```
import numpy as np
import matplotlib.pyplot as plt
import ipywidgets as widgets
from IPython.display import display, clear_output
import io
import base64
from datetime import datetime

# Global variables to store results
filtered_signal = None
original_signal = None
x_values = None

def parse_function(func_str, x):
    """Parse and evaluate the function string"""
    try:
        # Replace common math expressions
        func_str = func_str.replace('sin', 'np.sin')
        func_str = func_str.replace('cos', 'np.cos')
        func_str = func_str.replace('tan', 'np.tan')
        func_str = func_str.replace('pi', 'np.pi')
        func_str = func_str.replace('exp', 'np.exp')
        func_str = func_str.replace('sqrt', 'np.sqrt')

        # Evaluate the function
        result = eval(func_str)
        return result
    except Exception as e:
        raise ValueError(f"Function parse error: {str(e)}")

def apply_fir_filter(signal, fs, fc=4.0, M=51):
    """Apply FIR filter to the signal"""
    n = np.arange(M)
    mid = (M - 1) / 2.0

    # Ideal low-pass filter (sinc function)
    h_ideal = (2 * fc / fs) * np.sinc(2 * fc * (n - mid) / fs)

    # Apply Hamming window
    window = np.hamming(M)
    h = h_ideal * window

    # Normalize
    h = h / np.sum(h)

    # Apply filter
    filtered = np.convolve(signal, h, mode='same')
```

```
    return filtered

def on_filter_click(b):
    """Handle filter button click"""
    global filtered_signal, original_signal, x_values

    with output:
        clear_output(wait=True)

        try:
            # Get function string
            func_str = function_input.value.strip()
            if not func_str:
                print("⚠ Please enter a function!")
                return

            # Signal parameters
            N = int(samples_input.value)
            fs = int(fs_input.value)
            fc = float(cutoff_input.value)
            M = int(filter_order_input.value)

            # Validation
            if N < 10:
                print("⚠ Samples (N) must be at least 10!")
                return
            if M >= N:
                print(f"⚠ Filter Order (M={M}) must be less than Samples
(N={N})!")
                return
            if fc >= fs/2:
                print(f"⚠ Cutoff Frequency ({fc} Hz) must be less than
Nyquist frequency ({fs/2} Hz)!")
                return

            dx = 1.0 / N
            x_values = np.arange(0.0, 1.0, dx)  # Fixed: removed +dx/2 to
ensure exact N points

            # Generate signal
            original_signal = parse_function(func_str, x_values)

            # Apply FIR filter
            filtered_signal = apply_fir_filter(original_signal, fs, fc, M)

            # Create plots
```

```
            fig, (ax1, ax2) = plt.subplots(2, 1, figsize=(12, 8))

            # Original signal
            ax1.plot(x_values, original_signal, 'b-o', markersize=3,
linewidth=1.5, label='Original Signal')
            ax1.set_title('Original Signal', fontsize=14, fontweight='bold')
            ax1.set_xlabel('Time (s)', fontsize=12)
            ax1.set_ylabel('Amplitude', fontsize=12)
            ax1.grid(True, alpha=0.3)
            ax1.legend(fontsize=10)

            # Filtered signal
            ax2.plot(x_values, filtered_signal, 'r-o', markersize=3,
linewidth=1.5, label='Filtered Signal (FIR)')
            ax2.set_title(f'Filtered Signal (Cutoff: {fc} Hz)', fontsize=14,
fontweight='bold')
            ax2.set_xlabel('Time (s)', fontsize=12)
            ax2.set_ylabel('Amplitude', fontsize=12)
            ax2.grid(True, alpha=0.3)
            ax2.legend(fontsize=10)

            plt.tight_layout()
            plt.show()

            print("☑ Filtration completed successfully!")
            print(f"📊 Samples: {N}, Sampling Rate: {fs} Hz, Cutoff: {fc}
Hz, Filter Order: {M}")

            # Enable download buttons
            download_txt_btn.disabled = False
            download_xls_btn.disabled = False

        except Exception as e:
            print(f"❌ Error: {str(e)}")
            print("Please check your function syntax.")
            print("Example: 2*sin(2*pi*x) + (1/6)*cos(34*pi*x)")

def download_txt(b):
    """Download results as TXT file"""
    global filtered_signal, original_signal, x_values

    if filtered_signal is None:
        print("⚠ Please run filtration first!")
        return

    # Create TXT content
    txt_content = "# FIR Filter Results\n"
```

```
    txt_content += f"# Generated: {datetime.now().strftime('%Y-%m-%d
%H:%M:%S')}\n"
    txt_content += f"# Function: {function_input.value}\n"
    txt_content += f"# Samples: {samples_input.value}, Sampling Rate:
{fs_input.value} Hz\n"
    txt_content += f"# Cutoff Frequency: {cutoff_input.value} Hz, Filter
Order: {filter_order_input.value}\n"
    txt_content += "#" + "="*60 + "\n\n"
    txt_content += "Time(s)\t\tOriginal\t\tFiltered\n"
    txt_content += "-"*60 + "\n"

    for i in range(len(x_values)):
        txt_content +=
f"{x_values[i]:.6f}\t\t{original_signal[i]:.6f}\t\t{filtered_signal[i]:.6f}\
n"

    # Save file
    filename =
f"fir_filter_results_{datetime.now().strftime('%Y%m%d_%H%M%S')}.txt"
    with open(filename, 'w') as f:
        f.write(txt_content)

    print(f"☑ TXT file saved: {filename}")
    print(f"🗀 Download it from Colab's file browser (left panel)")

def download_xls(b):
    """Download results as Excel file"""
    global filtered_signal, original_signal, x_values

    if filtered_signal is None:
        print("⚠ Please run filtration first!")
        return

    try:
        import pandas as pd

        # Create DataFrame
        df = pd.DataFrame({
            'Time (s)': x_values,
            'Original Signal': original_signal,
            'Filtered Signal': filtered_signal
        })

        # Save to Excel
        filename =
f"fir_filter_results_{datetime.now().strftime('%Y%m%d_%H%M%S')}.xlsx"
        df.to_excel(filename, index=False, sheet_name='FIR Filter Results')
```

```
        print(f"☑ Excel file saved: {filename}")
        print(f"🗀 Download it from Colab's file browser (left panel)")

    except ImportError:
        print("⚠ Installing openpyxl for Excel support...")
        import subprocess
        subprocess.check_call(['pip', 'install', '-q', 'openpyxl'])
        download_xls(b)  # Retry after installation

# Create UI components
print("🎛 FIR FILTER APPLICATION")
print("="*60)

# Function input
function_input = widgets.Text(
    value='2*sin(2*pi*x) + (1/6)*cos(34*pi*x)',
    description='Function:',
    style={'description_width': '120px'},
    layout=widgets.Layout(width='600px')
)

# Parameter inputs
samples_input = widgets.IntText(
    value=64,
    description='Samples (N):',
    style={'description_width': '120px'},
    layout=widgets.Layout(width='300px')
)

fs_input = widgets.IntText(
    value=64,
    description='Sampling Rate (Hz):',
    style={'description_width': '120px'},
    layout=widgets.Layout(width='300px')
)

cutoff_input = widgets.FloatText(
    value=4.0,
    description='Cutoff Freq (Hz):',
    style={'description_width': '120px'},
    layout=widgets.Layout(width='300px')
)

filter_order_input = widgets.IntText(
    value=51,
    description='Filter Order (M):',
```

```
    style={'description_width': '120px'},
    layout=widgets.Layout(width='300px')
)

# Buttons
filter_btn = widgets.Button(
    description='🔄 Start Filtration',
    button_style='primary',
    layout=widgets.Layout(width='200px', height='40px'),
    style={'font_weight': 'bold'}
)

download_txt_btn = widgets.Button(
    description='📥 Download TXT',
    button_style='success',
    layout=widgets.Layout(width='200px', height='40px'),
    disabled=True
)

download_xls_btn = widgets.Button(
    description='📥 Download XLS',
    button_style='success',
    layout=widgets.Layout(width='200px', height='40px'),
    disabled=True
)

# Output area
output = widgets.Output()

# Connect button events
filter_btn.on_click(on_filter_click)
download_txt_btn.on_click(download_txt)
download_xls_btn.on_click(download_xls)

# Display UI
display(widgets.VBox([
    widgets.HTML('<h3>📝 Signal Function</h3>'),
    function_input,
    widgets.HTML('<h3>⚙️ Filter Parameters</h3>'),
    widgets.HBox([samples_input, fs_input]),
    widgets.HBox([cutoff_input, filter_order_input]),
    widgets.HTML('<h3>🎮 Controls</h3>'),
    widgets.HBox([filter_btn, download_txt_btn, download_xls_btn]),
    widgets.HTML('<hr>'),
    output
]))
```

```
print("\n💡 Tips:")
print("  • Use 'x' as the variable in your function")
print("  • Write naturally: sin, cos, tan, pi, exp, sqrt")
print("  • Example: 2*sin(2*pi*x) + (1/6)*cos(34*pi*x)")
print("  • Example: 5*sin(10*pi*x) + 3*cos(20*pi*x)")
print("  • Click 'Start Filtration' to see results")
print("  • Download files from the left sidebar (📁 Files)")
print("="*60)
```