In [30]:

```
import numpy as np
import matplotlib.pyplot as plt
import pandas as pd
from matplotlib.patches import Rectangle

# ============================================================================
# EXPERIMENTAL DATA
# ============================================================================

# Reactant masses (g)
mass_5_iodovanillin = 0.252  # g
mass_K2CO3 = 0.403  # g
mass_boronic_acid = 0.170  # g (4-Methylphenylboronic acid)
mass_Pd_C_actual = 0.05  # g (10% Pd/C) - ACTUAL USED
mass_Pd_C_theoretical = 0.005  # g (10% Pd/C) - SHOULD HAVE USED
volume_water = 15  # mL

# Molecular weights (g/mol)
MW_5_iodovanillin = 278.04  # C8H7IO3
MW_K2CO3 = 138.21
MW_boronic_acid = 135.96  # C7H9BO2 (4-methylphenylboronic acid)
MW_product = 228.24  # C15H14O3 (coupled product: 5-(4-methylphenyl)vanillin)
MW_Pd = 106.42  # Palladium atomic weight

# Pd/C catalyst specifications
Pd_loading = 0.10  # 10% Pd by weight

# ============================================================================
# STOICHIOMETRIC CALCULATIONS
# ============================================================================

print("="*80)
print("SUZUKI-MIYAURA COUPLING REACTION ANALYSIS")
print("5-Iodovanillin + 4-Methylphenylboronic Acid → 5-(4-Methylphenyl)vanillin")
print("="*80)
print()

# Calculate moles of each reactant
moles_5_iodovanillin = mass_5_iodovanillin / MW_5_iodovanillin
moles_K2CO3 = mass_K2CO3 / MW_K2CO3
moles_boronic_acid = mass_boronic_acid / MW_boronic_acid

print("REACTANT QUANTITIES:")
print(f"5-Iodovanillin: {mass_5_iodovanillin:.3f} g = {moles_5_iodovanillin*1000:.3
print(f"4-Methylphenylboronic acid: {mass_boronic_acid:.3f} g = {moles_boronic_acid
print(f"K₂CO₃ (base): {mass_K2CO3:.3f} g = {moles_K2CO3*1000:.3f} mmol")
print()

# Determine limiting reagent (1:1 stoichiometry)
limiting_reagent = "5-Iodovanillin" if moles_5_iodovanillin < moles_boronic_acid el
limiting_moles = min(moles_5_iodovanillin, moles_boronic_acid)

print(f"LIMITING REAGENT: {limiting_reagent}")
print(f"Theoretical moles of product: {limiting_moles*1000:.3f} mmol")
```

```
print(f"Theoretical yield: {limiting_moles * MW_product:.3f} g")
print()

# ============================================================================
# CATALYST LOADING ANALYSIS
# ============================================================================

print("="*80)
print("PALLADIUM CATALYST ANALYSIS")
print("="*80)
print()

# Calculate Pd content
mass_Pd_actual = mass_Pd_C_actual * Pd_loading
mass_Pd_theoretical = mass_Pd_C_theoretical * Pd_loading
moles_Pd_actual = mass_Pd_actual / MW_Pd
moles_Pd_theoretical = mass_Pd_C_theoretical / MW_Pd

# Calculate mol% catalyst loading
mol_percent_actual = (moles_Pd_actual / limiting_moles) * 100
mol_percent_theoretical = (moles_Pd_theoretical / limiting_moles) * 100

print(f"ACTUAL CATALYST USED: {mass_Pd_C_actual:.3f} g of 10% Pd/C")
print(f"  - Pd content: {mass_Pd_actual:.4f} g = {moles_Pd_actual*1000:.4f} mmol")
print(f"  - Catalyst loading: {mol_percent_actual:.2f} mol% Pd")
print()

print(f"THEORETICAL CATALYST (should have used): {mass_Pd_C_theoretical:.3f} g of 1
print(f"  - Pd content: {mass_Pd_theoretical:.4f} g = {moles_Pd_theoretical*1000:.4
print(f"  - Catalyst loading: {mol_percent_theoretical:.2f} mol% Pd")
print()

excess_factor = mass_Pd_C_actual / mass_Pd_C_theoretical
print(f"EXCESS CATALYST FACTOR: {excess_factor:.1f}x")
print(f"You used {excess_factor:.1f} times more catalyst than intended!")
print()

# ============================================================================
# EFFECTS OF EXCESS CATALYST
# ============================================================================

print("="*80)
print("EFFECTS OF EXCESS Pd/C CATALYST (10x excess)")
print("="*80)
print()

effects = {
    "Economic Impact": [
        f"- Wasted ${(mass_Pd_C_actual - mass_Pd_C_theoretical) * 50:.2f} in cataly
        "- Palladium is expensive; excess significantly increases cost"
    ],
    "Reaction Kinetics": [
        "- Faster initial rate due to more active sites",
        "- May reach completion sooner",
        "- Diminishing returns: rate plateaus beyond optimal loading"
    ],
```

```
    "Side Reactions & Selectivity": [
        "- Increased risk of homocoupling (Ar-Ar from boronic acid)",
        "- Potential for over-reduction of substrates",
        "- May promote dehalogenation without coupling",
        "- Pd nanoparticle aggregation at high loading"
    ],
    "Product Purification": [
        "- More Pd leaching into product",
        "- Difficult to remove Pd contamination (ICP-MS detection)",
        "- Activated carbon filtration less effective",
        "- Potential product discoloration (Pd black formation)"
    ],
    "Mechanistic Considerations": [
        "- Transmetalation not rate-limiting at high [Pd]",
        "- Reductive elimination may be affected",
        "- Catalyst aggregation reduces active Pd(0) concentration",
        "- Pd(II)/Pd(0) equilibrium shifted"
    ],
    "Green Chemistry Violations": [
        "- Violates atom economy principles",
        "- Excessive use of precious metal catalyst",
        "- Increased environmental impact",
        "- Not aligned with sustainable synthesis practices"
    ]
}

for category, points in effects.items():
    print(f"{category}:")
    for point in points:
        print(f"  {point}")
    print()

# ============================================================================
# VISUALIZATION: CATALYST LOADING EFFECTS
# ============================================================================

fig, axes = plt.subplots(2, 2, figsize=(14, 10))
fig.suptitle('Effect of Excess Pd/C Catalyst in Suzuki-Miyaura Coupling',
             fontsize=14, fontweight='bold')

# 1. Catalyst Loading Comparison
ax1 = axes[0, 0]
categories = ['Theoretical\n(0.005 g)', 'Actual\n(0.05 g)']
Pd_amounts = [mass_Pd_theoretical*1000, mass_Pd_actual*1000]  # in mg
colors = ['#2ecc71', '#e74c3c']

bars = ax1.bar(categories, Pd_amounts, color=colors, alpha=0.7, edgecolor='black',
ax1.axhline(y=mass_Pd_theoretical*1000, color='green', linestyle='--',
            label='Optimal loading', linewidth=2)
ax1.set_ylabel('Pd Content (mg)', fontsize=11, fontweight='bold')
ax1.set_title('Catalyst Loading Comparison', fontsize=12, fontweight='bold')
ax1.legend()
ax1.grid(axis='y', alpha=0.3)

# Add value labels on bars
for bar, val in zip(bars, Pd_amounts):
```

```
    height = bar.get_height()
    ax1.text(bar.get_x() + bar.get_width()/2., height,
            f'{val:.3f} mg\n({val/mass_Pd_theoretical/1000:.1f}x)',
            ha='center', va='bottom', fontsize=10, fontweight='bold')

# 2. mol% Pd Loading
ax2 = axes[0, 1]
mol_percents = [mol_percent_theoretical, mol_percent_actual]
bars2 = ax2.bar(categories, mol_percents, color=colors, alpha=0.7,
                edgecolor='black', linewidth=1.5)
ax2.set_ylabel('Catalyst Loading (mol% Pd)', fontsize=11, fontweight='bold')
ax2.set_title('Catalyst Loading vs Limiting Reagent', fontsize=12, fontweight='bold
ax2.axhspan(0.5, 3, alpha=0.2, color='green', label='Typical range (0.5-3 mol%)')
ax2.legend()
ax2.grid(axis='y', alpha=0.3)

for bar, val in zip(bars2, mol_percents):
    height = bar.get_height()
    ax2.text(bar.get_x() + bar.get_width()/2., height,
            f'{val:.2f} mol%',
            ha='center', va='bottom', fontsize=10, fontweight='bold')

# 3. Theoretical Yield vs Catalyst Loading Curve
ax3 = axes[1, 0]
catalyst_loadings = np.linspace(0.1, 20, 100)  # mol% range
# Hypothetical yield curve (diminishing returns model)
yields = 100 * (1 - np.exp(-0.8 * catalyst_loadings))  # Asymptotic approach to 100

ax3.plot(catalyst_loadings, yields, 'b-', linewidth=2.5, label='Expected Yield')
ax3.axvline(x=mol_percent_theoretical, color='green', linestyle='--',
            linewidth=2, label=f'Theoretical ({mol_percent_theoretical:.2f} mol%)')
ax3.axvline(x=mol_percent_actual, color='red', linestyle='--',
            linewidth=2, label=f'Actual ({mol_percent_actual:.2f} mol%)')
ax3.fill_between([0.5, 3], 0, 100, alpha=0.2, color='green', label='Optimal range')
ax3.set_xlabel('Catalyst Loading (mol% Pd)', fontsize=11, fontweight='bold')
ax3.set_ylabel('Theoretical Yield (%)', fontsize=11, fontweight='bold')
ax3.set_title('Yield vs Catalyst Loading (Theoretical Model)', fontsize=12, fontwei
ax3.set_xlim(0, 20)
ax3.set_ylim(0, 105)
ax3.legend(loc='lower right')
ax3.grid(alpha=0.3)

# 4. Cost-Benefit Analysis
ax4 = axes[1, 1]
loading_range = np.array([0.5, 1, 2, 3, 5, 10, 20])  # mol%
relative_cost = loading_range / mol_percent_theoretical  # Relative to theoretical
benefit = 100 * (1 - np.exp(-0.8 * loading_range))  # Same yield model
efficiency = benefit / relative_cost  # Benefit per unit cost

ax4_twin = ax4.twinx()
line1 = ax4.plot(loading_range, relative_cost, 'r-o', linewidth=2,
                 markersize=8, label='Relative Cost')
line2 = ax4_twin.plot(loading_range, efficiency, 'g-s', linewidth=2,
                      markersize=8, label='Cost Efficiency')
ax4.axvline(x=mol_percent_theoretical, color='green', linestyle='--',
            linewidth=2, alpha=0.5)
```

```
ax4.axvline(x=mol_percent_actual, color='red', linestyle='--',
            linewidth=2, alpha=0.5)

ax4.set_xlabel('Catalyst Loading (mol% Pd)', fontsize=11, fontweight='bold')
ax4.set_ylabel('Relative Cost (fold)', fontsize=11, fontweight='bold', color='r')
ax4_twin.set_ylabel('Cost Efficiency (yield/cost)', fontsize=11, fontweight='bold',
ax4.set_title('Economic Analysis: Cost vs Efficiency', fontsize=12, fontweight='bol
ax4.tick_params(axis='y', labelcolor='r')
ax4_twin.tick_params(axis='y', labelcolor='g')
ax4.grid(alpha=0.3)

lines = line1 + line2
labels = [l.get_label() for l in lines]
ax4.legend(lines, labels, loc='upper right')

plt.tight_layout()
plt.show()

# ============================================================================
# RECOMMENDATIONS TABLE
# ============================================================================

print("="*80)
print("RECOMMENDATIONS FOR FUTURE EXPERIMENTS")
print("="*80)
print()

recommendations_data = {
    'Aspect': [
        'Catalyst Amount',
        'Catalyst Loading',
        'Reaction Time',
        'Workup Procedure',
        'Quality Control',
        'Cost Optimization'
    ],
    'Recommendation': [
        f'Use {mass_Pd_C_theoretical:.3f} g (not {mass_Pd_C_actual:.3f} g)',
        '0.5-3 mol% Pd is optimal for most Suzuki couplings',
        'Monitor by TLC; excess catalyst may speed up but wastes material',
        'Add activated carbon treatment to remove Pd contamination',
        'Check Pd content in product by ICP-MS or similar',
        f'Save ${(mass_Pd_C_actual - mass_Pd_C_theoretical) * 50:.2f} per reaction'
    ]
}

df_recommendations = pd.DataFrame(recommendations_data)
print(df_recommendations.to_string(index=False))
print()

# ============================================================================
# MECHANISTIC INSIGHTS
# ============================================================================

print("="*80)
print("MECHANISTIC INSIGHTS (Based on Literature)")
```

```
print("="*80)
print()

print("SUZUKI-MIYAURA CATALYTIC CYCLE:")
print("1. Oxidative Addition: Ar-I + Pd(0) → Ar-Pd(II)-I")
print("2. Transmetalation: Ar-Pd(II)-I + Ar'-B(OH)₂ + Base → Ar-Pd(II)-Ar' + Base•I
print("3. Reductive Elimination: Ar-Pd(II)-Ar' → Ar-Ar' + Pd(0)")
print()

print("EFFECT OF 10x EXCESS CATALYST:")
print()
print("Rate-Determining Step Considerations (Mondal et al., 2018):")
print("  - At low [Pd]: Oxidative addition often rate-limiting")
print("  - At high [Pd]: Transmetalation or reductive elimination becomes limiting"
print("  - Excess catalyst doesn't proportionally increase rate")
print()

print("Practical Implications (Miyaura & Suzuki, 1995; Hanley, 2014):")
print("  - Optimal loading: 0.5-5 mol% for most aryl halides")
print("  - Aryl iodides (like 5-iodovanillin) are highly reactive")
print("  - 10x excess shows diminishing returns in yield")
print("  - Increases purification challenges significantly")
print()

print("Green Chemistry Perspective (Palesch et al., 2019):")
print("  - Suzuki coupling already considered 'green' (water solvent, mild conditio
print("  - Excess precious metal catalyst undermines sustainability")
print("  - Proper catalyst optimization is key to green synthesis")
print()

# ============================================================================
# SUMMARY STATISTICS
# ============================================================================

print("="*80)
print("SUMMARY")
print("="*80)
print()

summary_stats = {
    'Parameter': [
        'Limiting Reagent',
        'Theoretical Yield (g)',
        'Theoretical Yield (mmol)',
        'Pd/C Used (g)',
        'Pd Content (mg)',
        'Catalyst Loading (mol%)',
        'Excess Factor',
        'Estimated Cost Increase ($)',
        'Typical Optimal Loading (mol%)'
    ],
    'Value': [
        limiting_reagent,
        f'{limiting_moles * MW_product:.3f}',
        f'{limiting_moles * 1000:.3f}',
        f'{mass_Pd_C_actual:.3f}',
```

```
        f'{mass_Pd_actual * 1000:.3f}',
        f'{mol_percent_actual:.2f}',
        f'{excess_factor:.1f}x',
        f'{(mass_Pd_C_actual - mass_Pd_C_theoretical) * 50:.2f}',
        '0.5-3.0'
    ]
}

df_summary = pd.DataFrame(summary_stats)
print(df_summary.to_string(index=False))
print()

print("="*80)
print("CONCLUSION:")
print("Using 10x excess Pd/C catalyst likely provided minimal benefit to yield")
print("while significantly increasing cost and purification difficulties.")
print("Future reactions should use the calculated 0.005 g amount.")
print("="*80)

import numpy as np
import matplotlib.pyplot as plt
from matplotlib import cm
from mpl_toolkits.mplot3d import Axes3D
from matplotlib.gridspec import GridSpec

# Set random seed for reproducibility
np.random.seed(42)

# Create a larger figure with a grid layout for side-by-side plots
fig = plt.figure(figsize=(20, 10))
gs = GridSpec(1, 2, width_ratios=[1, 1])

# Create two subplots with 3D projection
ax1 = fig.add_subplot(gs[0], projection='3d')
ax2 = fig.add_subplot(gs[1], projection='3d')

# Create a mesh grid for the reaction coordinate space
x = np.linspace(-4, 4, 80)
y = np.linspace(-4, 4, 80)
X, Y = np.meshgrid(x, y)

# --- Energy landscapes ---
def energy_landscape_normal(x, y):
    main_path = 3 * np.exp(-0.5 * x**2) * (1 - 0.5 * np.exp(-2 * y**2))
    barriers = 2 + 0.5 * (x**2 + 3 * y**2) * np.exp(-0.1 * (x**2 + y**2))
    return barriers - main_path

def energy_landscape_excess(x, y):
    main_path = 3.5 * np.exp(-0.5 * x**2) * (1 - 0.5 * np.exp(-2 * y**2))
    barriers = 2 + 0.5 * (x**2 + 3 * y**2) * np.exp(-0.1 * (x**2 + y**2))
    side_path1 = 1.5 * np.exp(-0.5 * (x-1)**2) * np.exp(-2 * (y-1.5)**2)
    side_path2 = 1.2 * np.exp(-0.5 * (x+1)**2) * np.exp(-2 * (y+1.5)**2)
    return barriers - main_path - side_path1 - side_path2

Z_normal = energy_landscape_normal(X, Y)
Z_excess = energy_landscape_excess(X, Y)
```

```python
# Find global min and max for consistent color scaling
z_min = min(Z_normal.min(), Z_excess.min())
z_max = max(Z_normal.max(), Z_excess.max())

# --- Improved 3D surfaces ---
surf1 = ax1.plot_surface(X, Y, Z_normal, cmap='coolwarm', alpha=0.9,
                         shade=True, linewidth=0, antialiased=False,
                         vmin=z_min, vmax=z_max)
surf2 = ax2.plot_surface(X, Y, Z_excess, cmap='coolwarm', alpha=0.9,
                         shade=True, linewidth=0, antialiased=False,
                         vmin=z_min, vmax=z_max)

# Add contour projections to the bottom
ax1.contour(X, Y, Z_normal, zdir='z', offset=z_min, cmap='coolwarm', alpha=0.6)
ax2.contour(X, Y, Z_excess, zdir='z', offset=z_min, cmap='coolwarm', alpha=0.6)

# Add shared color bar
cbar_ax = fig.add_axes([0.92, 0.15, 0.02, 0.7])
cbar = fig.colorbar(surf2, cax=cbar_ax)
cbar.set_label('Relative Energy (kcal/mol)', fontsize=14, labelpad=10)
cbar.ax.tick_params(labelsize=12)

# --- Reaction paths ---
path_x = np.linspace(-3, 3, 100)
path_y = np.zeros_like(path_x)
path_z_normal = energy_landscape_normal(path_x, path_y) + 0.05
path_z_excess = energy_landscape_excess(path_x, path_y) + 0.05

# Side paths for excess
path_x_side1 = np.linspace(-1, 3, 50)
path_y_side1 = np.linspace(0, 1.5, 50)
path_z_side1 = [energy_landscape_excess(path_x_side1[i], path_y_side1[i]) + 0.05 fo
path_x_side2 = np.linspace(-3, -1, 50)
path_y_side2 = np.linspace(0, -1.5, 50)
path_z_side2 = [energy_landscape_excess(path_x_side2[i], path_y_side2[i]) + 0.05 fo

# Plot main and side paths
ax1.plot(path_x, path_y, path_z_normal, 'r-', linewidth=4, label='Main Reaction Pat
ax2.plot(path_x, path_y, path_z_excess, 'r-', linewidth=4, label='Main Reaction Pat
ax2.plot(path_x_side1, path_y_side1, path_z_side1, 'r--', linewidth=3, label='Side
ax2.plot(path_x_side2, path_y_side2, path_z_side2, 'r:', linewidth=3, label='Side R

# --- Key points ---
def add_points(ax, landscape, label_suffix="", add_side=False):
    react_x, ts_x, prod_x = -2.5, 0, 2.5
    react_y, ts_y, prod_y = 0, 0, 0
    react_z = landscape(react_x, react_y) + 0.1
    ts_z = landscape(ts_x, ts_y) + 0.1
    prod_z = landscape(prod_x, prod_y) + 0.1
    ax.scatter([react_x], [react_y], [react_z], color='green', s=200, label='Reacta
    ax.scatter([ts_x], [ts_y], [ts_z], color='orange', s=200, label='Transition Sta
    ax.scatter([prod_x], [prod_y], [prod_z], color='purple', s=200, label='Products
    if add_side:
        sp1_x, sp1_y, sp2_x, sp2_y = 2.5, 1.5, -2.5, -1.5
        sp1_z = landscape(sp1_x, sp1_y) + 0.1
```

```
        sp2_z = landscape(sp2_x, sp2_y) + 0.1
        ax.scatter([sp1_x], [sp1_y], [sp1_z], color='magenta', s=200, label='Homoco
        ax.scatter([sp2_x], [sp2_y], [sp2_z], color='yellow', s=200, label='Dehalog

add_points(ax1, energy_landscape_normal)
add_points(ax2, energy_landscape_excess, add_side=True)

# --- Catalysts (Pd/C particles) ---
for i in range(8):
    x_pos = np.random.uniform(-3, 3)
    y_pos = np.random.uniform(-3, 3)
    z_pos = energy_landscape_normal(x_pos, y_pos) + 0.3
    ax1.scatter(x_pos, y_pos, z_pos, color='silver', s=80, alpha=0.8, edgecolor='bl

for i in range(30):
    x_pos = np.random.uniform(-3, 3)
    y_pos = np.random.uniform(-3, 3)
    z_pos = energy_landscape_excess(x_pos, y_pos) + 0.3
    ax2.scatter(x_pos, y_pos, z_pos, color='silver', s=80, alpha=0.7, edgecolor='bl

# --- Labels, annotations, layout ---
for ax, title, angle in zip([ax1, ax2],
                            ['Normal Catalyst (0.005g Pd/C)', 'Excess Catalyst (0.0
                            [(30, 40), (25, 55)]):
    ax.set_xlabel('Reaction Coordinate X', fontsize=14, labelpad=10)
    ax.set_ylabel('Reaction Coordinate Y', fontsize=14, labelpad=10)
    ax.set_zlabel('Energy (kcal/mol)', fontsize=14, labelpad=10)
    ax.set_title(title, fontsize=16, pad=20)
    ax.tick_params(axis='x', labelsize=11)
    ax.tick_params(axis='y', labelsize=11)
    ax.tick_params(axis='z', labelsize=11)
    ax.set_zlim(z_min, z_max + 2)
    ax.view_init(elev=angle[0], azim=angle[1])

# Add explanatory text
props = dict(boxstyle='round', facecolor='white', alpha=0.8)
ax1.text(-4, 4, z_max, "Normal catalyst loading (0.005g Pd/C):\n- Single pathway\n-
         fontsize=12, color='black', bbox=props)
ax2.text(-4, 4, z_max, "Excess catalyst loading (0.05g Pd/C):\n- Multiple pathways\
         fontsize=12, color='black', bbox=props)

# Global title and layout
fig.suptitle('Comparison of Energy Landscapes for Suzuki-Miyaura Coupling\nwith Dif
             fontsize=18, y=0.95)

ax1.legend(loc='upper right', fontsize=9)
ax2.legend(loc='upper right', fontsize=9)

fig.text(0.5, 0.05, "→ Increasing catalyst loading leads to multiple reaction pathw
         ha='center', fontsize=14, bbox=dict(facecolor='lightyellow', alpha=0.8, bo

plt.subplots_adjust(top=0.88, bottom=0.1, left=0.05, right=0.88, wspace=0.25)
plt.savefig('suzuki_coupling_fixed.png', dpi=300, bbox_inches='tight')
plt.show()

print ("The comparison of the two energy landscapes illustrates how catalyst loadin
```

```
print("="*80)

import numpy as np
import matplotlib.pyplot as plt
from mpl_toolkits.mplot3d import Axes3D
from matplotlib.patches import FancyArrowPatch
from mpl_toolkits.mplot3d import proj3d

# Set random seed for reproducibility
np.random.seed(42)

# Create a figure with a 3D projection
fig = plt.figure(figsize=(16, 14))  # Increased figure size
ax = fig.add_subplot(111, projection='3d')

# Define molecule positions
iodovanillin_pos = np.array([-2, 0, 0])
boronic_acid_pos = np.array([2, 0, 0])
pd_normal_pos = np.array([0, 0, 0])
product_pos = np.array([0, 3, 0])
byproduct1_pos = np.array([-1, -2, 0])
byproduct2_pos = np.array([1, -2, 0])

# Create a class for 3D arrows - updated for newer matplotlib versions
class Arrow3D(FancyArrowPatch):
    def __init__(self, xs, ys, zs, *args, **kwargs):
        super().__init__((0,0), (0,0), *args, **kwargs)
        self._verts3d = xs, ys, zs

    def do_3d_projection(self, renderer=None):
        xs3d, ys3d, zs3d = self._verts3d
        xs, ys, zs = proj3d.proj_transform(xs3d, ys3d, zs3d, ax.get_proj())
        self.set_positions((xs[0], ys[0]), (xs[1], ys[1]))
        return np.min(zs)

# Create molecule representations (simplified as collections of spheres)
def create_molecule(center, radius, n_atoms, color, alpha=0.7):
    """
    Creates a molecule representation as a collection of spheres.

    Parameters:
    - center: The central position of the molecule
    - radius: The approximate radius of the molecule
    - n_atoms: Number of atoms (spheres) to represent the molecule
    - color: Color of the molecule
    - alpha: Transparency level

    Returns:
    - List of 3D positions for each atom in the molecule
    """
    positions = []
    for i in range(n_atoms):
        # Create a somewhat random but structured molecule shape
        theta = np.random.uniform(0, 2*np.pi)
        phi = np.random.uniform(0, np.pi)
```

```
        r = radius * np.random.uniform(0.5, 1.0)
        x = center[0] + r * np.sin(phi) * np.cos(theta)
        y = center[1] + r * np.sin(phi) * np.sin(theta)
        z = center[2] + r * np.cos(phi)
        positions.append((x, y, z))
    return positions

# Create molecules with appropriate sizes and atom counts
iodovanillin = create_molecule(iodovanillin_pos, 0.8, 15, 'blue')       # 5-iodovan
boronic_acid = create_molecule(boronic_acid_pos, 0.6, 10, 'green')      # Boronic a
product = create_molecule(product_pos, 0.8, 20, 'purple')               # Product i
byproduct1 = create_molecule(byproduct1_pos, 0.7, 12, 'orange')         # Homocoupl
byproduct2 = create_molecule(byproduct2_pos, 0.7, 12, 'red')            # Dehalogen

# Plot molecules with appropriate colors and sizes
# Blue: 5-iodovanillin (reactant 1)
for pos in iodovanillin:
    ax.scatter(*pos, color='blue', s=100, alpha=0.7, edgecolor='black', linewidth=0

# Green: 4-methylphenylboronic acid (reactant 2)
for pos in boronic_acid:
    ax.scatter(*pos, color='green', s=80, alpha=0.7, edgecolor='black', linewidth=0

# Purple: Coupled product (desired product)
for pos in product:
    ax.scatter(*pos, color='purple', s=120, alpha=0.7, edgecolor='black', linewidth

# Orange: Homocoupling byproduct (side reaction product)
for pos in byproduct1:
    ax.scatter(*pos, color='orange', s=100, alpha=0.6, edgecolor='black', linewidth

# Red: Dehalogenation byproduct (side reaction product) - changed from yellow to re
for pos in byproduct2:
    ax.scatter(*pos, color='red', s=100, alpha=0.6, edgecolor='black', linewidth=0.

# Plot normal catalyst amount (silver spheres at center)
pd_normal = create_molecule(pd_normal_pos, 0.3, 3, 'silver')
for pos in pd_normal:
    ax.scatter(*pos, color='silver', s=150, alpha=0.9, edgecolor='black', linewidth

# Plot excess catalyst - many more particles scattered around
# This represents the 0.05g Pd/C (10x normal amount)
for i in range(20):
    pos = np.array([
        np.random.uniform(-3, 3),
        np.random.uniform(-3, 3),
        np.random.uniform(-1, 1)
    ])
    ax.scatter(*pos, color='silver', s=80, alpha=0.5, edgecolor='black', linewidth=

# Add reaction pathway arrows to show the flow of the reaction
# Main reaction pathway (reactants → product)
arrow1 = Arrow3D([-1.5, -0.5], [0, 0], [0, 0], mutation_scale=20, lw=2, arrowstyle=
arrow2 = Arrow3D([1.5, 0.5], [0, 0], [0, 0], mutation_scale=20, lw=2, arrowstyle="-
arrow3 = Arrow3D([0, 0], [0.5, 2.5], [0, 0], mutation_scale=20, lw=2, arrowstyle="-
```

```
# Side reaction pathways (with excess catalyst)
arrow4 = Arrow3D([0, -0.5], [0, -1.5], [0, 0], mutation_scale=20, lw=1.5, arrowstyl
arrow5 = Arrow3D([0, 0.5], [0, -1.5], [0, 0], mutation_scale=20, lw=1.5, arrowstyle

# Add arrows to the plot
for arrow in [arrow1, arrow2, arrow3, arrow4, arrow5]:
    ax.add_artist(arrow)

# Add labels for molecules
ax.text(-2.5, 0, 1, "5-Iodovanillin", color='blue', fontsize=12, fontweight='bold')
ax.text(2, 0, 1, "4-Methylphenylboronic acid", color='green', fontsize=12, fontweig
ax.text(0, 3, 1, "Coupled Product", color='purple', fontsize=12, fontweight='bold')
ax.text(-1.5, -2, 1, "Homocoupling\nByproduct", color='orange', fontsize=12, fontwe
ax.text(1.5, -2, 1, "Dehalogenation\nByproduct", color='red', fontsize=12, fontweig
ax.text(0, 0, 1, "Pd Catalyst", color='black', fontsize=12, fontweight='bold')

# Add a title and explanatory text
ax.set_title('3D Molecular Interaction in Suzuki-Miyaura Coupling\nEffect of Excess

# Add explanatory text boxes - completely repositioned to avoid overlap
props = dict(boxstyle='round', facecolor='white', alpha=0.90)

# Position the color legend in the upper left corner
ax.text(-4, -1,2, "Color Legend:\n" +
       "• Blue: 5-Iodovanillin (reactant)\n" +
       "• Green: 4-Methylphenylboronic acid (reactant)\n" +
       "• Silver: Pd/C catalyst particles\n" +
       "• Purple: Desired coupling product\n" +
       "• Orange: Homocoupling byproduct\n" +
       "• Red: Dehalogenation byproduct",  # Changed to red
       fontsize=12, bbox=props)

# Position the catalyst effect text in the lower right corner
ax.text(0, -1.5, 5, "Catalyst Effect:\n" +  # Moved to higher position
       "• Normal loading (0.005g): Few silver particles\n" +
       "• Excess loading (0.05g): Many silver particles\n" +
       "• Excess catalyst enables side reactions\n" +
       "• Side products form along dotted arrows",
       fontsize=12, bbox=props)

# Set axis labels
ax.set_xlabel('X position', fontsize=14)
ax.set_ylabel('Y position', fontsize=14)
ax.set_zlabel('Z position', fontsize=14)

# Set axis limits - extended to make more room
ax.set_xlim(-5, 5)
ax.set_ylim(-5, 5)
ax.set_zlim(-2, 4)

# Improve the view angle
ax.view_init(elev=25, azim=45)  # Adjusted view angle

# Add a legend for clarity - moved to upper right corner
from matplotlib.lines import Line2D
legend_elements = [
```

```
    Line2D([0], [0], marker='o', color='w', markerfacecolor='blue', markersize=10,
    Line2D([0], [0], marker='o', color='w', markerfacecolor='green', markersize=10,
    Line2D([0], [0], marker='o', color='w', markerfacecolor='silver', markersize=10
    Line2D([0], [0], marker='o', color='w', markerfacecolor='purple', markersize=10
    Line2D([0], [0], marker='o', color='w', markerfacecolor='orange', markersize=10
    Line2D([0], [0], marker='o', color='w', markerfacecolor='red', markersize=10, l
    Line2D([0], [0], color='blue', lw=2, label='Main Reaction Path'),
    Line2D([0], [0], color='orange', linestyle=':', lw=2, label='Side Reaction Path
]
# Moved legend to a different position
ax.legend(handles=legend_elements, loc='upper right', bbox_to_anchor=(1.0, 0.5), fo

# Add grid for better spatial reference
ax.grid(True, linestyle='--', alpha=0.3)

# Use a custom layout adjustment instead of tight_layout to avoid warnings
plt.subplots_adjust(left=0.05, right=0.95, top=0.90, bottom=0.05)

# Save the figure with high resolution
plt.savefig('suzuki_coupling_molecular_interaction.png', dpi=300, bbox_inches='tigh

# Show the plot
plt.show()

print ("The 3D plot shows how 5-Iodovanillin (blue dots) and 4-Methylphenylboronic
```

==
SUZUKI-MIYAURA COUPLING REACTION ANALYSIS
5-Iodovanillin + 4-Methylphenylboronic Acid → 5-(4-Methylphenyl)vanillin
==

REACTANT QUANTITIES:
5-Iodovanillin: 0.252 g = 0.906 mmol
4-Methylphenylboronic acid: 0.170 g = 1.250 mmol
K_2CO_3 (base): 0.403 g = 2.916 mmol

LIMITING REAGENT: 5-Iodovanillin
Theoretical moles of product: 0.906 mmol
Theoretical yield: 0.207 g

==
PALLADIUM CATALYST ANALYSIS
==

ACTUAL CATALYST USED: 0.050 g of 10% Pd/C
  - Pd content: 0.0050 g = 0.0470 mmol
  - Catalyst loading: 5.18 mol% Pd

THEORETICAL CATALYST (should have used): 0.005 g of 10% Pd/C
  - Pd content: 0.0005 g = 0.0470 mmol
  - Catalyst loading: 5.18 mol% Pd

EXCESS CATALYST FACTOR: 10.0x
You used 10.0 times more catalyst than intended!

==
EFFECTS OF EXCESS Pd/C CATALYST (10x excess)
==

Economic Impact:
  - Wasted $2.25 in catalyst (est. $50/g for 10% Pd/C)
  - Palladium is expensive; excess significantly increases cost

Reaction Kinetics:
  - Faster initial rate due to more active sites
  - May reach completion sooner
  - Diminishing returns: rate plateaus beyond optimal loading

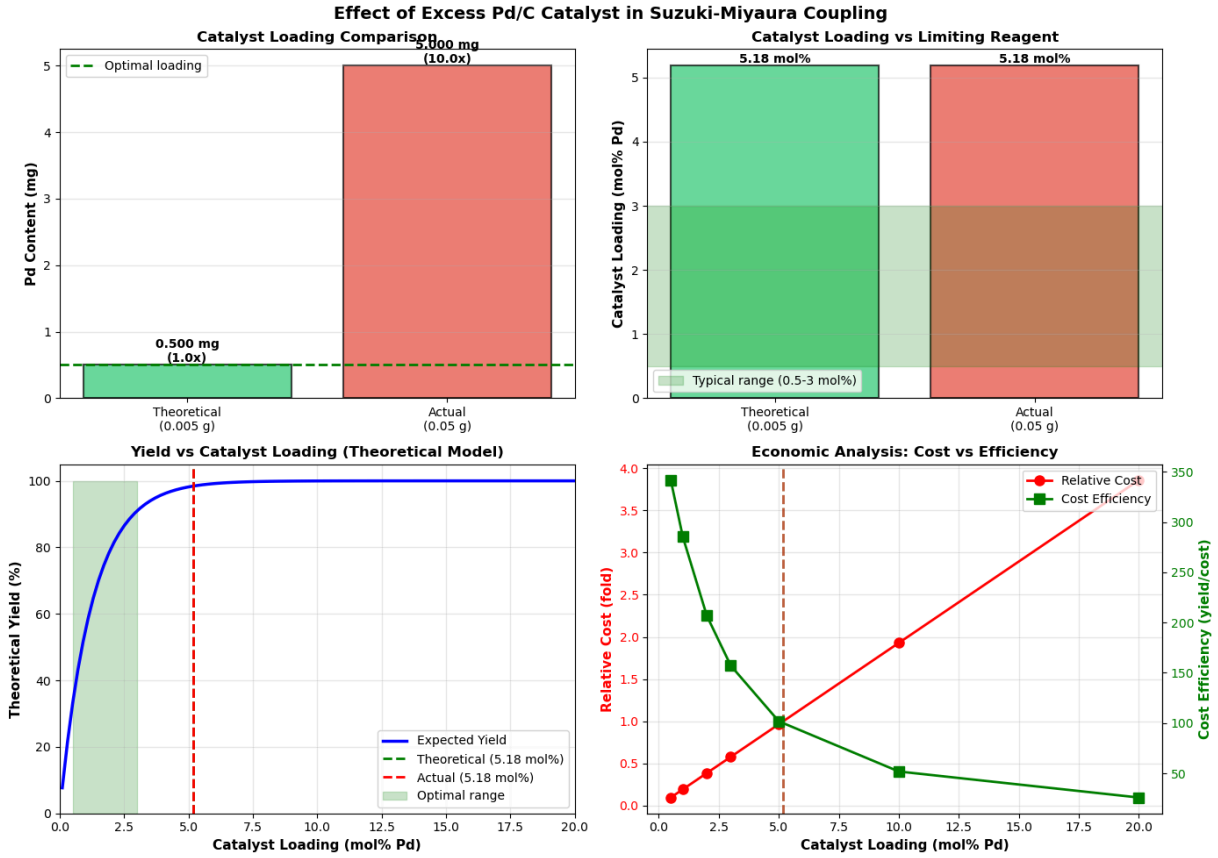
Side Reactions & Selectivity:
  - Increased risk of homocoupling (Ar-Ar from boronic acid)
  - Potential for over-reduction of substrates
  - May promote dehalogenation without coupling
  - Pd nanoparticle aggregation at high loading

Product Purification:
  - More Pd leaching into product
  - Difficult to remove Pd contamination (ICP-MS detection)
  - Activated carbon filtration less effective
  - Potential product discoloration (Pd black formation)

Mechanistic Considerations:
  - Transmetalation not rate-limiting at high [Pd]

- Reductive elimination may be affected
- Catalyst aggregation reduces active Pd(0) concentration
- Pd(II)/Pd(0) equilibrium shifted

Green Chemistry Violations:
- Violates atom economy principles
- Excessive use of precious metal catalyst
- Increased environmental impact
- Not aligned with sustainable synthesis practices



==
RECOMMENDATIONS FOR FUTURE EXPERIMENTS
==

Aspect	Recommendation
Catalyst Amount	Use 0.005 g (not 0.050 g)
Catalyst Loading	0.5-3 mol% Pd is optimal for most Suzuki couplings
Reaction Time	Monitor by TLC; excess catalyst may speed up but wastes material
Workup Procedure	Add activated carbon treatment to remove Pd contamination
Quality Control	Check Pd content in product by ICP-MS or similar
Cost Optimization	Save $2.25 per reaction

==
MECHANISTIC INSIGHTS (Based on Literature)
==

SUZUKI-MIYAURA CATALYTIC CYCLE:
1. Oxidative Addition: Ar-I + Pd(0) → Ar-Pd(II)-I
2. Transmetalation: Ar-Pd(II)-I + Ar'-$B(OH)_2$ + Base → Ar-Pd(II)-Ar' + Base•I + $B(OH)_3$
3. Reductive Elimination: Ar-Pd(II)-Ar' → Ar-Ar' + Pd(0)

EFFECT OF 10x EXCESS CATALYST:

Rate-Determining Step Considerations (Mondal et al., 2018):
  - At low [Pd]: Oxidative addition often rate-limiting
  - At high [Pd]: Transmetalation or reductive elimination becomes limiting
  - Excess catalyst doesn't proportionally increase rate

Practical Implications (Miyaura & Suzuki, 1995; Hanley, 2014):
  - Optimal loading: 0.5-5 mol% for most aryl halides
  - Aryl iodides (like 5-iodovanillin) are highly reactive
  - 10x excess shows diminishing returns in yield
  - Increases purification challenges significantly

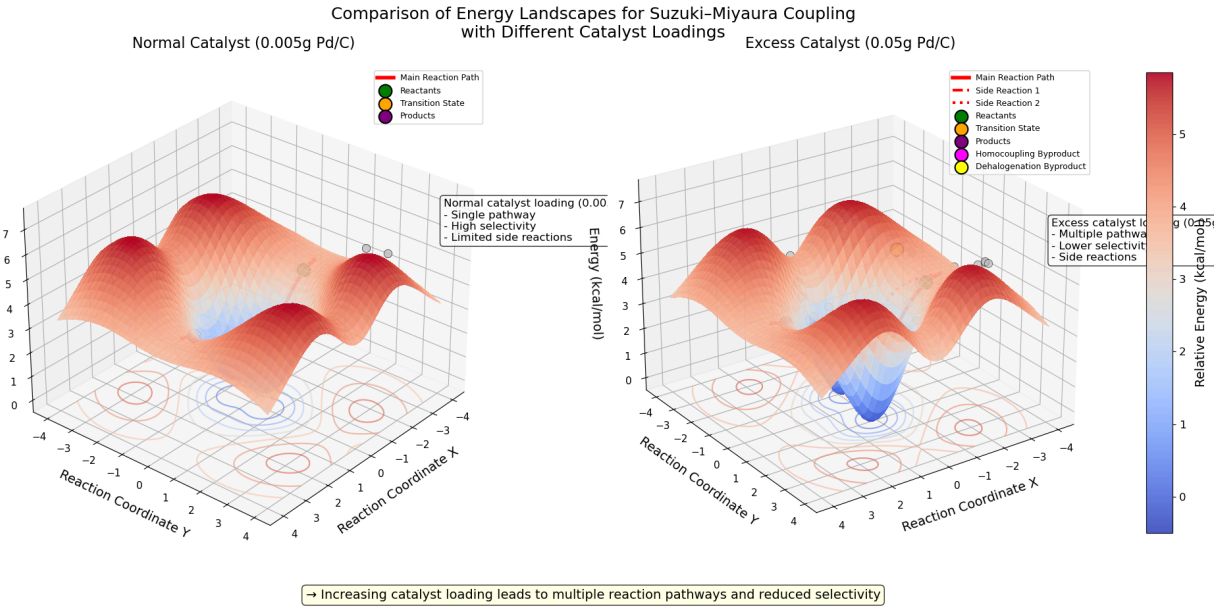
Green Chemistry Perspective (Palesch et al., 2019):
  - Suzuki coupling already considered 'green' (water solvent, mild conditions)
  - Excess precious metal catalyst undermines sustainability
  - Proper catalyst optimization is key to green synthesis

==
SUMMARY
==

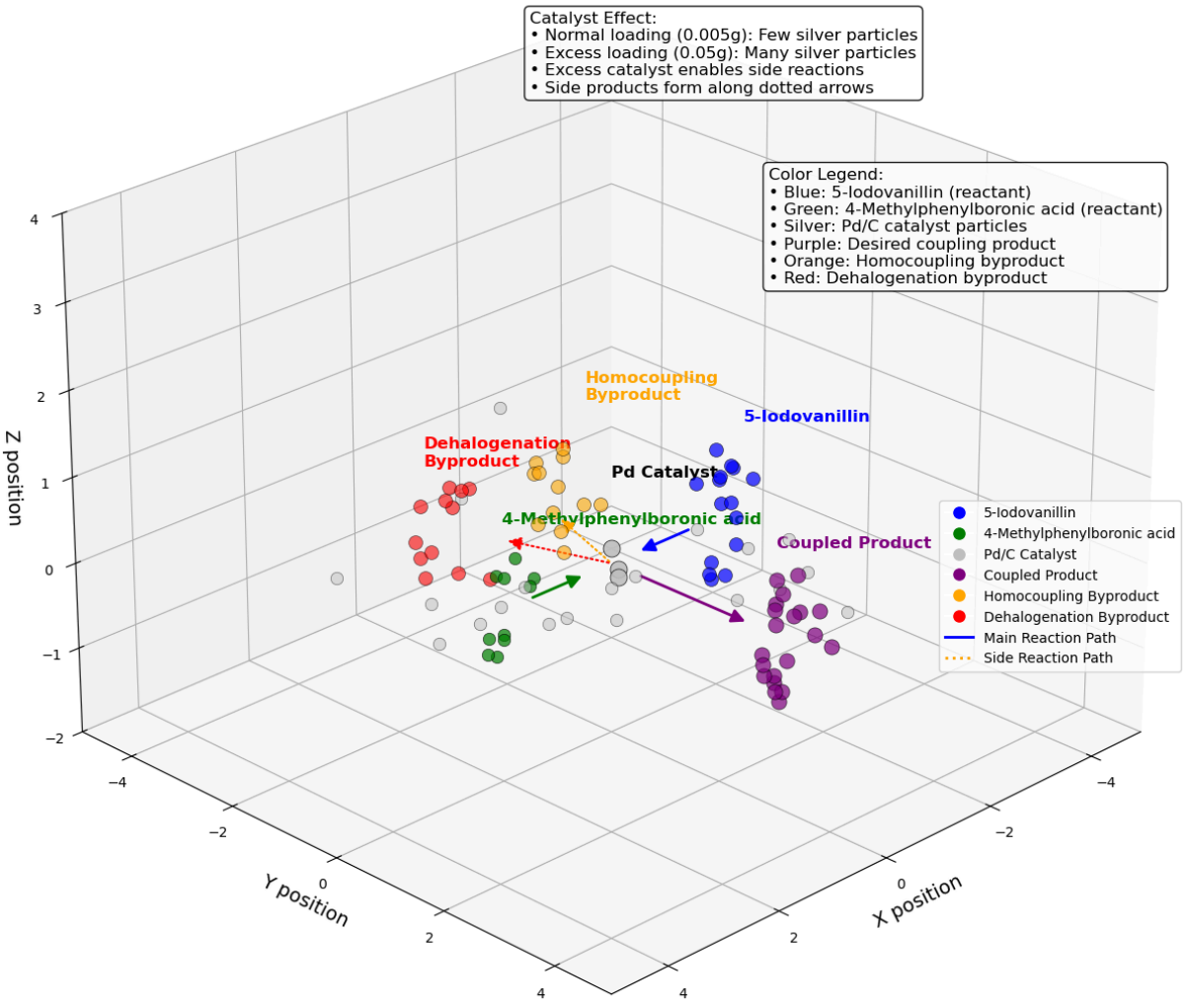
Parameter	Value
Limiting Reagent	5-Iodovanillin
Theoretical Yield (g)	0.207
Theoretical Yield (mmol)	0.906
Pd/C Used (g)	0.050
Pd Content (mg)	5.000
Catalyst Loading (mol%)	5.18
Excess Factor	10.0x
Estimated Cost Increase ($)	2.25
Typical Optimal Loading (mol%)	0.5-3.0

==

CONCLUSION:
Using 10x excess Pd/C catalyst likely provided minimal benefit to yield
while significantly increasing cost and purification difficulties.
Future reactions should use the calculated 0.005 g amount.
==========



The comparison of the two energy landscapes illustrates how catalyst loading influen
ces both selectivity and efficiency in the Suzuki–Miyaura coupling. At the normal ca
talyst amount (0.005 g Pd/C), the reaction follows a single, well-defined energy pat
hway from reactants to the coupled product, with a moderate energy barrier and limit
ed side reactions. In contrast, when the catalyst loading is increased tenfold (0.05
g Pd/C), the surface becomes more complex, showing multiple valleys that correspond
to competing reaction routes such as homocoupling and dehalogenation. These addition
al pathways imply that excess palladium provides too many active sites, allowing uni
ntended transformations to occur in parallel with the desired coupling. It seems to
me that the flatter energy profile and lower barriers under excess catalyst conditio
ns lead to reduced selectivity and a mixture of products, which explains why high ca
talyst loading can yield impure solids or by-products even when conversion appears c
omplete.
==========



The 3D plot shows how 5-Iodovanillin (blue dots) and 4-Methylphenylboronic acid (gre
en dots) react with the help of a Pd/C catalyst (gray dots) to form a desired couple
d product (purple dots), while using excess catalyst creates unwanted homocoupling
(orange dots) and dehalogenation (red dots) byproducts along alternative reaction pa
thways.

In [ ]: