# Fig.1: Symmetry-Selective Pairing Strength

```
import numpy as np

import matplotlib.pyplot as plt


# Simulated dominant eigenvalues for different pairing symmetry channels

symmetry_channels = ['s-wave', 'd-wave', 'p-wave', 'f-wave', 'extended s-wave']

eigenvalues = [0.85, 1.20, 1.35, 0.65, 1.10]  # Example values from your model context


# Create bar plot

plt.figure(figsize=(8, 5))

bars = plt.bar(symmetry_channels, eigenvalues, color='teal')


# Add labels and title

plt.xlabel('Symmetry Channel')

plt.ylabel('Dominant Eigenvalue (λ)')

plt.title('Symmetry-Selective Pairing Strength in Unconventional Superconductivity Model')


# Reference line showing λ = 1 (pairing threshold)

plt.axhline(y=1.0, color='gray', linestyle='--', linewidth=1, label='Critical λ = 1 (Pairing Threshold)')

plt.legend()


# Highlight the symmetry channel with the strongest pairing (max eigenvalue)

max_index = np.argmax(eigenvalues)

bars[max_index].set_color('darkred')


# Improve layout
```

```
plt.tight_layout()
plt.grid(axis='y', linestyle=':', alpha=0.6)

# Show the plot
plt.show()
```

## Fig.2: s-wave superconducting gap

```
import numpy as np
import matplotlib.pyplot as plt

def s_wave_gap(angle_rad, delta_0):
    """
    Calculates the magnitude of a s-wave superconducting gap as a function of angle.
    In the simplest BCS picture, the s-wave gap is isotropic (constant) in k-space.

    Args:
        angle_rad (float or np.ndarray): Angle in radians (not used for calculation,
                                        but kept for consistent function signature).
        delta_0 (float): The constant gap amplitude.

    Returns:
        float or np.ndarray: The s-wave gap, which is simply delta_0.
    """
    # For an s-wave gap, the magnitude is constant regardless of the angle.
    # We return delta_0 for each angle provided.
    if isinstance(angle_rad, np.ndarray):
```

```
        return np.full_like(angle_rad, delta_0)
    else:
        return delta_0

# --- Graph Parameters ---
delta_0_theoretical = 0.8  # Constant gap amplitude (arbitrary units)
angles_rad = np.linspace(0, 2 * np.pi, 200) # 200 points from 0 to 2*pi

# --- Theoretical Data ---
theoretical_gap_s_wave = s_wave_gap(angles_rad, delta_0_theoretical)

# --- Simulated Experimental Data ---
# Simulate 'measurements' at specific angles, adding some noise
experimental_angles_deg = np.array([0, 30, 60, 90, 120, 150, 180, 210, 240, 270, 300, 330,
360])
experimental_angles_rad = np.deg2rad(experimental_angles_deg)

# Base 'experimental' values from the theoretical model (should be constant)
base_experimental_gap_s_wave = s_wave_gap(experimental_angles_rad,
delta_0_theoretical)

# Add random noise to simulate experimental uncertainty
np.random.seed(44) # Changed seed for a different noise pattern
noise = np.random.normal(0, 0.05, len(experimental_angles_rad)) # Gaussian noise with
smaller std dev
simulated_experimental_gap_s_wave = base_experimental_gap_s_wave + noise

# Ensure simulated gap values are non-negative
simulated_experimental_gap_s_wave[simulated_experimental_gap_s_wave < 0] = 0
```

```
# --- Plotting ---
plt.figure(figsize=(10, 7))
plt.plot(angles_rad, theoretical_gap_s_wave, label=r'Theoretical s-wave gap: $|\Delta_0|$ (constant)', color='purple', linewidth=2)
plt.scatter(experimental_angles_rad, simulated_experimental_gap_s_wave,
            label='Simulated Experimental Data', color='cyan', marker='o', s=50, zorder=5)

plt.xlabel(r'Angle in k-space ($\phi$, radians)', fontsize=12)
plt.ylabel(r'Superconducting Gap Magnitude ($|\Delta|$)', fontsize=12)
plt.title('Theoretical s-wave Superconducting Gap vs. Simulated Experimental Data', fontsize=14)
plt.xticks([0, np.pi/4, np.pi/2, 3*np.pi/4, np.pi, 5*np.pi/4, 3*np.pi/2, 7*np.pi/4, 2*np.pi],
           [r'0', r'$\pi/4$', r'$\pi/2$', r'$3\pi/4$', r'$\pi$', r'$5\pi/4$', r'$3\pi/2$', r'$7\pi/4$', r'$2\pi$'])
plt.grid(True, linestyle='--', alpha=0.7)
plt.legend(fontsize=10)
plt.ylim(bottom=0) # Ensure y-axis starts at 0 for gap magnitude
plt.tight_layout()
plt.show()
```

## Fig.3: p-wave superconducting gap

```
import numpy as np
import matplotlib.pyplot as plt

def modulated_p_wave_gap(angle_rad, delta_0, anisotropy_factor):
    """
    Calculates the magnitude of a p-wave superconducting gap with an additional
```

```
    angular modulation, representing a more complex unconventional state.
    This form captures features beyond a simple p_x or p_y wave, potentially
    arising from interplay with crystal anisotropy or multi-band effects.

    The base p-wave dependence is sin(phi), and it's modulated by cos(2*phi).
    The absolute value is taken as experimental probes typically measure the magnitude.

    Args:
      angle_rad (float or np.ndarray): Angle in radians.
      delta_0 (float): The maximum base gap amplitude.
      anisotropy_factor (float): Factor controlling the strength of the modulation.
                                 Should be between 0 and 1 for typical behaviors.

    Returns:
      float or np.ndarray: The absolute value of the modulated p-wave gap at the given angle.
    """
    # Define the angular dependence for a p-wave with higher-order modulation
    # This form is still odd-parity in its base (before abs) but with added complexity
    gap_function = np.sin(angle_rad) * (1 + anisotropy_factor * np.cos(2 * angle_rad))
    return delta_0 * np.abs(gap_function)

# --- Graph Parameters ---
delta_0_theoretical = 1.0      # Maximum base gap amplitude (arbitrary units)
anisotropy_factor = 0.6        # Strength of the angular modulation
angles_rad = np.linspace(0, 2 * np.pi, 300) # More points for smooth curve with fine features

# --- Theoretical Data ---
```

```
theoretical_gap_p_wave_modulated = modulated_p_wave_gap(angles_rad,
delta_0_theoretical, anisotropy_factor)


# --- Simulated Experimental Data ---

# Simulate 'measurements' at specific angles, adding some noise

experimental_angles_deg = np.linspace(0, 360, 25, endpoint=False) # 25 points evenly
spaced

experimental_angles_rad = np.deg2rad(experimental_angles_deg)


# Base 'experimental' values from the theoretical model

base_experimental_gap_p_wave_modulated =
modulated_p_wave_gap(experimental_angles_rad, delta_0_theoretical, anisotropy_factor)


# Add random noise to simulate experimental uncertainty

np.random.seed(45) # Changed seed for a different noise pattern

noise = np.random.normal(0, 0.06, len(experimental_angles_rad)) # Gaussian noise

simulated_experimental_gap_p_wave_modulated =
base_experimental_gap_p_wave_modulated + noise


# Ensure simulated gap values are non-negative

simulated_experimental_gap_p_wave_modulated[simulated_experimental_gap_p_wave_m
odulated < 0] = 0



# --- Plotting ---

plt.figure(figsize=(10, 7))

plt.plot(angles_rad, theoretical_gap_p_wave_modulated,

        label=r'Theoretical modulated p-wave gap: $|\Delta_0 \sin(\phi) (1 + \alpha
\cos(2\phi))|$',

        color='darkblue', linewidth=2)
```

```
plt.scatter(experimental_angles_rad, simulated_experimental_gap_p_wave_modulated,
            label='Simulated Experimental Data', color='green', marker='o', s=50, zorder=5)

plt.xlabel(r'Angle in k-space ($\phi$, radians)', fontsize=12)
plt.ylabel(r'Superconducting Gap Magnitude ($|\Delta|$)', fontsize=12)
plt.title('Theoretical Modulated p-wave Superconducting Gap vs. Simulated Experimental Data', fontsize=14)
plt.xticks([0, np.pi/4, np.pi/2, 3*np.pi/4, np.pi, 5*np.pi/4, 3*np.pi/2, 7*np.pi/4, 2*np.pi],
           [r'0', r'$\pi/4$', r'$\pi/2$', r'$3\pi/4$', r'$\pi$', r'$5\pi/4$', r'$3\pi/2$', r'$7\pi/4$', r'$2\pi$'])
plt.grid(True, linestyle='--', alpha=0.7)
plt.legend(fontsize=10)
plt.ylim(bottom=0) # Ensure y-axis starts at 0 for gap magnitude
plt.tight_layout()
plt.show()
```

## Fig.4: d-wave superconducting gap

```
import numpy as np
import matplotlib.pyplot as plt

def d_wave_gap(angle_rad, delta_0):
    """
    Calculates the magnitude of a d-wave superconducting gap as a function of angle.

    Args:
        angle_rad (float or np.ndarray): Angle in radians.
        delta_0 (float): The maximum gap amplitude.
```

```
    Returns:
        float or np.ndarray: The absolute value of the d-wave gap at the given angle.
                             The absolute value is taken because experimental probes
                             typically measure the magnitude of the gap.
    """
    return delta_0 * np.abs(np.cos(2 * angle_rad))

# --- Graph Parameters ---
delta_0_theoretical = 1.0  # Maximum gap amplitude (arbitrary units)
angles_rad = np.linspace(0, 2 * np.pi, 200) # 200 points from 0 to 2*pi for smooth curve

# --- Theoretical Data ---
theoretical_gap = d_wave_gap(angles_rad, delta_0_theoretical)

# --- Simulated Experimental Data ---
# Simulate 'measurements' at specific angles, adding some noise
experimental_angles_deg = np.array([0, 22.5, 45, 67.5, 90, 112.5, 135, 157.5, 180,
                                    202.5, 225, 247.5, 270, 292.5, 315, 337.5, 360])
experimental_angles_rad = np.deg2rad(experimental_angles_deg)

# Base 'experimental' values from the theoretical model
base_experimental_gap = d_wave_gap(experimental_angles_rad, delta_0_theoretical)

# Add random noise to simulate experimental uncertainty
np.random.seed(42) # for reproducibility
noise = np.random.normal(0, 0.08, len(experimental_angles_rad)) # Gaussian noise with std dev 0.08
```

```
simulated_experimental_gap = base_experimental_gap + noise


# Ensure simulated gap values are non-negative
simulated_experimental_gap[simulated_experimental_gap < 0] = 0


# --- Plotting ---
plt.figure(figsize=(10, 7))
plt.plot(angles_rad, theoretical_gap, label=r'Theoretical d-wave gap: $|\Delta_0 \cos(2\phi)|$', color='blue', linewidth=2)
plt.scatter(experimental_angles_rad, simulated_experimental_gap,
        label='Simulated Experimental Data', color='red', marker='o', s=50, zorder=5)


plt.xlabel(r'Angle in k-space ($\phi$, radians)', fontsize=12)
plt.ylabel(r'Superconducting Gap Magnitude ($|\Delta|$)', fontsize=12)
plt.title('Theoretical d-wave Superconducting Gap vs. Simulated Experimental Data', fontsize=14)
plt.xticks([0, np.pi/4, np.pi/2, 3*np.pi/4, np.pi, 5*np.pi/4, 3*np.pi/2, 7*np.pi/4, 2*np.pi],
        [r'0', r'$\pi/4$', r'$\pi/2$', r'$3\pi/4$', r'$\pi$', r'$5\pi/4$', r'$3\pi/2$', r'$7\pi/4$', r'$2\pi$'])
plt.grid(True, linestyle='--', alpha=0.7)
plt.legend(fontsize=10)
plt.ylim(bottom=0) # Ensure y-axis starts at 0 for gap magnitude
plt.tight_layout()
plt.show()
```

## Fig.5: f-wave superconducting gap

```
import numpy as np
import matplotlib.pyplot as plt
```

```
def f_wave_gap(angle_rad, delta_0):
    """
    Calculates the magnitude of an f-wave superconducting gap as a function of angle.
    An f-wave gap typically exhibits six nodes in 2D k-space, often described by
    a cos(3*phi) or sin(3*phi) dependence. The absolute value is taken as
    experimental probes typically measure the magnitude of the gap.

    Args:
        angle_rad (float or np.ndarray): Angle in radians.
        delta_0 (float): The maximum gap amplitude.

    Returns:
        float or np.ndarray: The absolute value of the f-wave gap at the given angle.
    """
    # Define the angular dependence for a pure f-wave symmetry
    # This form is often used for a chiral f-wave or related symmetries.
    gap_function = np.cos(3 * angle_rad)
    return delta_0 * np.abs(gap_function)

# --- Graph Parameters ---
delta_0_theoretical = 1.0       # Maximum gap amplitude (arbitrary units)
angles_rad = np.linspace(0, 2 * np.pi, 400) # More points for smooth curve with more
oscillations

# --- Theoretical Data ---
theoretical_gap_f_wave = f_wave_gap(angles_rad, delta_0_theoretical)
```

```
# --- Simulated Experimental Data ---
# Simulate 'measurements' at specific angles, adding some noise
# More experimental points are used to resolve the intricate nodal structure of f-wave
experimental_angles_deg = np.linspace(0, 360, 40, endpoint=False) # 40 points evenly spaced
experimental_angles_rad = np.deg2rad(experimental_angles_deg)

# Base 'experimental' values from the theoretical model
base_experimental_gap_f_wave = f_wave_gap(experimental_angles_rad, delta_0_theoretical)

# Add random noise to simulate experimental uncertainty
np.random.seed(46) # Changed seed for a new noise pattern
noise = np.random.normal(0, 0.05, len(experimental_angles_rad)) # Gaussian noise
simulated_experimental_gap_f_wave = base_experimental_gap_f_wave + noise

# Ensure simulated gap values are non-negative
simulated_experimental_gap_f_wave[simulated_experimental_gap_f_wave < 0] = 0

# --- Plotting ---
plt.figure(figsize=(10, 7))
plt.plot(angles_rad, theoretical_gap_f_wave,
    label=r'Theoretical f-wave gap: $|\Delta_0 \cos(3\phi)|$',
    color='maroon', linewidth=2) # Distinct color
plt.scatter(experimental_angles_rad, simulated_experimental_gap_f_wave,
     label='Simulated Experimental Data', color='darkviolet', marker='o', s=50, zorder=5) # Distinct color

plt.xlabel(r'Angle in k-space ($\phi$, radians)', fontsize=12)
```

```
plt.ylabel(r'Superconducting Gap Magnitude ($|\Delta|$)', fontsize=12)
plt.title('Theoretical f-wave Superconducting Gap vs. Simulated Experimental Data', fontsize=14)
plt.xticks([0, np.pi/6, np.pi/3, np.pi/2, 2*np.pi/3, 5*np.pi/6, np.pi,
            7*np.pi/6, 4*np.pi/3, 3*np.pi/2, 5*np.pi/3, 11*np.pi/6, 2*np.pi],
           [r'0', r'$\pi/6$', r'$\pi/3$', r'$\pi/2$', r'$2\pi/3$', r'$5\pi/6$', r'$\pi$',
            r'$7\pi/6$', r'$4\pi/3$', r'$3\pi/2$', r'$5\pi/3$', r'$11\pi/6$', r'$2\pi$'],
           rotation=45) # Rotate for better readability due to more labels
plt.grid(True, linestyle='--', alpha=0.7)
plt.legend(fontsize=10)
plt.ylim(bottom=0) # Ensure y-axis starts at 0 for gap magnitude
plt.tight_layout()
plt.show()


# --- Experimental Relevance Paragraph ---
"""
### Experimental Relevance (for an F-wave Model)
```

## Fig.6: g-wave superconducting gap

```
import numpy as np
import matplotlib.pyplot as plt


def g_wave_gap(angle_rad, delta_0):
    """
    Calculates the magnitude of a g-wave superconducting gap as a function of angle.
    A simple form of g-wave symmetry often involves cos(4*phi) dependence.
```

```
    The absolute value is taken as experimental probes typically measure the magnitude.

    Args:
        angle_rad (float or np.ndarray): Angle in radians.
        delta_0 (float): The maximum gap amplitude.

    Returns:
        float or np.ndarray: The absolute value of the g-wave gap at the given angle.
    """
    return delta_0 * np.abs(np.cos(4 * angle_rad))

# --- Graph Parameters ---
delta_0_theoretical = 1.0  # Maximum gap amplitude (arbitrary units)
angles_rad = np.linspace(0, 2 * np.pi, 300) # More points for smooth curve with more
oscillations

# --- Theoretical Data ---
theoretical_gap_g_wave = g_wave_gap(angles_rad, delta_0_theoretical)

# --- Simulated Experimental Data ---
# Simulate 'measurements' at specific angles, adding some noise
# More experimental points to capture the finer features of g-wave
experimental_angles_deg = np.linspace(0, 360, 30, endpoint=False) # 30 points evenly
spaced
experimental_angles_rad = np.deg2rad(experimental_angles_deg)

# Base 'experimental' values from the theoretical model
base_experimental_gap_g_wave = g_wave_gap(experimental_angles_rad,
delta_0_theoretical)
```

```
# Add random noise to simulate experimental uncertainty
np.random.seed(43) # Changed seed for a different noise pattern
noise = np.random.normal(0, 0.07, len(experimental_angles_rad)) # Gaussian noise, slightly less spread
simulated_experimental_gap_g_wave = base_experimental_gap_g_wave + noise

# Ensure simulated gap values are non-negative
simulated_experimental_gap_g_wave[simulated_experimental_gap_g_wave < 0] = 0

# --- Plotting ---
plt.figure(figsize=(10, 7))
plt.plot(angles_rad, theoretical_gap_g_wave, label=r'Theoretical g-wave gap: $|\Delta_0 \cos(4\phi)|$', color='darkgreen', linewidth=2)
plt.scatter(experimental_angles_rad, simulated_experimental_gap_g_wave,
        label='Simulated Experimental Data', color='darkorange', marker='o', s=50, zorder=5)

plt.xlabel(r'Angle in k-space ($\phi$, radians)', fontsize=12)
plt.ylabel(r'Superconducting Gap Magnitude ($|\Delta|$)', fontsize=12)
plt.title('Theoretical g-wave Superconducting Gap vs. Simulated Experimental Data', fontsize=14)
plt.xticks([0, np.pi/4, np.pi/2, 3*np.pi/4, np.pi, 5*np.pi/4, 3*np.pi/2, 7*np.pi/4, 2*np.pi],
       [r'0', r'$\pi/4$', r'$\pi/2$', r'$3\pi/4$', r'$\pi$', r'$5\pi/4$', r'$3\pi/2$', r'$7\pi/4$', r'$2\pi$'])
plt.grid(True, linestyle='--', alpha=0.7)
plt.legend(fontsize=10)
plt.ylim(bottom=0) # Ensure y-axis starts at 0 for gap magnitude
plt.tight_layout()
plt.show()
```

# Fig.7: Python Code for gap_function.png

```
import numpy as np

import matplotlib.pyplot as plt


# Define the Brillouin zone

kx = np.linspace(-np.pi, np.pi, 300)

ky = np.linspace(-np.pi, np.pi, 300)

KX, KY = np.meshgrid(kx, ky)


# Define the chiral p-wave gap function amplitude

Delta0 = 1.0

Delta_abs = Delta0 * np.sqrt(np.sin(KX)**2 + np.sin(KY)**2)


# Plot

plt.figure(figsize=(7,6))

plt.contourf(KX, KY, Delta_abs, levels=100, cmap='inferno')

plt.colorbar(label=r'$|\Delta(\mathbf{k})|$')

plt.title('Chiral $p$-wave Gap Amplitude in Momentum Space')

plt.xlabel(r'$k_x$')

plt.ylabel(r'$k_y$')

plt.xticks([-np.pi, 0, np.pi], [r'$-\pi$', '0', r'$\pi$'])

plt.yticks([-np.pi, 0, np.pi], [r'$-\pi$', '0', r'$\pi$'])

plt.tight_layout()

plt.savefig("gap_function.png", dpi=300)

plt.show()
```