2025.10.22_MM_Hill_analysis

October 23, 2025

1 Basic Enzyme Kinetics

[9]:
```python
import numpy as np
import matplotlib.pyplot as plt
from mpl_toolkits.mplot3d import Axes3D
import plotly.graph_objects as go
```

1.1 Function definition

[12]:
```python
def menten_hill_kinetics(S, k_max, K_m, n):
    """
    Hill kinetics equation; reduces to Michaelis-Menten for n=1
    S: substrate concentration
    k_max: maximum reaction rate (includes enzyme concentration)
    K_m: Michaelis constant (fixed at 1)
    n: Hill coefficient (exponent of cooperativity, positive real number)
    """
    return (k_max * S**n) / (K_m**n + S**n)
```

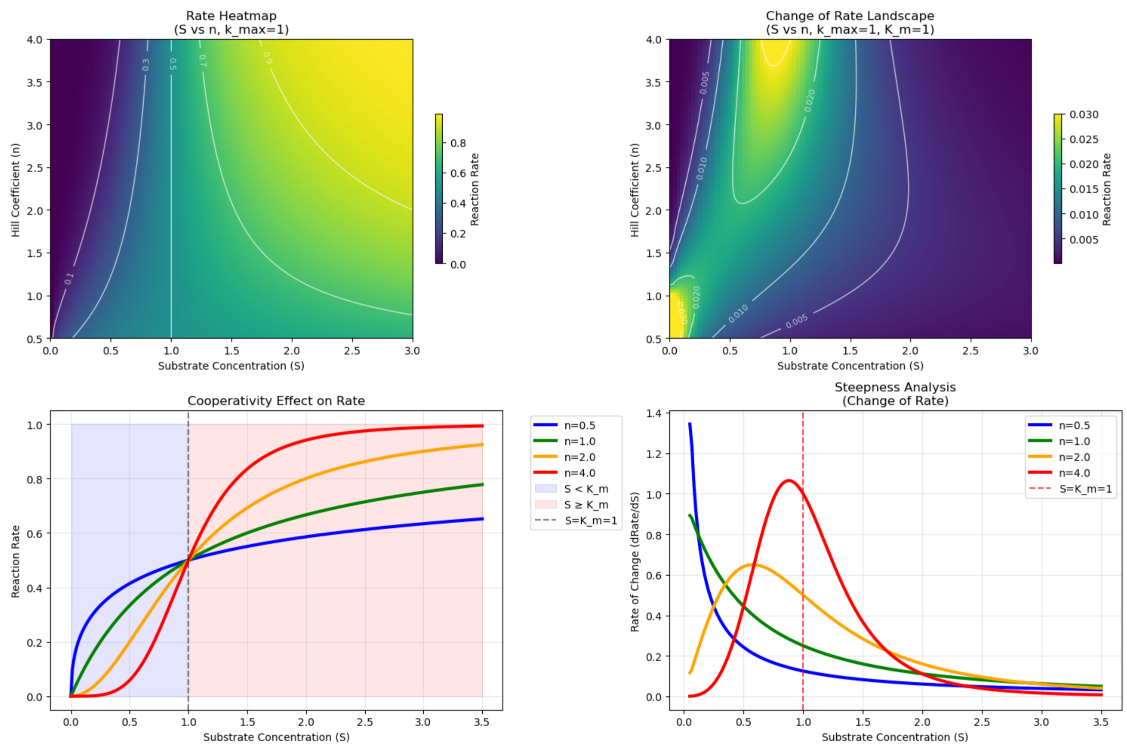
1.2 Rate and Change of Rate as a function of Substrate concentration

[15]:
```python
# Set fixed parameters
K_m   = 1
k_max = 1

# Ranges for substrate and Hill coefficient
S_values = np.linspace(0,   3, 100)
n_values = np.linspace(0.5, 4, 100)

# Meshgrid of S and n
S, N = np.meshgrid(S_values, n_values)

# Reaction rate and its gradient
rate_landscape = np.zeros_like(S)
for i in range(S.shape[0]):
    for j in range(S.shape[1]):
        rate_landscape[i,j] = menten_hill_kinetics(S[i,j], k_max, K_m, N[i,j])
```

```
rate_landscape_gradient = np.gradient(rate_landscape)


# Plot
fig = plt.figure(figsize=(15, 10))

# 1. Rate Landscape (S vs n)
ax1 = fig.add_subplot(2, 2, 1)

im1 = ax1.imshow(rate_landscape, extent=[S_values.min(), S_values.max(),
                                       n_values.min(), n_values.max()],
               origin='lower', aspect='auto', cmap='viridis')
ax1.set_xlabel('Substrate Concentration (S)')
ax1.set_ylabel('Hill Coefficient (n)')
ax1.set_title(f'Rate Heatmap\n(S vs n, k_max={k_max})')
plt.colorbar(im1, ax=ax1, label='Reaction Rate', shrink=0.5)

# Add contour lines
contour_levels = [0.1, 0.3, 0.5, 0.7, 0.9]
contours = ax1.contour(S, N, rate_landscape, levels=contour_levels,
                       colors='white', alpha=0.7, linewidths=1)
ax1.clabel(contours, inline=True, fontsize=8, fmt='%.1f')


# 2. Derivative of Rate (S vs n)
ax2 = fig.add_subplot(2, 2, 2)

im2 = ax2.imshow(rate_landscape_gradient[1], extent=[S_values.min(), S_values.
 ↪max(),
                                       n_values.min(), n_values.max()], vmax=0.
 ↪03,
               origin='lower', aspect='auto', cmap='viridis')
ax2.set_xlabel('Substrate Concentration (S)')
ax2.set_ylabel('Hill Coefficient (n)')
ax2.set_title(f'Change of Rate Landscape\n(S vs n, k_max={k_max}, K_m={K_m})')
plt.colorbar(im2, ax=ax2, label='Reaction Rate', shrink=0.5)

# Add contour lines
contour_levels_gradient = [0.005, 0.01, 0.02, 0.03]
contours_grad = ax2.contour(S, N, rate_landscape_gradient[1],␣
 ↪levels=contour_levels_gradient,
                       colors='white', alpha=0.7, linewidths=1)
ax2.clabel(contours_grad, inline=True, fontsize=8, fmt='%.3f')


# 3. Impact of cooperativity on reaction rate
```

```python
ax3 = fig.add_subplot(2, 2, 3)

S_fine = np.linspace(0, 3.5, 300)
n_transition = [0.5, 1.0, 2.0, 4.0]
colors = ['blue', 'green', 'orange', 'red']

for n, color in zip(n_transition, colors):
    rate = menten_hill_kinetics(S_fine, k_max, K_m, n)
    ax3.plot(S_fine, rate, color=color, linewidth=3, label=f'n={n}')

# Annotations
ax3.fill_between(S_fine, 0, 1, where=(S_fine<1), alpha=0.1, color='blue',␣
 ↪label='S < K_m')
ax3.fill_between(S_fine, 0, 1, where=(S_fine>=1), alpha=0.1, color='red',␣
 ↪label='S   K_m')
ax3.axvline(x=1, color='black', linestyle='--', alpha=0.5, label='S=K_m=1')

ax3.set_xlabel('Substrate Concentration (S)')
ax3.set_ylabel('Reaction Rate')
ax3.set_title('Cooperativity Effect on Rate')
ax3.legend(bbox_to_anchor=(1.05, 1), loc='upper left')
ax3.grid(True, alpha=0.3)

# 4. Impact of cooperativity on derivative of reaction rate
ax4 = fig.add_subplot(2, 2, 4)

S_analysis = np.linspace(0.05, 3.5, 200)

for ind, n in enumerate([0.5, 1.0, 2.0, 4.0]):
    rate = menten_hill_kinetics(S_analysis, k_max, K_m, n)
    derivative = np.gradient(rate, S_analysis)
    ax4.plot(S_analysis, derivative, label=f'n={n}', color=colors[ind],␣
 ↪linewidth=3)

ax4.axvline(x=1, color='red', linestyle='--', alpha=0.7, label='S=K_m=1')
ax4.set_xlabel('Substrate Concentration (S)')
ax4.set_ylabel('Rate of Change (dRate/dS)')
ax4.set_title('Steepness Analysis\n(Change of Rate)')
ax4.legend()
ax4.grid(True, alpha=0.3)


plt.tight_layout()
plt.show()


# Print analysis
```

```python
print("Enzyme Kinetics without and with cooperativity")
print("=" * 50)
print(f"Parameters: k_max = {k_max}, K_m = {K_m}")
print("\nKey observations:")
print("1. n = 1: Classic hyperbolic Michaelis-Menten kinetics")
print("2. n < 1: Negative cooperativity - shallow curves")
print("3. n > 1: Positive cooperativity - sigmoidal curves")
print("4. n >> 1: Switch-like behavior - very steep transition")
print(f"5. Half-maximal rate always occurs at S = K_m = {K_m}")
print("\nThe 3D landscape clearly shows how increasing n transforms")
print("the response from gradual to switch-like behavior!")
```



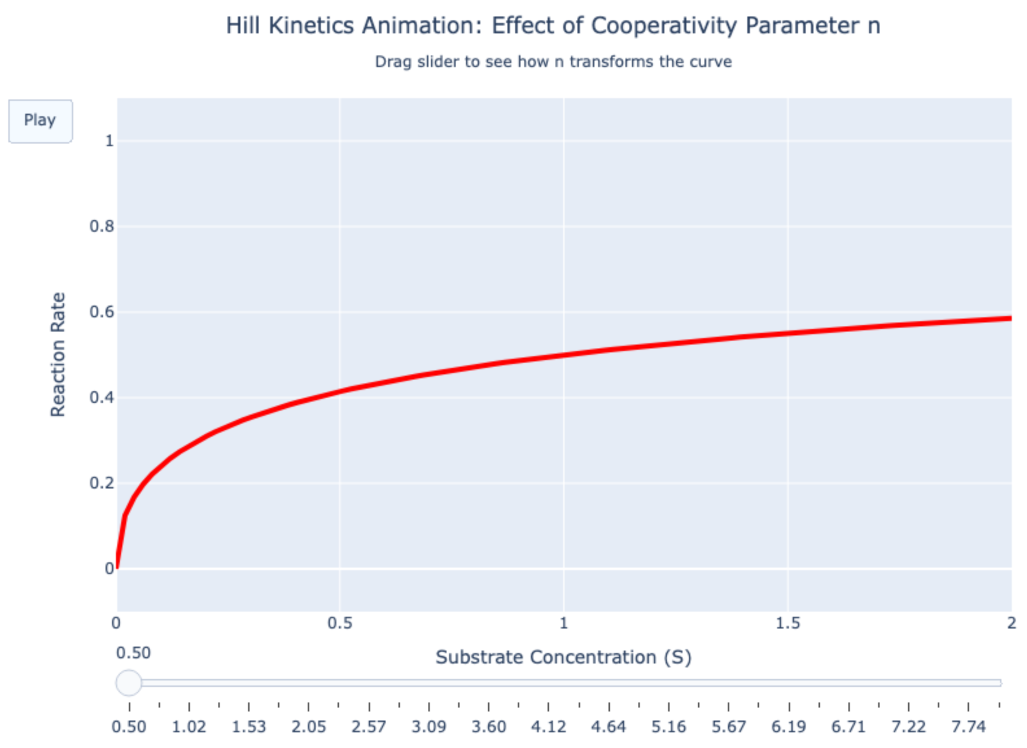
```
Enzyme Kinetics without and with cooperativity
==================================================
Parameters: k_max = 1, K_m = 1

Key observations:
1. n = 1: Classic hyperbolic Michaelis-Menten kinetics
2. n < 1: Negative cooperativity - shallow curves
3. n > 1: Positive cooperativity - sigmoidal curves
4. n >> 1: Switch-like behavior - very steep transition
5. Half-maximal rate always occurs at S = K_m = 1

The 3D landscape clearly shows how increasing n transforms
```

the response from gradual to switch-like behavior!

```
[ ]:
```

1.3 Interactive Graph: slider for Hill coefficient

```
[29]: S_values = np.linspace(0, 2, 100)  # substrate range

frames = []

n_animation = np.linspace(0.5, 8, 30)

for n_val in n_animation:
    rate_curve = menten_hill_kinetics(S_values, k_max, K_m, n_val)

    frames.append(go.Frame(
        data=[go.Scatter(
            x=S_values,
            y=rate_curve,
            mode='lines',
            line=dict(color='red', width=4),
            name=f'n = {n_val:.2f}'
        )],
        name=f"n={n_val:.2f}"
    ))


# Create the animation figure
fig_animation = go.Figure(
    data=[go.Scatter(
        x=S_values,
        y=menten_hill_kinetics(S_values, k_max, K_m, n_animation[0]),
        mode='lines',
        line=dict(color='red', width=4),
        name='Rate'
    )],
    layout=go.Layout(
        title=dict(
            text="Hill Kinetics Animation: Effect of Cooperativity Parameter␣
 ↪n<br><sub>Drag slider to see how n transforms the curve</sub>",
            x=0.5
        ),
        xaxis=dict(title="Substrate Concentration (S)", range=[0,␣
 ↪S_values[-1]]),
        yaxis=dict(title="Reaction Rate", range=[-0.1, 1.1]),
        width=600,    # Figure width in pixels
        height=600,
        updatemenus=[dict(
            type="buttons",
            buttons=[dict(label="Play",
                        method="animate",
                        args=[None, {"frame": {"duration": 100, "redraw":␣
↪True},
                                    "fromcurrent": True}])]
        )],
        sliders=[dict(
            steps=[dict(method="animate",
                        args=[[f"n={n_val:.2f}"],
                              {"mode": "immediate",
                               "frame": {"duration": 100, "redraw": True},
                               "transition": {"duration": 0}}],
                        label=f"{n_val:.2f}") for n_val in n_animation]
        )]

    ),
    frames=frames
)

fig_animation.show()
```



[ ]:

```

```

[ ]:

```

```

[ ]:

```

```

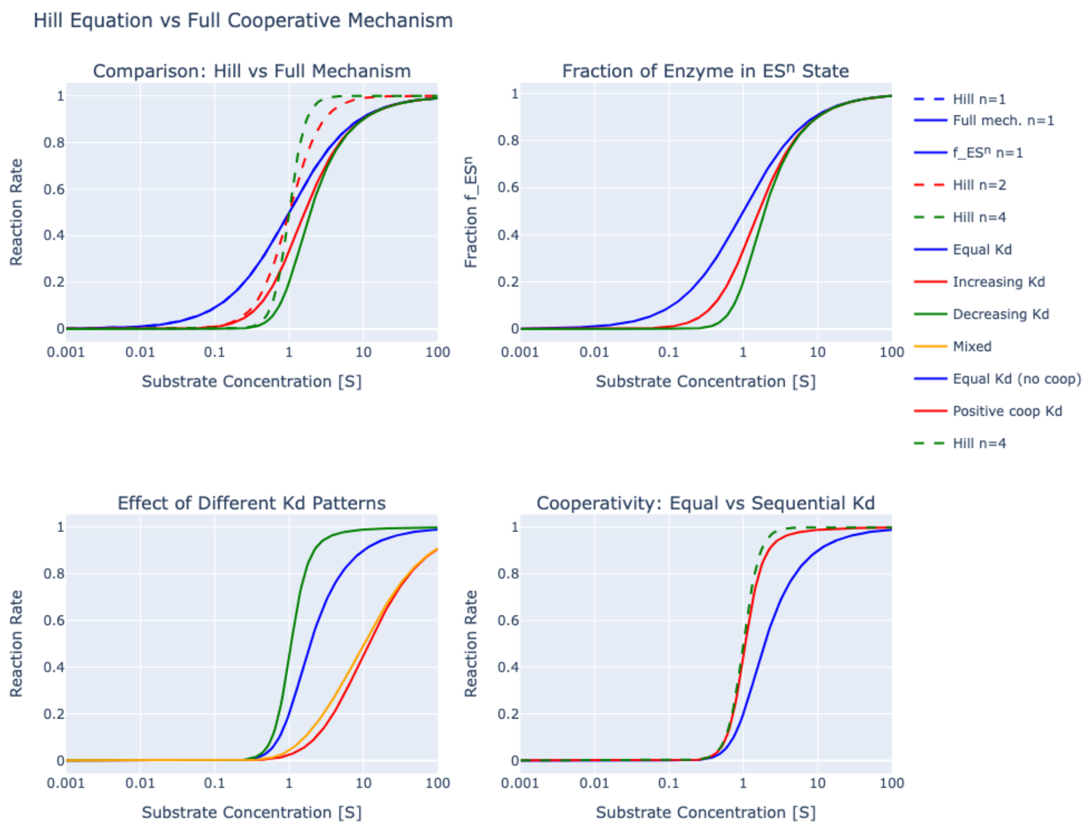
1.4 Optional: Comparison of Hill approximation with full mechanism

[9]:

```python
import numpy as np
import plotly.graph_objects as go
from plotly.subplots import make_subplots

def hill_kinetics(S, k_max, K_m, n):
    """Classic Hill equation (approximate)"""
    return (k_max * S**n) / (K_m**n + S**n)

def full_cooperative_kinetics(S, k_cat, E_total, Kd_values):
    """
    Full cooperative mechanism with explicit binding steps
    S: substrate concentration
    k_cat: catalytic rate constant
    E_total: total enzyme concentration
    Kd_values: dissociation constants for each binding step [Kd , Kd , ..., Kd ]
    """
    n = len(Kd_values)

    # Calculate concentrations of all enzyme species using partition function
    Z_terms = [1]  # Free enzyme
    for i in range(1, n + 1):
        # Product of binding constants up to step i
        binding_term = 1
        for j in range(i):
            binding_term *= (S / Kd_values[j])
        Z_terms.append(binding_term)

    Z = sum(Z_terms)

    # Fraction of enzyme in fully bound state ES
    f_ESn = Z_terms[-1] / Z

    # Reaction rate = k_cat * [ES ]
    rate = k_cat * E_total * f_ESn

    return rate  # Return just the rate, not a tuple

def full_cooperative_kinetics_equal_Kd(S, k_cat, E_total, Kd, n):
    """Simplified case where all Kd values are equal"""
    Kd_values = [Kd] * n
    return full_cooperative_kinetics(S, k_cat, E_total, Kd_values)

def get_f_ESn(S, k_cat, E_total, Kd_values):
    """Helper function to get just the fraction of ES"""
    n = len(Kd_values)
    Z_terms = [1]
    for i in range(1, n + 1):
        binding_term = 1
        for j in range(i):
            binding_term *= (S / Kd_values[j])
        Z_terms.append(binding_term)

    Z = sum(Z_terms)
    f_ESn = Z_terms[-1] / Z
    return f_ESn

# Parameters for comparison
k_cat = 1.0      # catalytic rate constant
E_total = 1.0    # total enzyme concentration
n_values = [1, 2, 4]  # Different cooperativities

# Create substrate range
S_range = np.logspace(-3, 2, 200)  # 0.001 to 100

# Create comparison plots
fig = make_subplots(
    rows=2, cols=2,
    subplot_titles=(
        'Comparison: Hill vs Full Mechanism',
        'Fraction of Enzyme in ES  State',
        'Effect of Different Kd Patterns',
        'Cooperativity: Equal vs Sequential Kd'
    ),
    specs=[[{"secondary_y": False}, {"secondary_y": False}],
           [{"secondary_y": False}, {"secondary_y": False}]]
)

# Colors for different n values
colors = ['blue', 'red', 'green']

for i, n in enumerate(n_values):
    # Hill equation parameters
    K_m_hill = 1.0
    k_max_hill = k_cat * E_total

    # Full mechanism with equal Kd (simplest cooperative case)
    Kd_equal = 1.0  # All binding sites have same affinity

    # Calculate rates
    hill_rates = hill_kinetics(S_range, k_max_hill, K_m_hill, n)
    full_rates_equal = full_cooperative_kinetics_equal_Kd(S_range, k_cat,␣
 ↪E_total, Kd_equal, n)
    f_ESn_equal = get_f_ESn(S_range, k_cat, E_total, [Kd_equal] * n)

    # Plot comparison
    fig.add_trace(
        go.Scatter(x=S_range, y=hill_rates,
                   name=f'Hill n={n}', line=dict(color=colors[i], dash='dash'),
                   legendgroup=f'group{n}'),
        row=1, col=1
    )

    fig.add_trace(
        go.Scatter(x=S_range, y=full_rates_equal,
                   name=f'Full mech. n={n}', line=dict(color=colors[i]),
                   legendgroup=f'group{n}', showlegend=(i==0)),
        row=1, col=1
    )

    # Plot fraction of ES
    fig.add_trace(
        go.Scatter(x=S_range, y=f_ESn_equal,
                   name=f'f_ES  n={n}', line=dict(color=colors[i]),
                   legendgroup=f'frac{n}', showlegend=(i==0)),
        row=1, col=2
    )

# Show effect of different Kd patterns for n=4
n_complex = 4
Kd_patterns = {
    'Equal Kd': [1.0, 1.0, 1.0, 1.0],
    'Increasing Kd': [0.1, 0.5, 2.0, 10.0],  # Binding gets harder
    'Decreasing Kd': [10.0, 2.0, 0.5, 0.1],  # Binding gets easier (positive␣
 ↪cooperativity)
    'Mixed': [0.1, 10.0, 0.1, 10.0]
}

pattern_colors = ['blue', 'red', 'green', 'orange']

for j, (pattern_name, Kd_list) in enumerate(Kd_patterns.items()):
    full_rates_pattern = full_cooperative_kinetics(S_range, k_cat, E_total,␣
 ↪Kd_list)
```

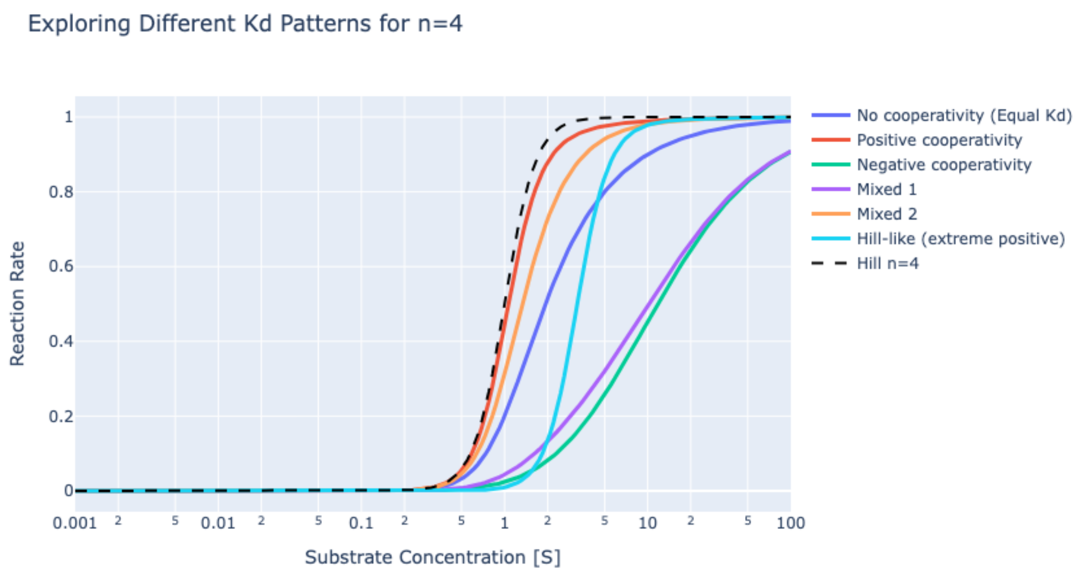
```
    fig.add_trace(
        go.Scatter(x=S_range, y=full_rates_pattern,
                   name=pattern_name, line=dict(color=pattern_colors[j]),
                   legendgroup=f'pattern{j}'),
        row=2, col=1
    )

# Compare equal Kd vs true positive cooperativity for n=4
Kd_equal = [1.0, 1.0, 1.0, 1.0]
Kd_positive_coop = [10.0, 2.0, 0.5, 0.1]  # True positive cooperativity

full_rates_equal = full_cooperative_kinetics(S_range, k_cat, E_total, Kd_equal)
full_rates_coop = full_cooperative_kinetics(S_range, k_cat, E_total,␣
 ↪Kd_positive_coop)
hill_rates = hill_kinetics(S_range, k_max_hill, K_m_hill, 4)

fig.add_trace(
    go.Scatter(x=S_range, y=full_rates_equal,
               name='Equal Kd (no coop)', line=dict(color='blue')),
    row=2, col=2
)

fig.add_trace(
    go.Scatter(x=S_range, y=full_rates_coop,
               name='Positive coop Kd', line=dict(color='red')),
    row=2, col=2
)

fig.add_trace(
    go.Scatter(x=S_range, y=hill_rates,
               name='Hill n=4', line=dict(color='green', dash='dash')),
    row=2, col=2
)

# Update layout
fig.update_layout(
    title_text="Hill Equation vs Full Cooperative Mechanism",
    height=800,
    width=1000,
    showlegend=True
)

# Update axes
fig.update_xaxes(title_text="Substrate Concentration [S]", type="log", row=1,␣
 ↪col=1)
fig.update_yaxes(title_text="Reaction Rate", row=1, col=1)
fig.update_xaxes(title_text="Substrate Concentration [S]", type="log", row=1,␣
 ↪col=2)
fig.update_yaxes(title_text="Fraction f_ES ", row=1, col=2)
fig.update_xaxes(title_text="Substrate Concentration [S]", type="log", row=2,␣
 ↪col=1)
fig.update_yaxes(title_text="Reaction Rate", row=2, col=1)
fig.update_xaxes(title_text="Substrate Concentration [S]", type="log", row=2,␣
 ↪col=2)
fig.update_yaxes(title_text="Reaction Rate", row=2, col=2)

fig.show()

# Let's also create a focused comparison for n=2 to see the math clearly
print("\n" + "="*60)
print("MATHEMATICAL COMPARISON FOR n=2")
print("="*60)

# For n=2 case
S_sample = np.array([0.1, 0.5, 1.0, 2.0, 5.0, 10.0])
Kd_equal_2 = [1.0, 1.0]
Kd_positive_2 = [10.0, 0.1]  # Strong positive cooperativity

print(f"{'S':>6} {'Hill':>8} {'Equal Kd':>8} {'Pos Coop':>8} {'f_ES2_eq':>8}␣
 ↪{'f_ES2_coop':>8}")
print("-" * 60)

for s in S_sample:
    hill_rate = hill_kinetics(s, k_cat * E_total, 1.0, 2)
    full_equal = full_cooperative_kinetics(s, k_cat, E_total, Kd_equal_2)
    full_coop = full_cooperative_kinetics(s, k_cat, E_total, Kd_positive_2)
    f_ES2_eq = get_f_ESn(s, k_cat, E_total, Kd_equal_2)
    f_ES2_coop = get_f_ESn(s, k_cat, E_total, Kd_positive_2)

    print(f"{s:6.1f} {hill_rate:8.4f} {full_equal:8.4f} {full_coop:8.4f}␣
 ↪{f_ES2_eq:8.4f} {f_ES2_coop:8.4f}")

print("\nKEY OBSERVATIONS:")
print("1. Hill equation overestimates cooperativity for equal Kd case")
print("2. True positive cooperativity (decreasing Kd) gives sigmoidal shape")
print("3. Equal Kd case is less cooperative - intermediates are populated")
print("4. Hill coefficient n is a measure of apparent cooperativity, not␣
 ↪necessarily number of sites")

# Create a simple interactive plot to explore different Kd patterns
print("\nCreating interactive Kd explorer...")
```

```python
def explore_Kd_patterns():
    """Interactive exploration of different Kd patterns"""

    # Test cases for n=4
    test_cases = {
        'No cooperativity (Equal Kd)': [1.0, 1.0, 1.0, 1.0],
        'Positive cooperativity': [10.0, 2.0, 0.5, 0.1],
        'Negative cooperativity': [0.1, 0.5, 2.0, 10.0],
        'Mixed 1': [0.1, 10.0, 0.1, 10.0],
        'Mixed 2': [10.0, 0.1, 10.0, 0.1],
        'Hill-like (extreme positive)': [100.0, 10.0, 1.0, 0.1]
    }

    fig_explore = go.Figure()

    for case_name, Kd_list in test_cases.items():
        rates = full_cooperative_kinetics(S_range, k_cat, E_total, Kd_list)
        fig_explore.add_trace(
            go.Scatter(x=S_range, y=rates, name=case_name, line=dict(width=3))
        )

    # Add Hill equation for comparison
    hill_rates = hill_kinetics(S_range, k_cat * E_total, 1.0, 4)
    fig_explore.add_trace(
        go.Scatter(x=S_range, y=hill_rates, name='Hill n=4',
                   line=dict(dash='dash', color='black', width=2))
    )

    fig_explore.update_layout(
        title="Exploring Different Kd Patterns for n=4",
        xaxis_title="Substrate Concentration [S]",
        yaxis_title="Reaction Rate",
        xaxis_type="log",
        width=800,
        height=500
    )

    return fig_explore

fig_explore = explore_Kd_patterns()
fig_explore.show()
```



```
============================================================
MATHEMATICAL COMPARISON FOR n=2
============================================================
     S     Hill Equal Kd Pos Coop f_ES2_eq f_ES2_coop
------------------------------------------------------------
   0.1   0.0099   0.0090   0.0098   0.0090   0.0098
   0.5   0.2000   0.1429   0.1923   0.1429   0.1923
   1.0   0.5000   0.3333   0.4762   0.3333   0.4762
   2.0   0.8000   0.5714   0.7692   0.5714   0.7692
   5.0   0.9615   0.8065   0.9434   0.8065   0.9434
  10.0   0.9901   0.9009   0.9804   0.9009   0.9804

KEY OBSERVATIONS:
1. Hill equation overestimates cooperativity for equal Kd case
2. True positive cooperativity (decreasing Kd) gives sigmoidal shape
3. Equal Kd case is less cooperative - intermediates are populated
4. Hill coefficient n is a measure of apparent cooperativity, not necessarily
number of sites
```

```
Creating interactive Kd explorer…
```



```
[ ]:
```