An example of complex signal analysis

For the following 50 Hz waveforms:

$$v_1 = 60 \times \sin(\omega t)$$

$$v_2 = 85 \times \sin\left(\omega t + \frac{\pi}{6}\right)$$

Evaluate the following and show the phasor diagrams of the results. Plot the resultant waveforms over two complete cycles:

$$v_1 + v_2$$

$$v_1 - v_2$$

$$v_2 - v_1$$

Solution:

The waveforms v_1 and v_2 can be represented as two vectors $u_1 = (60, 0)$ and $u_2 = (85\cos(\pi/6), 85\sin(\pi/6))$ rotating synchronously on the same frame $\sin(\omega t)$. For the sum of two vectors, we get:

$$u_1 + u_2 = (60, 0) + (85\cos(\pi/6), 85\sin(\pi/6)) = (60 + 85\cos(\pi/6), 85\sin(\pi/6))$$

$$|u_1 + u_2| = \sqrt{(60 + 85\cos(\pi/6))^2 + (85\sin(\pi/6))^2}$$

$$\arg(u_1 + u_2) = \mathrm{atan2}(85\sin(\pi/6), 60 + 85\cos(\pi/6))$$

where for calculating the vector angle in radians we used the function $\mathrm{atan2}(y, x)$. Similarly, we get:

$$u_1 - u_2 = (60, 0) - (85\cos(\pi/6), 85\sin(\pi/6)) = (60 - 85\cos(\pi/6), -85\sin(\pi/6))$$

$$|u_1 - u_2| = \sqrt{(60 - 85\cos(\pi/6))^2 + (85\sin(\pi/6))^2}$$

$$\arg(u_1 - u_2) = \mathrm{atan2}(-85\sin(\pi/6), 60 - 85\cos(\pi/6))$$

$$u_2 - u_1 = (85\cos(\pi/6) - 60, 85\sin(\pi/6))$$

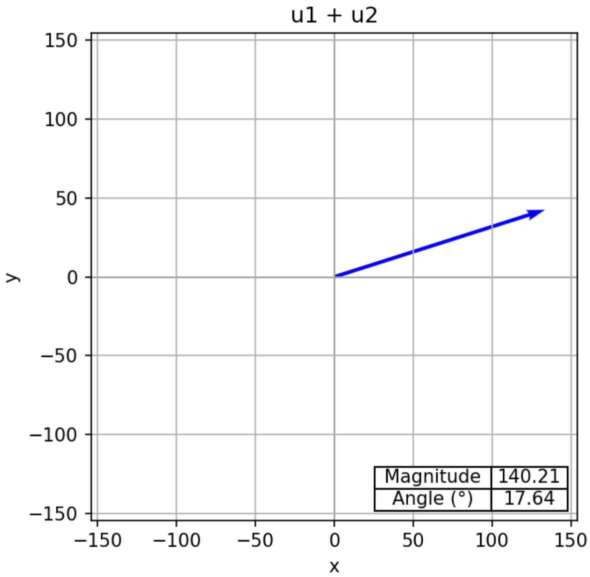
$$|u_2 - u_1| = \sqrt{(60 - 85\cos(\pi/6))^2 + (85\sin(\pi/6))^2}$$

$$\arg(u_2 - u_1) = \mathrm{atan2}(85\sin(\pi/6), 85\cos(\pi/6) - 60)$$

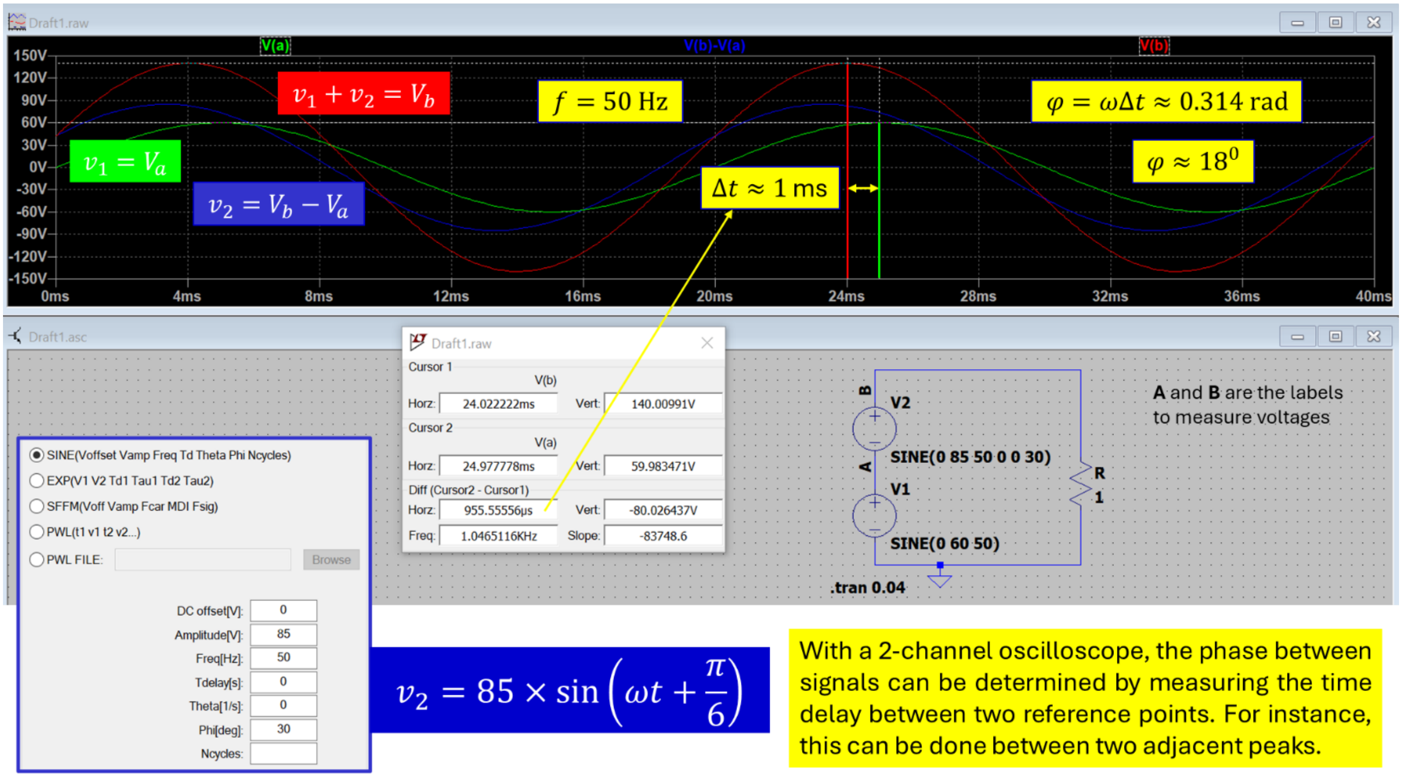
Numerical calculations:

```
1. u1 + u2: (133.612, 42.5)
2. |u1 + u2|: 140.208
3. arg(u1 + u2): 0.308  rad
4. arg(u1 + u2): 17.645  deg
```

$$v_1 + v_2 = 140.208 \times \sin(\omega t + 17.645^0)$$



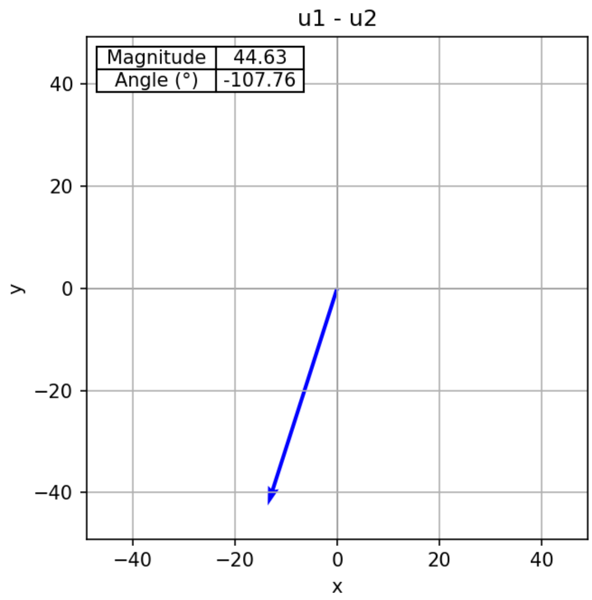
LTspice simulations

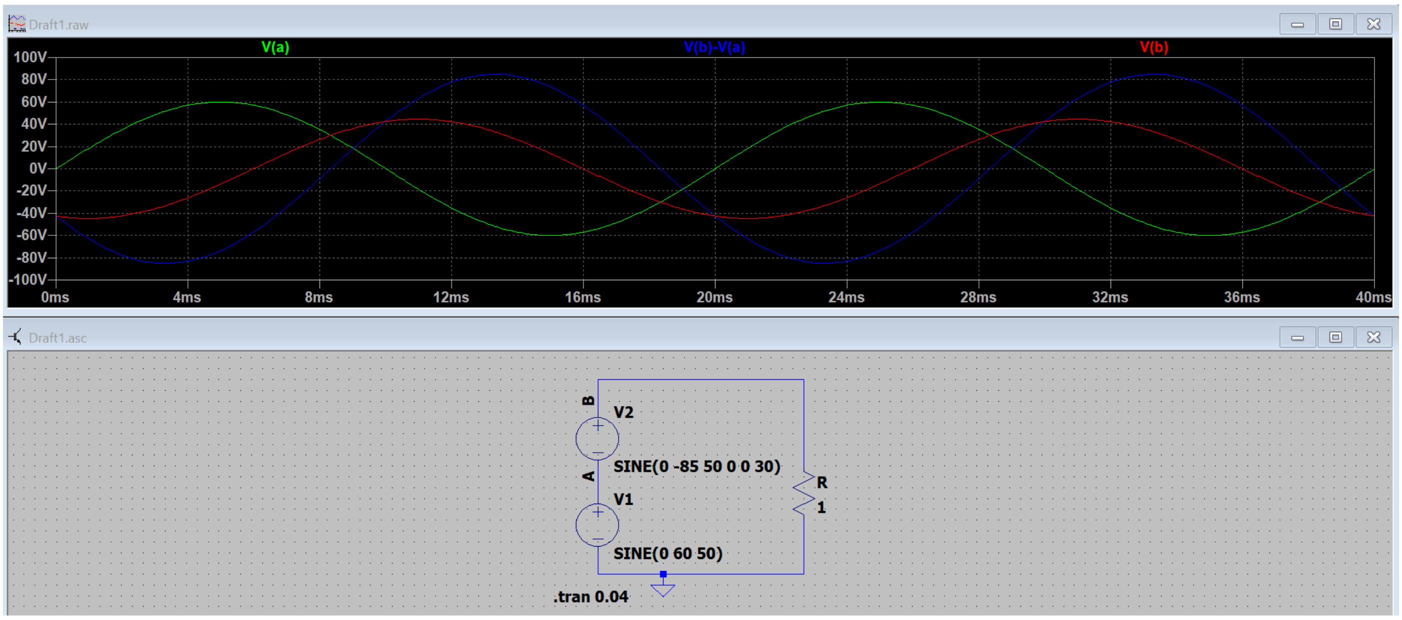


Numerical calculations:

```
1. u1 - u2: (-13.612, -42.5)
2. |u1 - u2|: 44.627
3. arg(u1 - u2): -1.881  rad
4. arg(u1 - u2): -107.759  deg
```

$$v_1 - v_2 = 44.627 \times \sin\left(\omega t - 107.759^0\right)$$

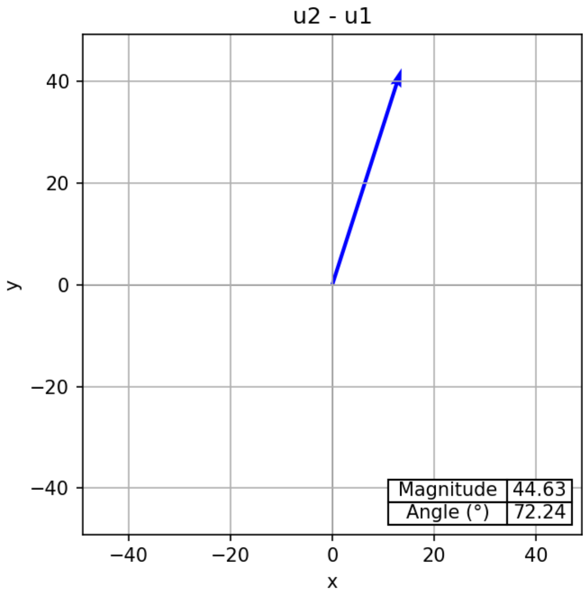


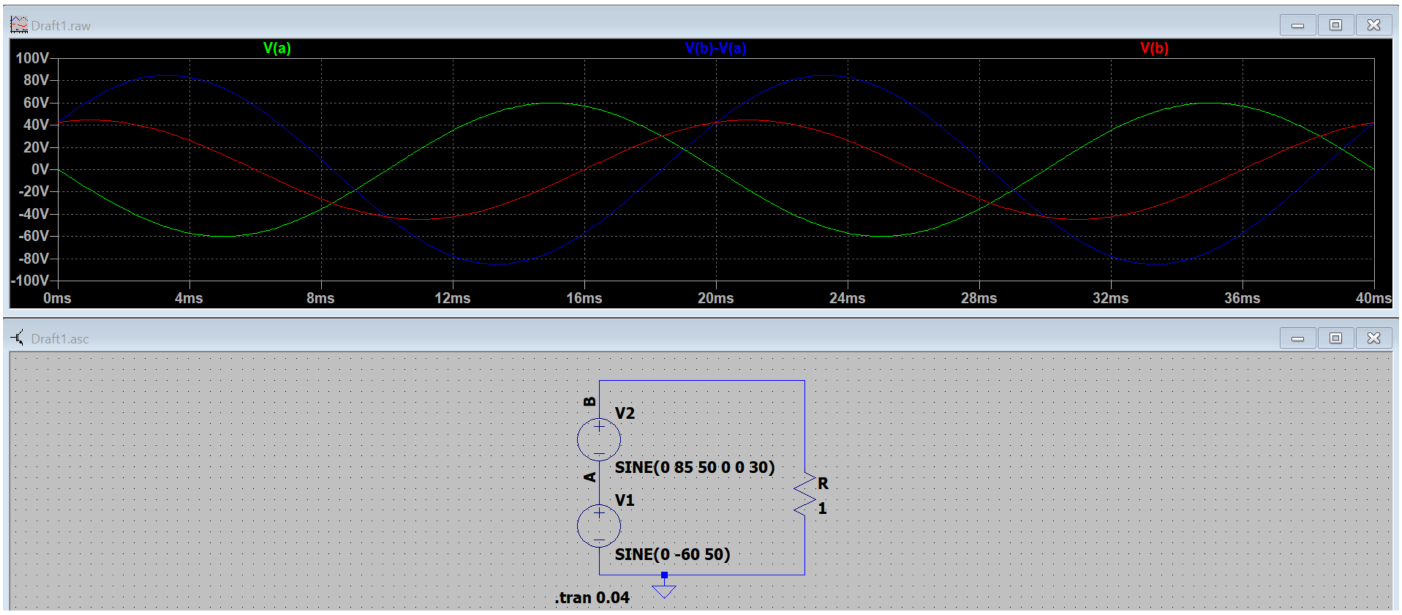


Numerical calculations:

```
1. u2 - u1: (13.612, 42.5)
2. |u2 - u1|: 44.627
3. arg(u2 - u1): 1.261  rad
4. arg(u2 - u1): 72.241  deg
```

$$v_2 - v_1 = 44.627 \times \sin\left(\omega t + 72.241^0\right)$$





Python code:

```
#
# Sum and difference of two vectors u1 and u2
#
# Dr. Dmitriy Makhnovskiy, City College Plymouth, England
# 12.04.2024
#

import math
import matplotlib.pyplot as plt

# Number of decimal places when calculating real values
accuracy = 3

# Vector 1
phase1 = 0.0  # angle in radians (modify if you use degrees)
mag1 = 60  # magnitude
u1 = (mag1 * math.cos(phase1), mag1 * math.sin(phase1))  # vector

# Vector 2
phase2 = math.pi/6  # angle in radians (modify if you use degrees)
mag2 = 85  # magnitude
u2 = (mag2 * math.cos(phase2), mag2 * math.sin(phase2))  # vector

def vector_sum(u1, u2):
    x = round(u1[0] + u2[0], accuracy)
    y = round(u1[1] + u2[1], accuracy)
    magnitude = round(math.sqrt(x**2 + y**2), accuracy)
    angle = math.atan2(y, x)
    angle_rad = round(angle, accuracy)
    angle_deg = round(angle * 180.0 / math.pi, accuracy)
    return (x, y), magnitude, angle_rad, angle_deg

def vector_difference(u1, u2):
    x = round(u1[0] - u2[0], accuracy)
    y = round(u1[1] - u2[1], accuracy)
    magnitude = round(math.sqrt(x**2 + y**2), accuracy)
    angle = math.atan2(y, x)
    angle_rad = round(angle, accuracy)
    angle_deg = round(angle * 180.0 / math.pi, accuracy)
    return (x, y), magnitude, angle_rad, angle_deg


def draw_vector_with_arguments(vector, magnitude, angle_degrees, title='', color='blue'):
    x, y = vector

    # Set the axis limits to be proportional to the magnitude of the vector
    max_limit = magnitude * 1.1  # Adding some padding
    plt.figure()
    ax = plt.gca()
    ax.set_xlim([-max_limit, max_limit])
    ax.set_ylim([-max_limit, max_limit])
    ax.set_aspect('equal')
    ax.axhline(0, color='black', linewidth=0.5, zorder=0)
    ax.axvline(0, color='black', linewidth=0.5, zorder=0)
    plt.grid(True, zorder=0)

    ax.quiver(0, 0, x, y, angles='xy', scale_units='xy', scale=1, color=color, zorder=1)

    table_data = [
        ["Magnitude", f'{magnitude:.2f}'],
        ["Angle (°)", f'{angle_degrees:.2f}']
    ]

    # Determine the position of the table
    if x >= 0 and y >= 0:
        table_loc = 'lower right'
    elif x < 0 and y >= 0:
        table_loc = 'lower left'
    elif x < 0 and y < 0:
        table_loc = 'upper left'
```

```
    else:
        table_loc = 'upper right'

    table = ax.table(cellText=table_data, loc=table_loc, cellLoc='center', edges='closed', zorder=2)
    table.auto_set_font_size(False)
    table.set_fontsize(10)
    table.auto_set_column_width([0, 1])
    for key, cell in table._cells.items():
        cell.set_facecolor('white')
        cell.set_alpha(1)

    plt.title(title)
    plt.xlabel("x")
    plt.ylabel("y")
    plt.show()

print('')
# u1 + u2
vector, magnitude, angle_rad, angle_deg = vector_sum(u1, u2)
print("u1 + u2:", vector)
print("|u1 + u2|:", magnitude)
print("arg(u1 + u2):", angle_rad, ' rad')
print("arg(u1 + u2):", angle_deg, ' deg')
draw_vector_with_arguments(vector, magnitude, angle_deg, title='u1 + u2', color='blue')

# u1 - u2
vector, magnitude, angle_rad, angle_deg = vector_difference(u1, u2)
print("\nu1 - u2:", vector)
print("|u1 - u2|:", magnitude)
print("arg(u1 - u2):", angle_rad, ' rad')
print("arg(u1 - u2):", angle_deg, ' deg')
draw_vector_with_arguments(vector, magnitude, angle_deg, title='u1 - u2', color='blue')

# u2 - u1
vector, magnitude, angle_rad, angle_deg = vector_difference(u2, u1)
print("\nu2 - u1:", vector)
print("|u2 - u1|:", magnitude)
print("arg(u2 - u1):", angle_rad, ' rad')
print("arg(u2 - u1):", angle_deg, ' deg')
draw_vector_with_arguments(vector, magnitude, angle_deg, title='u2 - u1', color='blue')

```