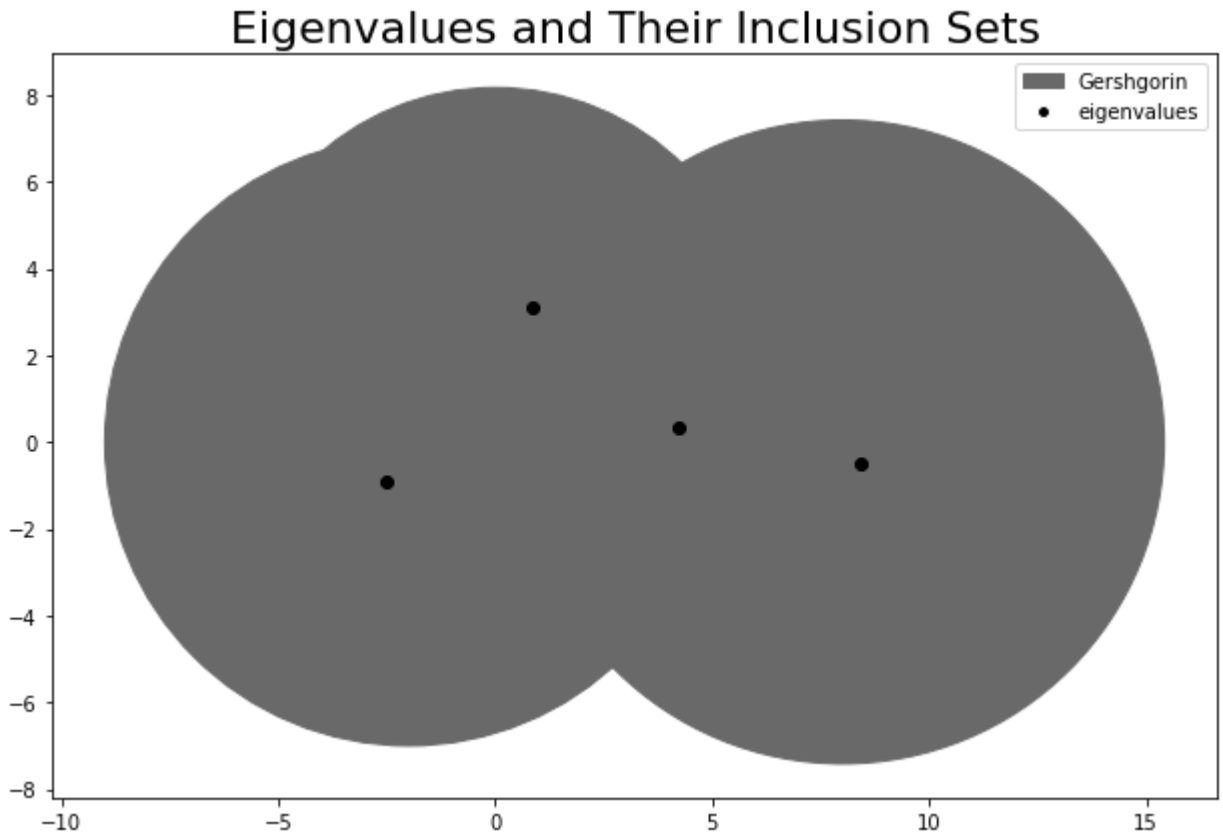
$$\Gamma^R(A) = \bigcup_{i=1}^{n} \left\{ z \in \mathbb{C} : |z - a_{ii}| \leq R_i'(A) \right\}$$

In [5]:

```
fig, ax = plt.subplots(1, 1, figsize=(12, 6), tight_layout=True)
plt.title('Eigenvalues and Their Inclusion Sets', fontsize=22)


A = np.array([[      8, -1 + 1j,  4, -2],
              [-3 + 1j,       2j,  0,  3],
              [      0,        1, -2, 6j],
              [     -1,       -2,  1,  5]])

plot_sets(A, ax, ['gamma'])
```



2. Cassini Ovals (Brauer Set)

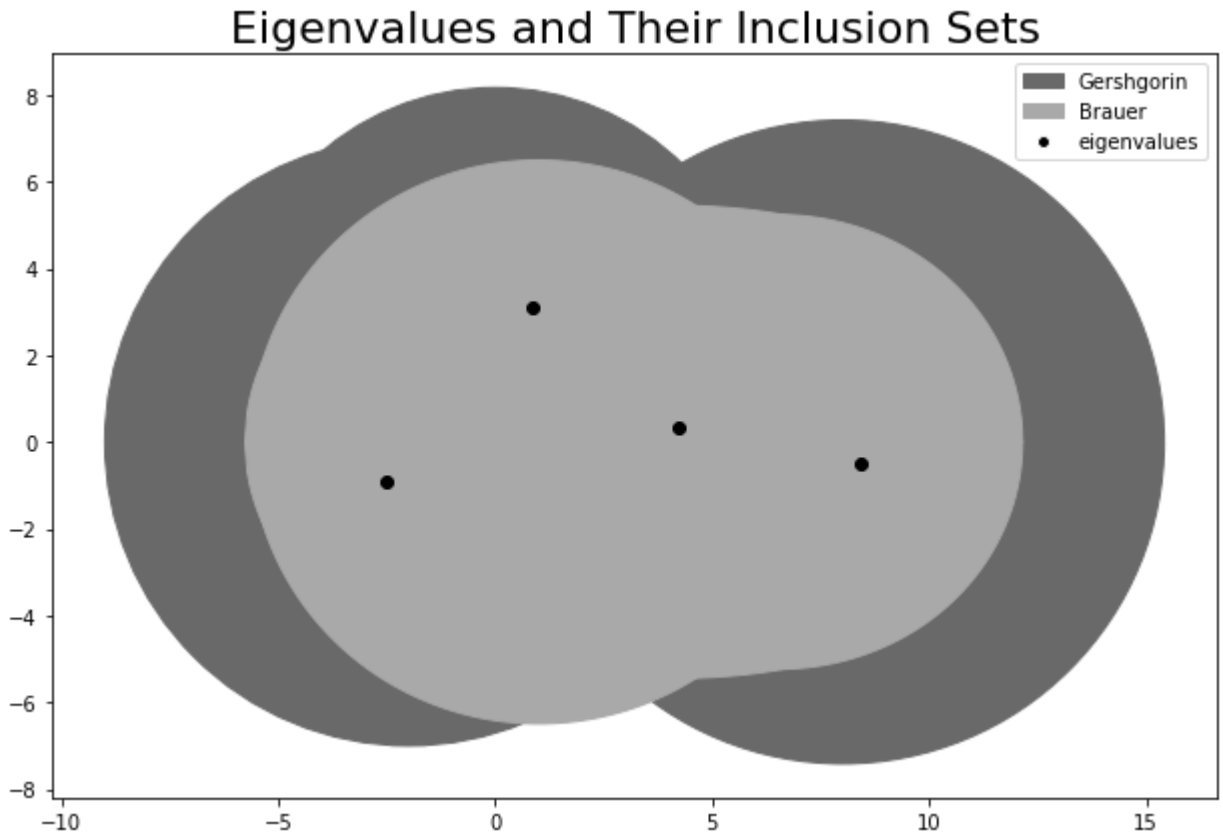
$$\Delta^R(A) = \bigcup_{i \neq j}^{n} \Big\{ z \in \mathbb{C} : |z - a_{ii}||z - a_{jj}| \leq R'_i(A)R'_j(A) \Big\}$$

In [6]:

```
fig, ax = plt.subplots(1, 1, figsize=(12, 6), tight_layout=True)
plt.title('Eigenvalues and Their Inclusion Sets', fontsize=22)


A = np.array([[      8, -1 + 1j,  4, -2],
              [-3 + 1j,       2j,  0,  3],
              [      0,        1, -2, 6j],
              [     -1,       -2,  1,  5]])

plot_sets(A, ax, ['gamma', 'delta'])
```



Ok, but about the following matrix?

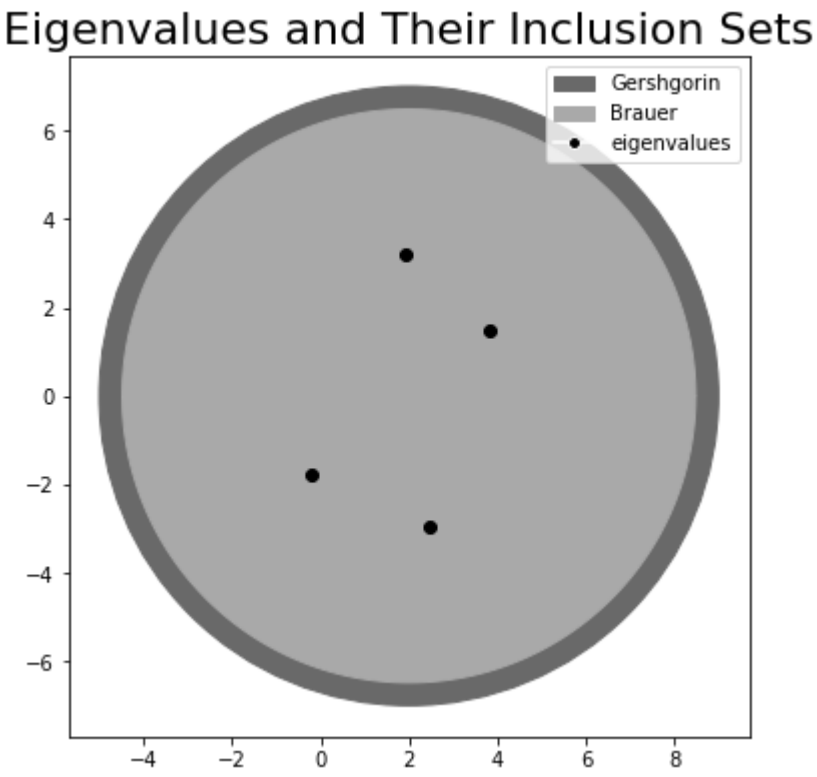
$$\begin{bmatrix} 2 & i & -3 & -i \\ 0 & 2 & 1 & -5 \\ 4 & 1 & 2 & 2 \\ i & -1 & 1 & 2 \end{bmatrix}$$

In [7]:

```
fig, ax = plt.subplots(1, 1, figsize=(6, 6), tight_layout=True)
plt.title('Eigenvalues and Their Inclusion Sets', fontsize=22)

A = np.array([[ 2, 1j, -3, -1j],
              [ 0,  2,  1, -5j],
              [ 4,  1,  2,   2],
              [1j, -1,  1,   2]])

plot_sets(A, ax, ['gamma', 'delta'])
```



3*. Cassini Ovals (Melman Set)

Cassini Ovals for matrices with constant main diagonal.

Main idea: form A_0^2 and utilize Gershgorin Theorem:

$$\Omega^R(A) = \bigcup_{i=1}^{n} \left\{ z \in \mathbb{C} : |z - \alpha - \sqrt{(A_0^2)_{ii}}||z - \alpha + \sqrt{(A_0^2)_{ii}}| \leq R_i'(A_0^2) \right\},$$

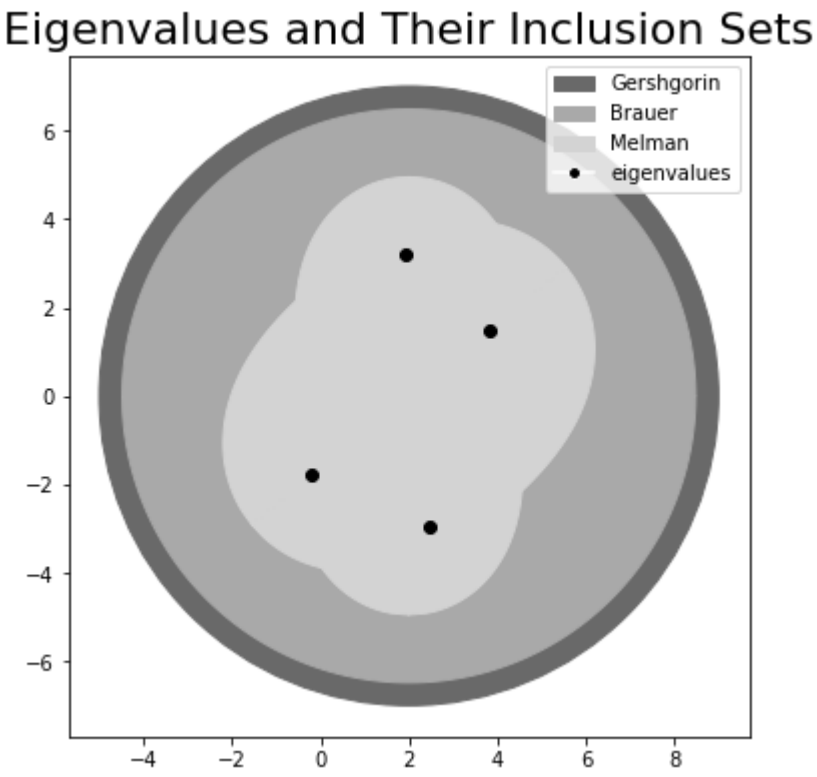
where α is the diagonal element and $A_0 = A - \alpha I$

In [8]:

```
fig, ax = plt.subplots(1, 1, figsize=(6, 6), tight_layout=True)
plt.title('Eigenvalues and Their Inclusion Sets', fontsize=22)

A = np.array([[ 2, 1j, -3, -1j],
              [ 0,  2,  1, -5j],
              [ 4,  1,  2,   2],
              [1j, -1,  1,   2]])

plot_sets(A, ax)
```



Important class of matrices with constant main diagonal are Toeplitz matrices!

$$T = \begin{bmatrix} t_0 & t_1 & \dots & t_{n-1} \\ t_{-1} & t_0 & \dots & t_{n-2} \\ \vdots & \vdots & \ddots & \vdots \\ t_{1-n} & t_{2-n} & \dots & t_0 \end{bmatrix}$$

II. Melman Set Properties

1. Inclusion Sets for Random Band Matrices with a Constant Main Diagonal

Let us denote the radius of the disc containing $\Omega^R(A)$ and centred at α by ρ_Ω and the radius of the discs defining $\Gamma^R(A)$ and $\Delta^R(A)$ by ρ_Γ and ρ_Δ, respectively.

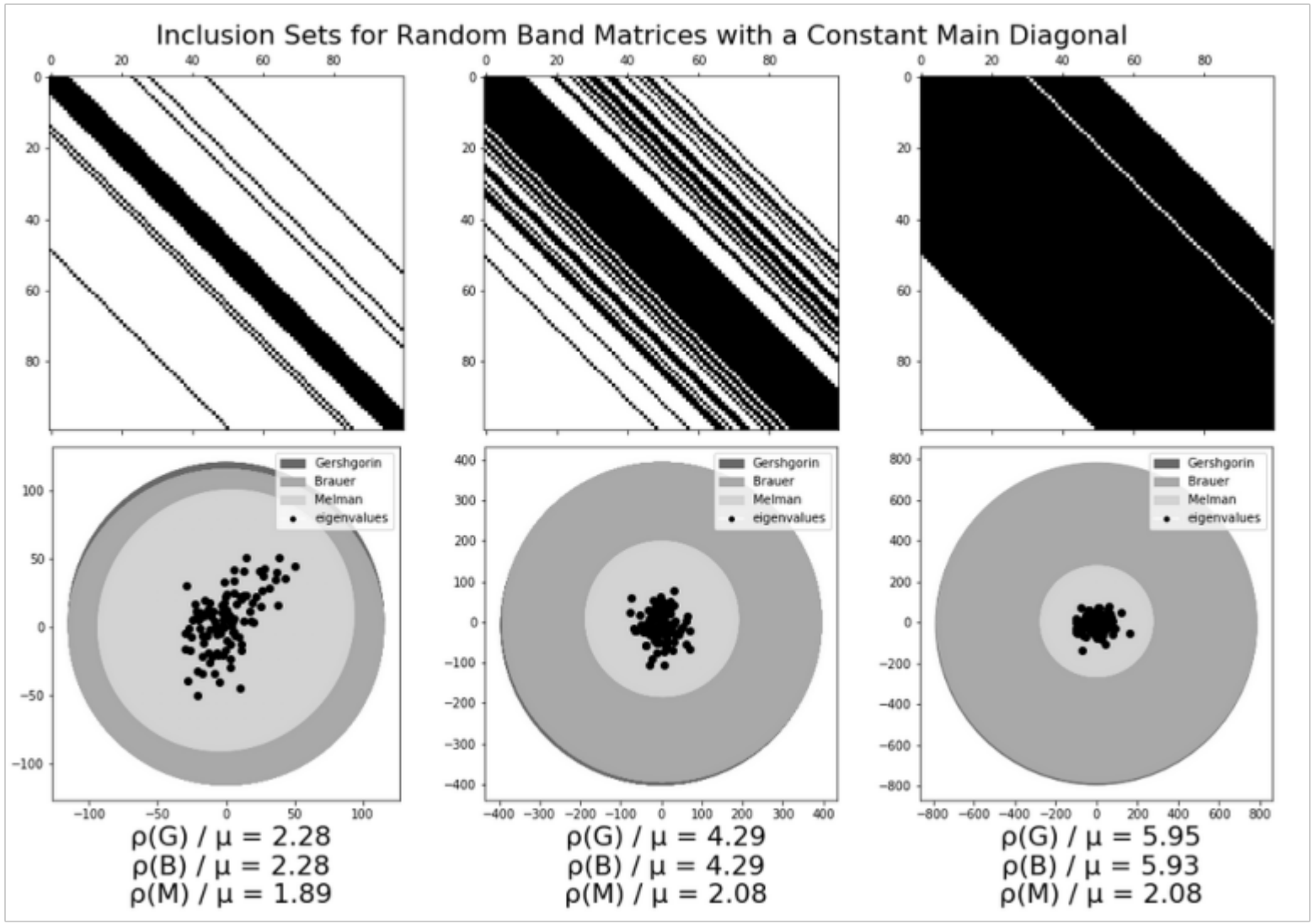
We also denote $\mu = \max_{1 \leq i \leq n} |\lambda_i - \alpha|$, where the λ_i's are the eigenvalues of A and α isthe constant diagonal element of A.

Then

$$\rho_\Omega \leq \rho_\Delta \leq \rho_\Gamma$$

In [19]:

```
plt.figure(figsize=(20, 10))
plt.imshow(mpimg.imread(f'img/band.png'))
plt.axis('off')
plt.show()
```



2. Toeplitz Matrices Dominated by Two Side Diagonals

When the absolute values of the elements in upper diagonal, symmetrically placed lower diagonal, dominate the absolute values of the other elements of the matrix.

The square of such a matrix has a dominant main diagonal. Gershgorin's discs give the best results for such matrices and since the sets are based on them, we can expect them to work particularly well in these cases.

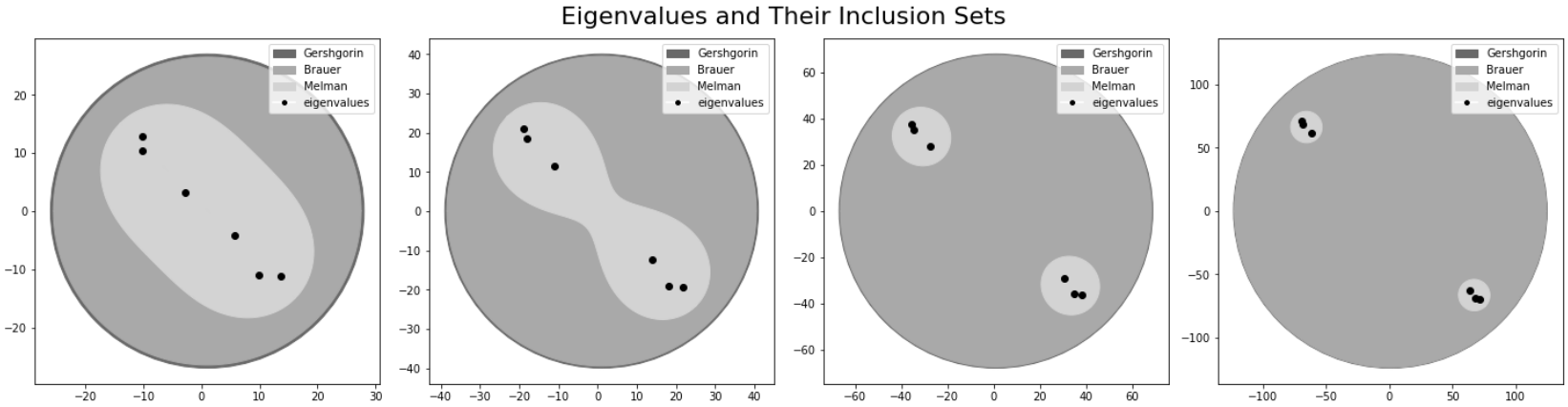
In [12]:

```
fig, ax = plt.subplots(2, 4, figsize=(20, 10), tight_layout=True)
fig.suptitle('Eigenvalues and Their Inclusion Sets', fontsize=22, y=.55)

for i in range(4):
    r = [1, 2j, -1, -10j * (2 ** i), -6j, 8j]
    c = [1, -1, 1j,   14 * (2 ** i),   8, 1j]

    A = toeplitz(c, r)
    ax[0][i].imshow(mpimg.imread(f'img/m{i+1}.png'))
    ax[0][i].axis('off')
    plot_sets(A, ax[1][i])
```

$$\begin{bmatrix} 1 & 2i & -1 & -10i & -6i & 8i \\ -1 & 1 & 2i & -1 & -10i & -6i \\ i & -1 & 1 & 2i & -1 & -10i \\ 14 & i & -1 & 1 & 2i & -1 \\ 8 & 14 & i & -1 & 1 & 2i \\ 1 & 8 & 14 & i & -1 & 1 \end{bmatrix} \quad \begin{bmatrix} 1 & 2i & -1 & -20i & -6i & 8i \\ -1 & 1 & 2i & -1 & -20i & -6i \\ i & -1 & 1 & 2i & -1 & -20i \\ 28 & i & -1 & 1 & 2i & -1 \\ 8 & 28 & i & -1 & 1 & 2i \\ 1 & 8 & 28 & i & -1 & 1 \end{bmatrix} \quad \begin{bmatrix} 1 & 2i & -1 & -40i & -6i & 8i \\ -1 & 1 & 2i & -1 & -40i & -6i \\ i & -1 & 1 & 2i & -1 & -40i \\ 56 & i & -1 & 1 & 2i & -1 \\ 8 & 56 & i & -1 & 1 & 2i \\ 1 & 8 & 56 & i & -1 & 1 \end{bmatrix} \quad \begin{bmatrix} 1 & 2i & -1 & -80i & -6i & 8i \\ -1 & 1 & 2i & -1 & -80i & -6i \\ i & -1 & 1 & 2i & -1 & -80i \\ 112 & i & -1 & 1 & 2i & -1 \\ 8 & 112 & i & -1 & 1 & 2i \\ 1 & 8 & 112 & i & -1 & 1 \end{bmatrix}$$



III. Applications

0. Eigenvalue Spectrum Bounds Estimate

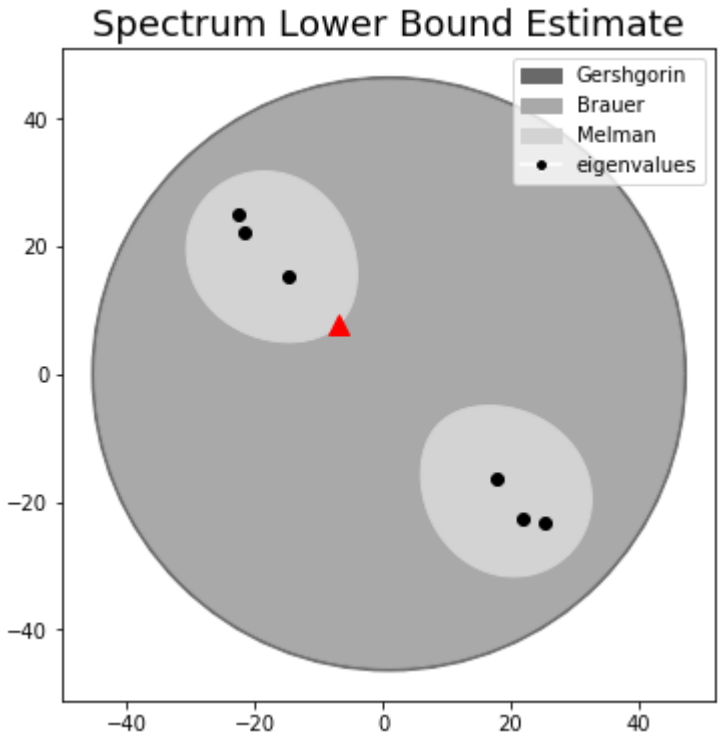
In [14]:

```
r, c = [1, 2j, -1, -10j*(2**1.3), -6j, 8j], [1, -1, 1j, 14*(2**1.3), 8, 1j]
A = toeplitz(c, r)

ovals = get_melman(A)
min_point, _ = get_nearest_melman(ovals)

fig, ax = plt.subplots(1, 1, figsize=(6, 6))

plot_sets(A, ax)
plt.plot(*min_point, '^', markersize=10, color='red')
plt.title('Spectrum Lower Bound Estimate', fontsize=18)
plt.show()
```



1. Inverse Power Iteration

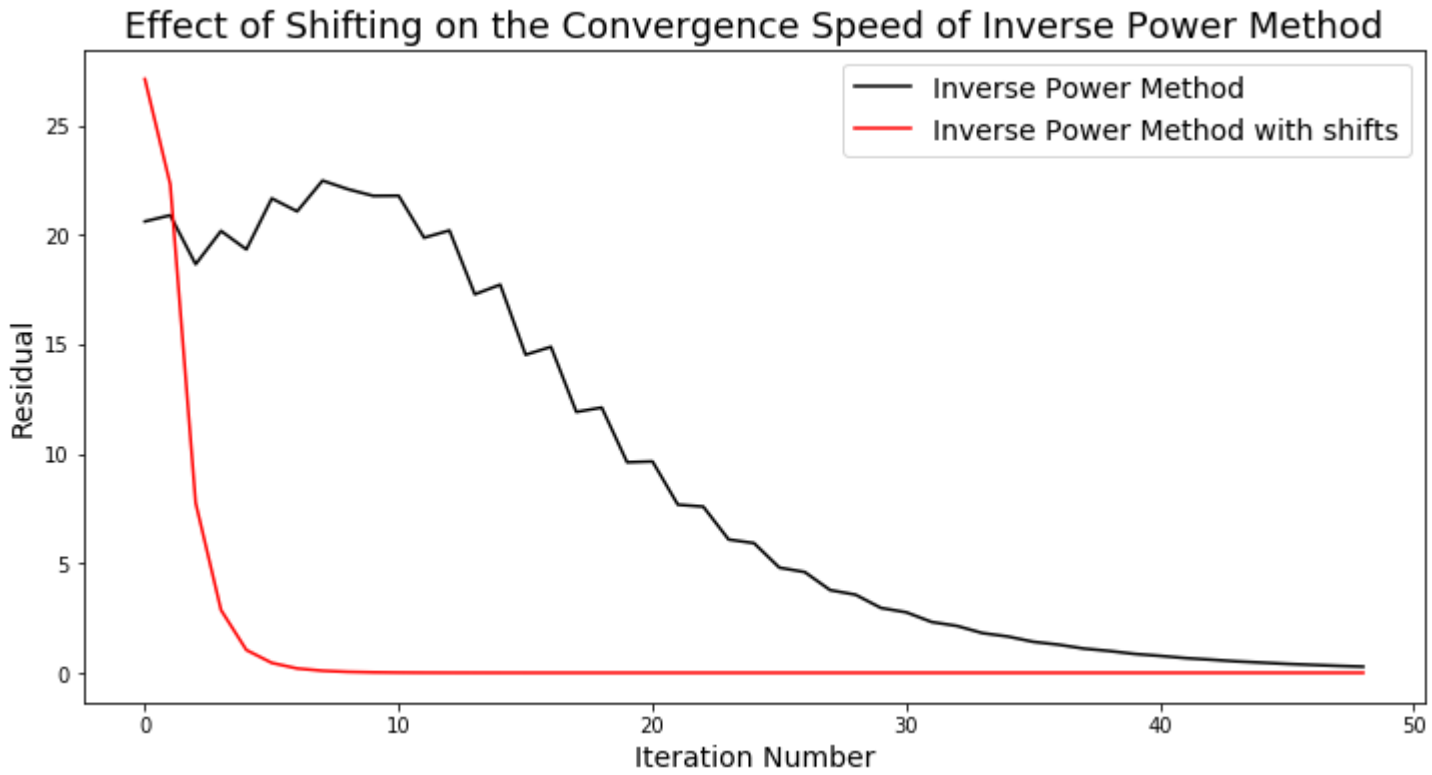
In [17]:

```
plt.figure(figsize=(12, 6))

plt.plot(res_, color='k', label='Inverse Power Method')
plt.plot(res, color='r', label='Inverse Power Method with shifts')

plt.title('Effect of Shifting on the Convergence Speed of Inverse Power Method', fontsize=18
plt.ylabel('Residual', fontsize=14)
plt.xlabel('Iteration Number', fontsize=14)

plt.legend(fontsize=14)
plt.show()
```



2. QR Algorithm

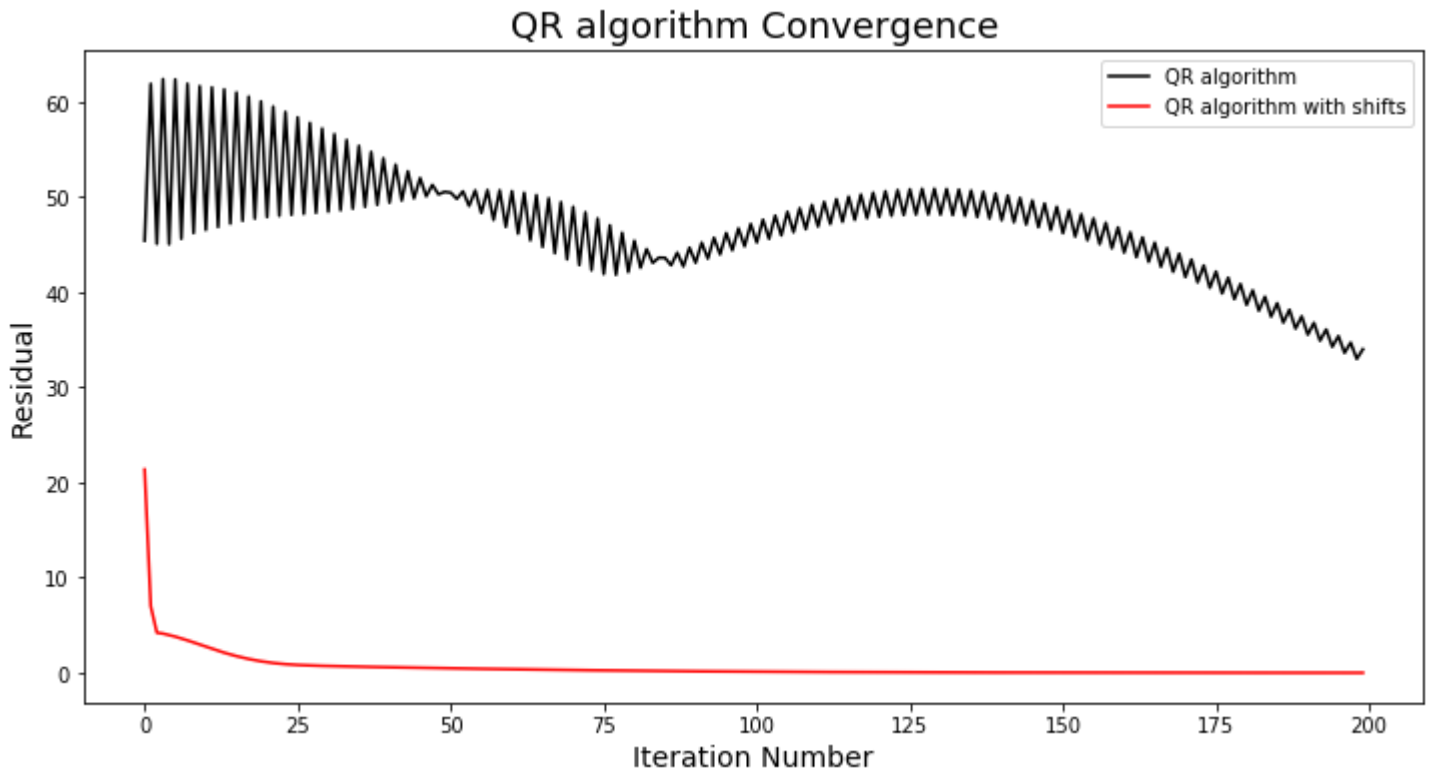
In [21]:

```
plt.figure(figsize=(12, 6))

plt.plot(deltas_0,  color='k', label='QR algorithm')
plt.plot(deltas,  color='r', label='QR algorithm with shifts')

plt.title('QR algorithm Convergence', fontsize=18)
plt.ylabel('Residual', fontsize=14)
plt.xlabel('Iteration Number', fontsize=14)

plt.legend()
plt.show()
```



Conclusion

- Matrices with constant main diagonal are quite common, e.g. Toeplitz
- Gershgorin and Brauer sets does not show sufficient about the spectrum in this case
- Melman set as a solution
- Applications: Inverse Power Method and QR algorithm speed-up