In [1]:
```
#          ##########################################################################################
#          #                                      IN THE NAME OF ALLAH                              #
#          #                                CABLE FOOT SUSPENSION BRIDGE 02                         #
#          #----------------------------------------------------------------------------------------#
#          #                  THIS PROGRAM WRITTEN BY SALAR DELAVAR GHASHGHAEI (QASHQAI)            #
#          #                           EMAIL: salar.d.ghashghaei@gmail.com                          #
#          ##########################################################################################
```

Suspension Foot Bridge:

A Suspension Foot Bridge is a type of bridge that uses the features of a suspension bridge but is specifically designed for pedestrian use. These bridges are typically used to cross natural obstacles like valleys, rivers, or difficult terrain, whereas regular footbridges are simpler in design and construction.

Features of a Suspension Foot Bridge:

1. Suspension Structure:
   - In a suspension footbridge, steel or wire cables are suspended to support the bridge deck. These cables are anchored to tall towers on each side of the bridge, transferring the load from the deck to the towers.
   - Unlike regular footbridges, which are usually made of simple beams (either concrete or steel), a suspension footbridge's primary structure consists of cables and towers.
2. Design Purpose:
   - These bridges are typically designed for pedestrian use, but compared to regular footbridges, they can span greater distances and are ideal for crossing deep valleys, rivers, or other natural barriers.
   - Suspension footbridges are often built in natural parks, hiking trails, or areas that require crossing significant natural obstacles.
3. Load Characteristics:
   - Suspension footbridges generally carry less load than full-scale suspension bridges (designed for vehicles or trains), as they are only built to support pedestrians. However, they still rely on suspended cables to carry the load.
   - The load on these bridges is primarily pedestrian traffic, but in some cases, they might also support bicycles or animals.
4. Advantages:
   - Longer Spans: Suspension footbridges can cover longer distances than regular pedestrian bridges without requiring additional supports in the middle of the span.
   - Crossing Natural Obstacles: These bridges are ideal for crossing rivers, valleys, and rugged terrain where building traditional bridges is difficult.
   - Aesthetic and Integration with Nature: Suspension footbridges often have a beautiful design that blends well with the natural surroundings, especially in tourist or recreational areas.
5. Construction and Installation:
   - Building a suspension footbridge requires specialized equipment for installing cables and towers. These bridges are commonly used in areas where the terrain is difficult, or access to traditional infrastructure is limited.
   - Installing cables and towers in such challenging locations requires precise engineering and safety considerations.

Examples: Suspension Foot Bridges in Mountainous Areas: Suspension footbridges are commonly found in mountainous regions or nature reserves where they allow pedestrians to cross valleys and rivers. These bridges provide access for both locals and tourists to difficult-to-reach areas.

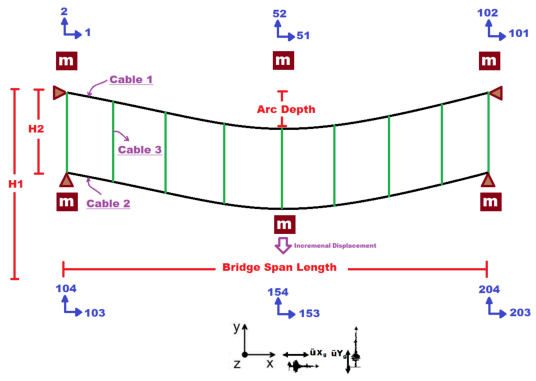
2. Suspension Footbridges in Parks: In some natural parks or recreational areas, suspension footbridges are used to cross rivers or valleys. These bridges serve both a practical purpose and act as tourist attractions.

Differences from Regular Suspension Bridges:

- Lower Load Capacity: While regular suspension bridges are designed to carry vehicles, trains, or heavy loads, and can bear significant stress, suspension footbridges are designed only for pedestrian use and therefore carry much lighter loads.
- Smaller Dimensions: Suspension footbridges are generally smaller in scale and have shorter spans than full-size suspension bridges. They are primarily intended for pedestrians and are rarely used for vehicle traffic.

Conclusion: A Suspension Foot Bridge is a type of suspension bridge specifically designed for pedestrian use. Due to the use of suspended cables, these bridges can span longer distances and are very useful in areas where crossing valleys, rivers, or difficult terrain is necessary.

In [1]:
```
# Load the image
def PLOT_IMAGE(image):
    import matplotlib.pyplot as plt
    import matplotlib.image as mpimg
    image = mpimg.imread(image_path)

    # Display the image
    plt.figure(figsize=(15, 8))
    plt.imshow(image)
    plt.axis('off')  # Hide axes
    plt.show()

image_path = 'OPENSEES_CABLE_SUSPENSION_BRIDGE_02.png'
PLOT_IMAGE(image_path)
```



In [2]:
```
# WEBSITE:
'https://www.semanticscholar.org/paper/CABLE-SUSPENDED-PEDESTRIAN-BRIDGE-DESIGN-FOR-RURAL-Bang/6dd8ec9c0d0ca6059af6d6cbb7e5b6e25aed5bcf'
```

```
# WEBSITE:
'https://www.istockphoto.com/nl/foto/voetgangers-brug-over-een-rivier-van-de-berg-leidt-tot-landelijke-huizen-verdrinken-gm1002006922-270792479'
```

Out[2]:
```
'https://www.istockphoto.com/nl/foto/voetgangers-brug-over-een-rivier-van-de-berg-leidt-tot-landelijke-huizen-verdrinken-gm1002006922-270792479'
```

In [6]:
```
#import the os module
import os
import time
import numpy as np
import openseespy.opensees as op
import matplotlib.pyplot as plt
```

In [7]:
```
#to create a directory at specified path with name "Data"
os.mkdir('C:\\OPENSEESPY_SALAR')
```

```
---------------------------------------------------------------------------
FileExistsError                           Traceback (most recent call last)
Cell In[7], line 2
      1 #to create a directory at specified path with name "Data"
----> 2 os.mkdir('C:\\OPENSEESPY_SALAR')

FileExistsError: [WinError 183] Cannot create a file when that file already exists: 'C:\\OPENSEESPY_SALAR'
```

In [8]:
```
FOLDER_NAME = 'OPENSEES_CABLE_SUSPENSION_BRIDGE'
dir = f"C:\\OPENSEESPY_SALAR\\{FOLDER_NAME}\\"
if not os.path.exists(dir):
    os.makedirs(dir)
```

In [9]:
```
# OUTPUT DATA ADDRESS:
SALAR_DIR = f'C://OPENSEESPY_SALAR//{FOLDER_NAME}//';
```

In [10]:
```
## DELETE ALL FILES IN DIRECTORY
def DELETE_FOLDER_CONTANTS(folder_path):
    import os
    for filename in os.listdir(folder_path):
        file_path = os.path.join(folder_path, filename)
        try:
            if os.path.isfile(file_path) or os.path.islink(file_path):
                os.unlink(file_path)
        except Exception as e:
            print(f"Failed to delete {file_path}. Reason: {e}")
    print("Deletion done")

FOLDER_PATH = f'C:\\OPENSEESPY_SALAR\\{FOLDER_NAME}'  # Specify the folder path
#DELETE_FOLDER_CONTANTS(FOLDER_PATH)
```

In [11]:
```
def CURRENT_TIME():
    import time
    t = time.localtime()
    current_time = time.strftime("%H:%M:%S", t)
    print(f"Current time (HH:MM:SS): {current_time}\n\n")

# ---------------------
def plot_shapes(initial_coords, displacements):
    import numpy as np
    import matplotlib.pyplot as plt
    """
    Plot the initial and deformed shapes of the arch.

    Parameters:
    - initial_coords: Initial coordinates of the nodes.
    - displacements: List of displacements for each node.
    """
    # Plot initial and deformed shape
    plt.figure(figsize=(12, 8))
    plt.plot(initial_coords[:num_nodes, 0], initial_coords[:num_nodes, 1], 'bo-', label='Initial Cable Shape')
    plt.plot(deformed_coords[:num_nodes, 0], deformed_coords[:num_nodes, 1], 'ro-', label='Deformed Cable Shape')
    plt.plot(initial_coords[num_nodes:, 0], initial_coords[num_nodes:, 1], 'go-', label='Initial Deck Shape')
    plt.plot(deformed_coords[num_nodes:, 0], deformed_coords[num_nodes:, 1], 'yo-', label='Deformed Deck Shape')

    # Plot connecting elements between the cable and the beam
    for i in range(num_nodes):
        plt.plot([initial_coords[i, 0], initial_coords[num_nodes + i, 0]],
                 [initial_coords[i, 1], initial_coords[num_nodes + i, 1]], 'c--', label='Initial Connecting Elements' if i == 0 else "")
        plt.plot([deformed_coords[i, 0], deformed_coords[num_nodes + i, 0]],
                 [deformed_coords[i, 1], deformed_coords[num_nodes + i, 1]], 'm--', label='Deformed Connecting Elements' if i == 0 else "")

    # Mark simply supported nodes with red triangles
    support_nodes = [0, num_nodes - 1, num_nodes, 2*num_nodes - 1]
    plt.plot(initial_coords[support_nodes, 0], initial_coords[support_nodes, 1], 'r^', markersize=20, label='Simply Supported Nodes')

    plt.xlabel('X [mm]')
    plt.ylabel('Y [mm]')
    plt.legend()
    plt.title('Pushover Analysis: Cable and Deck Deformed Shapes')
    plt.show()

# ---------------------

def plot_reactions(disp_x, reaction_x, disp_y, reaction_y):
    import matplotlib.pyplot as plt
    """
    Plot the base reaction versus displacement.

    Parameters:
    - disp_x: List of x displacements of the middle node.
    - reaction_x: List of base reactions in x.
    - disp_y: List of y displacements of the middle node.
    - reaction_y: List of base reactions in y.
    """
    plt.figure(figsize=(10, 6))
    plt.plot(disp_x, reaction_x, 'b-')
    plt.xlabel('Displacement X [mm]')
    plt.ylabel('Base Reaction X [N]')
    plt.title('Base Reaction X vs. Displacement X')
    plt.show()

    plt.figure(figsize=(10, 6))
    plt.plot(disp_y, reaction_y, 'r-')
    plt.xlabel('Displacement Y [mm]')
    plt.ylabel('Base Reaction Y [N]')
```

```
    plt.title('Base Reaction Y vs. Displacement Y')
    plt.show()

# --------------------
"""
When OK equals -1, it generally indicates that the command or operation was not executed
because it was already in progress or had already been completed. This can happen if you
try to run a command that is already running or has been completed in a previous step.

When OK equals -2, it typically indicates that the command or operation was not executed
because it was not recognized or not implemented. This could mean that the command
is either misspelled, not available in the current version of OpenSees, or not applicable to the current context.

When OK equals -3, it typically means that the command or operation failed.
This could be due to various reasons, such as incorrect input parameters,
syntax errors, or issues with the model setup.
"""
def ANALYSIS(OK, INCREMENT, TOLERANCE, MAX_ITERAIONS):
    import openseespy.opensees as op
    test = {1:'NormDispIncr', 2: 'RelativeEnergyIncr', 4: 'RelativeNormUnbalance',5: 'RelativeNormDispIncr', 6: 'NormUnbalance'}
    algorithm = {1:'KrylovNewton', 2: 'SecantNewton' , 4: 'RaphsonNewton',5: 'PeriodicNewton', 6: 'BFGS', 7: 'Broyden', 8: 'NewtonLineSearch'}

    for i in test:
        for j in algorithm:
            if OK != 0:
                if j < 4:
                    op.algorithm(algorithm[j], '-initial')

                else:
                    op.algorithm(algorithm[j])

                op.test(test[i], TOLERANCE, MAX_ITERAIONS)
                OK = op.analyze(INCREMENT)
                print(test[i], algorithm[j], OK)
                if OK == 0:
                    break
            else:
                continue
```

In [ ]:

```
#          ##########################################################################################
#          #                                        IN THE NAME OF ALLAH                            #
#          #                          PUSHOVER ANALYSIS OF CABLE SUSPENSION BRIDGE 02               #
#          #-----------------------------------------------------------------------------------------#
#          #                  THIS PROGRAM WRITTEN BY SALAR DELAVAR GHASHGHAEI (QASHQAI)             #
#          #                             EMAIL: salar.d.ghashghaei@gmail.com                         #
#          ##########################################################################################
```

In [12]:

```
# --------------------
#   PUSHOVER ANALYSIS
# --------------------
def PUSHOVER_ANALYSIS( LINEAR, L, H1, H2, arc_depth, E_cable, Cable_Dia_01, Cable_Dia_02, Cable_Dia_03, num_nodes, MAX_DISP, disp_incr, MAX_ITERATIONS, TOLERANCE):
    import openseespy.opensees as ops
    import numpy as np

    A_cable_01 = (np.pi * Cable_Dia_01 **2) / 4  # [mm^2] Horizontal Longitudinal Cable Area Bottom
    A_cable_02 = (np.pi * Cable_Dia_02 **2) / 4  # [mm^2] Horizontal Longitudinal Cable Area Top
    A_cable_03 = (np.pi * Cable_Dia_03 **2) / 4  # [mm^2] Vertical Longitudinal Cable Area Top

    # Define model builder
    ops.wipe()
    ops.model('basic', '-ndm', 2, '-ndf', 2)
    dx = L / (num_nodes - 1)
    # Create nodes
    for i in range(num_nodes):
        X = i * dx
        Y1 = H1 - arc_depth * np.sin(np.pi * X / L)
        Y2 = Y1 - H2
        ops.node(i + 1, X, Y1)
        ops.node(num_nodes + i + 1, i * dx, Y2)

    # Define boundary conditions (fixed at both ends)
    ops.fix(1, 1, 1)              # TOP CABLE
    ops.fix(num_nodes, 1, 1)      # TOP CABLE
    ops.fix(num_nodes + 1, 1, 1)  # BOTTOM CABLE
    ops.fix(2 * num_nodes, 1, 1)  # BOTTOM CABLE

    # Define material properties
    if LINEAR == True:
        ops.uniaxialMaterial('Elastic', 1, E_cable)
        #                                  TENSION    COMPRESSION
        #ops.uniaxialMaterial('Elastic', 1, E_cable, 0, 0.5 * E_cable)
    if LINEAR == False:
        Fy_cable = 355   # [N/mm^2] Yield strength of the cable
        b0 = 0.01
        ops.uniaxialMaterial('Steel01', 1, Fy_cable, E_cable, b0)

    # Define truss elements
    for i in range(num_nodes - 1):
        ops.element('corotTruss', i + 1, i + 1, i + 2, A_cable_02, 1)  # Top Cable element
        ops.element('corotTruss', num_nodes + i + 1, num_nodes + i + 1, num_nodes + i + 2, A_cable_01, 1)  # Bottom Cable Element

    # Connect each node of the cable with the corresponding node of the deck using truss elements
    for i in range(num_nodes):
        ops.element('corotTruss', 2 * num_nodes + i + 1, i + 1, num_nodes + i + 1, A_cable_03, 1) # Vertical Cable Element

    #mid_node = num_nodes // 2 + 1
    mid_node = int(num_nodes + 0.5 * num_nodes)
    # Define load pattern with displacement at the middle span
    ops.timeSeries('Linear', 1)
    ops.pattern('Plain', 1, 1)
    ops.load(mid_node, 0.0, -1)  # Apply a horizontal load at node

    n_steps = int(np.abs(MAX_DISP / disp_incr))  # Analysis Steps

    # Define analysis parameters
    ops.system('BandGeneral')
    ops.numberer('Plain')
    ops.constraints('Plain')
    ops.integrator('DisplacementControl', mid_node, 2, disp_incr)
    ops.test('EnergyIncr', TOLERANCE, MAX_ITERATIONS)
    ops.algorithm('ModifiedNewton')
    ops.analysis('Static')
```

```
    # OUTPUT DATA
    ops.recorder('Node', '-file', f"{SALAR_DIR}DTH_PUSH.txt",'-time', '-node', mid_node, '-dof', 1,2, 'disp')# Displacement Time History
    ops.recorder('Node', '-file', f"{SALAR_DIR}BTH_PUSH_01.txt",'-time', '-node', 1, '-dof', 1,2, 'reaction')# Base Shear Time History
    ops.recorder('Node', '-file', f"{SALAR_DIR}BTH_PUSH_02.txt",'-time', '-node', num_nodes, '-dof', 1,2, 'reaction')# Base Shear Time History
    ops.recorder('Node', '-file', f"{SALAR_DIR}BTH_PUSH_03.txt",'-time', '-node', num_nodes + 1, '-dof', 1,2, 'reaction')# Base Shear Time History
    ops.recorder('Node', '-file', f"{SALAR_DIR}BTH_PUSH_04.txt",'-time', '-node', 2*num_nodes, '-dof', 1,2, 'reaction')# Base Shear

    DISPLACEMENTS = []
    DISP_X , DISP_Y = [], []
    BASE_REACTION_X, BASE_REACTION_Y = [], []

    # Perform the analysis with increments
    for i in range(n_steps):
        OK = ops.analyze(1)
        ANALYSIS(OK, 1, TOLERANCE, MAX_ITERATIONS)
        DISPLACEMENTS.append([ops.nodeDisp(j + 1) for j in range(num_nodes * 2)])
        DISP_X.append(ops.nodeDisp(mid_node, 1))
        DISP_Y.append(ops.nodeDisp(mid_node, 2))
        x1 = ops.nodeResponse(1, 1, 6)
        x2 = ops.nodeResponse(num_nodes, 1, 6)
        x3 = ops.nodeResponse(num_nodes + 1, 1, 6)
        x4 = ops.nodeResponse(2*num_nodes, 1, 6)
        y1 = ops.nodeResponse(1, 2, 6)
        y2 = ops.nodeResponse(num_nodes, 2, 6)
        y3 = ops.nodeResponse(num_nodes + 1, 2, 6)
        y4 = ops.nodeResponse(2*num_nodes, 2, 6)
        BASE_REACTION_X.append(x1+x2+x3+x4) # CABLE REACION-X
        BASE_REACTION_Y.append(y1+y2+y3+y4) # CABLE REACION-Y
        #print(f'STEP: {i+1}')

    # Get initial and final node coordinates for plotting
    initial_coords = np.array([ops.nodeCoord(i + 1) for i in range(num_nodes * 2)])
    deformed_coords = initial_coords + np.array(DISPLACEMENTS[-1])

    # Output all unconstrained node displacements
    for i in range(num_nodes * 2):
        disp = ops.nodeDisp(i + 1)
        print(f"Node {i + 1}: X Displacement = {disp[0]}, Y Displacement = {disp[1]}")

    return initial_coords, deformed_coords, DISPLACEMENTS, DISP_X, DISP_Y, BASE_REACTION_X, BASE_REACTION_Y
```

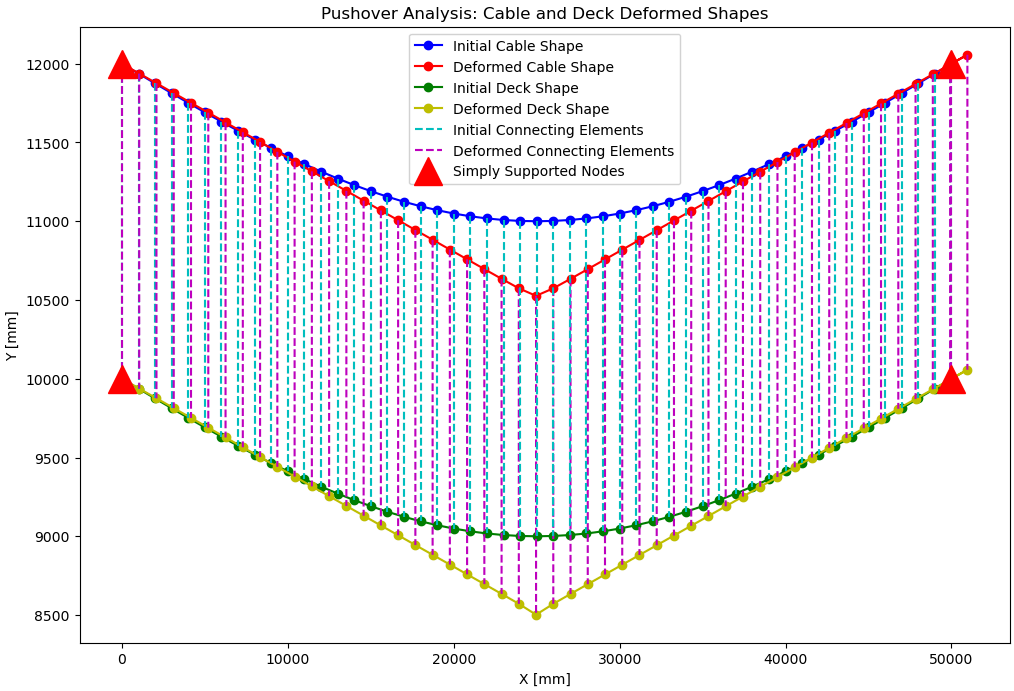
In [13]:

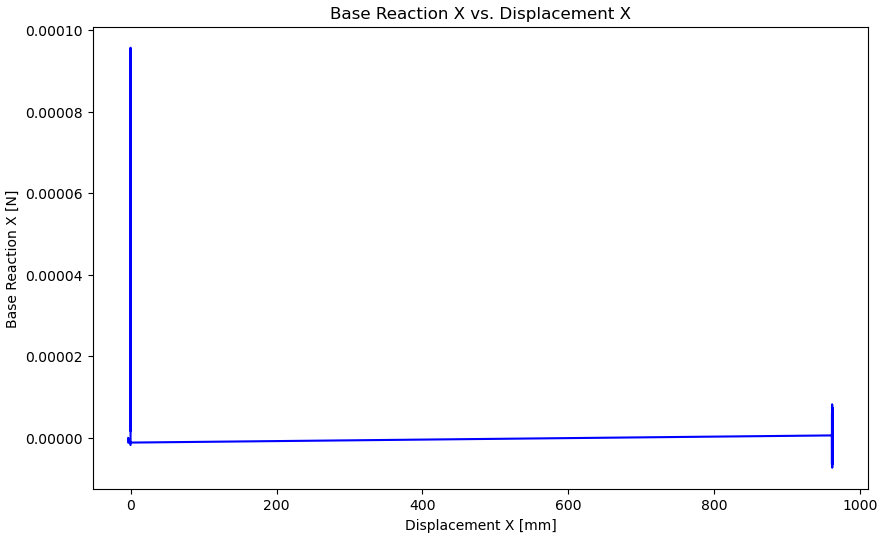
```
# --------------------
#   PUSHOVER ANALYSIS
# --------------------
# Parameters for the analysis
L = 50000.0   # [mm] Bridge span lentgh
H1 = 12000.0  # [mm] Height of Top Cable
H2 = 2000     # [mm] Height of Bottom Cable
arc_depth = 1000.0 # [mm]
E_cable = 210e5    # [N/mm^2] Modulus of elasticity Cable
Cable_Dia_01 = 25  # [mm] Horizontal Longitudinal Cable Diameter Bottom
Cable_Dia_02 = 18  # [mm] Horizontal Longitudinal Cable Diameter Top
Cable_Dia_03 = 10  # [mm] Vertical Longitudinal Cable Diameter Top
num_nodes = 51 # Cable Arc Number of nodes
MAX_DISP = -500.0   # [mm] Maximum Displacment
disp_incr = -0.01 # [mm] Incremental Displacement

MAX_ITERATIONS = 50000  # Maximum number of iterations
TOLERANCE = 1.0e-10     # Tolerance for convergence

LINEAR = True # False: Cable Nonlinear Materials Properties

starttime = time.process_time()
# Run the analysis
results = PUSHOVER_ANALYSIS(LINEAR, L, H1, H2, arc_depth, E_cable, Cable_Dia_01, Cable_Dia_02, Cable_Dia_03, num_nodes, MAX_DISP, disp_incr, MAX_ITERATIONS, TOLERANCE)
initial_coords, deformed_coords, DISPLACEMENTS, DISP_X, DISP_Y, BASE_REACTION_X, BASE_REACTION_Y = results
totaltime = time.process_time() - starttime
print(f'\nTotal time (s): {totaltime:.4f} \n\n')
```

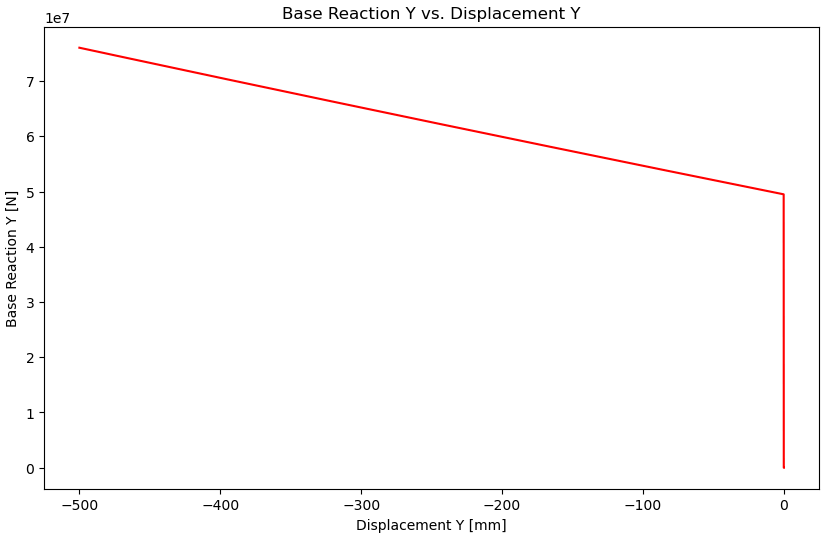
```
WARNING: CTestEnergyIncr::test() - failed to converge
after: 50000 iterations
 current EnergyIncr: -nan(ind) (max: 1e-10)     Norm deltaX: nan, Norm deltaR: nan
ModifiedNewton::solveCurrentStep() -the ConvergenceTest object failed in test()
StaticAnalysis::analyze() - the Algorithm failed at step: 0 with domain at load factor -nan(ind)
OpenSees > analyze failed, returned: -3 error flag
```

```
NormDispIncr KrylovNewton 0
Node 1: X Displacement = 0.0, Y Displacement = 0.0
Node 2: X Displacement = 40.9909964865675, Y Displacement = 0.6597248531402142
Node 3: X Displacement = 81.96589060086747, Y Displacement = 1.0726044562576544
Node 4: X Displacement = 122.90883321705017, Y Displacement = 0.9927571299360808
Node 5: X Displacement = 163.80447767906463, Y Displacement = 0.17622338640625676
Node 6: X Displacement = 204.63822122023302, Y Displacement = -1.6180821470261337
Node 7: X Displacement = 245.39643478238295, Y Displacement = -4.627429561193242
Node 8: X Displacement = 286.0666776364837, Y Displacement = -9.084346868911167
Node 9: X Displacement = 326.6378934396328, Y Displacement = -15.21571109446846
Node 10: X Displacement = 367.10058464532625, Y Displacement = -23.24186069894603
Node 11: X Displacement = 407.4469625137423, Y Displacement = -33.37573272112985
Node 12: X Displacement = 447.6710703421024, Y Displacement = -45.82202793258861
Node 13: X Displacement = 487.7688779469055, Y Displacement = -60.77640721312702
Node 14: X Displacement = 527.7383458741754, Y Displacement = -78.42472224853591
Node 15: X Displacement = 567.5794582846598, Y Displacement = -98.94228353185733
Node 16: X Displacement = 607.2942239533095, Y Displacement = -122.49316847535565
Node 17: X Displacement = 646.8866453455103, Y Displacement = -149.22957192361145
Node 18: X Displacement = 686.3626564086311, Y Displacement = -179.291198231254
Node 19: X Displacement = 725.7300315559506, Y Displacement = -212.80467057963173
Node 20: X Displacement = 764.9982792781656, Y Displacement = -249.882750383064
Node 21: X Displacement = 804.1786162338893, Y Displacement = -290.62172960866
Node 22: X Displacement = 843.2847592599405, Y Displacement = -335.0841661366055
Node 23: X Displacement = 882.3402730770282, Y Displacement = -383.1665198088227
Node 24: X Displacement = 921.4362107551046, Y Displacement = -433.56526595886896
Node 25: X Displacement = 961.1104106961169, Y Displacement = -475.67354307332886
Node 26: X Displacement = 1000.7770830607161, Y Displacement = -425.87308180728905
Node 27: X Displacement = 1039.8356774674128, Y Displacement = -367.8151497906017
Node 28: X Displacement = 1078.8228790050114, Y Displacement = -312.13761288123936
Node 29: X Displacement = 1117.830750749167, Y Displacement = -260.17465836188273
Node 30: X Displacement = 1156.884555964733, Y Displacement = -212.06010968028951
Node 31: X Displacement = 1196.000183968479, Y Displacement = -167.7611809567745
Node 32: X Displacement = 1235.1914392590252, Y Displacement = -127.21072282259884
Node 33: X Displacement = 1274.470789306477, Y Displacement = -90.32437393793745
Node 34: X Displacement = 1313.8492912236136, Y Displacement = -57.0030587765934 56
Node 35: X Displacement = 1353.3364332062051, Y Displacement = -27.133661846651194
Node 36: X Displacement = 1392.9399871454132, Y Displacement = -0.5895159529858277
Node 37: X Displacement = 1432.6658867193794, Y Displacement = 22.769087966816677
Node 38: X Displacement = 1472.5181343938514, Y Displacement = 43.094344758531726
Node 39: X Displacement = 1512.498738959863, Y Displacement = 60.550331557607294
Node 40: X Displacement = 1552.6076845772996, Y Displacement = 75.31235887210534
Node 41: X Displacement = 1592.8429317702564, Y Displacement = 87.56627876652823
Node 42: X Displacement = 1633.2004503120656, Y Displacement = 97.50775286617234
Node 43: X Displacement = 1673.6742834361744, Y Displacement = 105.34148304397132
Node 44: X Displacement = 1714.2566423194764, Y Displacement = 111.2804077769925
Node 45: X Displacement = 1754.938029314723, Y Displacement = 115.54486726594445
Node 46: X Displacement = 1795.7073879681477, Y Displacement = 118.3617404480164
Node 47: X Displacement = 1836.5522774765556, Y Displacement = 119.96355660341803
Node 48: X Displacement = 1877.4590690988275, Y Displacement = 120.58758024266699
Node 49: X Displacement = 1918.4131635417193, Y Displacement = 120.47483593167185
Node 50: X Displacement = 1959.3992424420678, Y Displacement = 119.86878811311152
Node 51: X Displacement = 0.0, Y Displacement = 0.0
Node 52: X Displacement = 0.0, Y Displacement = 0.0
Node 53: X Displacement = 41.04412905696479, Y Displacement = 0.6597257652829351
Node 54: X Displacement = 82.07215499039756, Y Displacement = 1.0726075119836262
Node 55: X Displacement = 123.0682278745019, Y Displacement = 0.992763717646462
Node 56: X Displacement = 164.01700027106375, Y Displacement = 0.17623491433281793
Node 57: X Displacement = 204.90386866820404, Y Displacement = -1.6180642683803508
Node 58: X Displacement = 245.7152033121163, Y Displacement = -4.627403921355337
Node 59: X Displacement = 286.4385628387677, Y Displacement = -9.084312057785011
Node 60: X Displacement = 327.06289034090804, Y Displacement = -15.215665702423417
Node 61: X Displacement = 367.57868778724475, Y Displacement = -23.24180331686353
Node 62: X Displacement = 407.9781660403836, Y Displacement = -33.37566194043596
Node 63: X Displacement = 448.25536809347847, Y Displacement = -45.821942345303036
Node 64: X Displacement = 488.40626355727545, Y Displacement = -60.77630541212892
Node 65: X Displacement = 528.428812873535, Y Displacement = -78.42460282962884
Node 66: X Displacement = 568.3230002010965, Y Displacement = -98.94214511001226
Node 67: X Displacement = 608.0908344090367, Y Displacement = -122.4930098120958
Node 68: X Displacement = 647.7363181014331, Y Displacement = -149.22939292478904
Node 69: X Displacement = 687.2653850150929, Y Displacement = -179.29100775999677
Node 70: X Displacement = 726.6858059404259, Y Displacement = -212.80454762735374
Node 71: X Displacement = 766.0070583844997, Y Displacement = -249.88332296743658
Node 72: X Displacement = 805.2401132348741, Y Displacement = -290.6279237539563
Node 73: X Displacement = 844.3967599262307, Y Displacement = -335.1345527159811
Node 74: X Displacement = 883.4854868291677, Y Displacement = -383.56303338475414
Node 75: X Displacement = 922.4802374865483, Y Displacement = -436.67148156858576
Node 76: X Displacement = 961.0576874774634, Y Displacement = -499.9999999996882
Node 77: X Displacement = 999.6312345421977, Y Displacement = -428.97921378107645
Node 78: X Displacement = 1038.5889365217286, Y Displacement = -368.21160460168517
Node 79: X Displacement = 1077.6092169221095, Y Displacement = -312.18794248652495
Node 80: X Displacement = 1116.6674021163535, Y Displacement = -260.1807965164312
Node 81: X Displacement = 1155.7737266711692, Y Displacement = -212.06062834084796
Node 82: X Displacement = 1194.942158634012, Y Displacement = -167.76100655793584
Node 83: X Displacement = 1234.1862576193173, Y Displacement = -127.21048349071941
Node 84: X Displacement = 1273.518460356762, Y Displacement = -90.3241486951409
Node 85: X Displacement = 1312.9498205134578, Y Displacement = -57.00285650041504
Node 86: X Displacement = 1352.4898262557049, Y Displacement = -27.13348245441269
Node 87: X Displacement = 1392.1462498044873, Y Displacement = -0.5893582172825105
Node 88: X Displacement = 1431.9250251313272, Y Displacement = 22.769225419909304
Node 89: X Displacement = 1471.8301549044218, Y Displacement = 43.09446332202176
Node 90: X Displacement = 1511.8636480167968, Y Displacement = 60.55043262735694
Node 91: X Displacement = 1552.0254886274836, Y Displacement = 75.3124438447938
Node 92: X Displacement = 1592.313637156827, Y Displacement = 87.56634903941817
Node 93: X Displacement = 1632.724063173168, Y Displacement = 97.50780983713665
Node 94: X Displacement = 1673.2508096069687, Y Displacement = 105.34152811184967
Node 95: X Displacement = 1713.8860872388102, Y Displacement = 111.28044234441685
Node 96: X Displacement = 1754.6203979369573, Y Displacement = 115.54489276157823
Node 97: X Displacement = 1795.4426846739118, Y Displacement = 118.36175850105776
Node 98: X Displacement = 1836.3405059241493, Y Displacement = 119.9635704118153
Node 99: X Displacement = 1877.3002315507447, Y Displacement = 120.58760529948005
Node 100: X Displacement = 1918.30725600898, Y Displacement = 120.47498411644925
Node 101: X Displacement = 1959.3462169665627, Y Displacement = 119.86992695138665
Node 102: X Displacement = 0.0, Y Displacement = 0.0

Total time (s): 83.7969
```

In [14]:
```
# Plot results
plot_shapes(initial_coords, DISPLACEMENTS)
plot_reactions(DISP_X, BASE_REACTION_X, DISP_Y, BASE_REACTION_Y)
```







In [9]:

```
#          ###########################################################################################
#          #                                    IN THE NAME OF ALLAH                                 #
#          #                     FREE VIBRATION ANALYSIS OF CABLE SUSPENSION BRIDGE 02               #
#          #-----------------------------------------------------------------------------------------#
#          #                THIS PROGRAM WRITTEN BY SALAR DELAVAR GHASHGHAEI (QASHQAI)               #
```

```
#         #                          EMAIL: salar.d.ghashghaei@gmail.com                          #
#         ########################################################################################
```

In [16]:

```
# -------------------------
#   FREE VIBRATION ANALYSIS
# -------------------------
def FREE_VIBRATION_ANALYSIS(damping, damping_ratio, LINEAR, duration, dt, TOTAL_MASS, u0, L, H1, H2, arc_depth, E_cable, Cable_Dia_01, Cable_Dia_02, Cable_Dia_03, num_nodes, MAX_ITERAT
    import openseespy.opensees as ops
    import numpy as np
    import matplotlib.pyplot as plt

    A_cable_01 = (np.pi * Cable_Dia_01 **2) / 4  # [mm^2] Horizontal Longitudinal Cable Area Bottom
    A_cable_02 = (np.pi * Cable_Dia_02 **2) / 4  # [mm^2] Horizontal Longitudinal Cable Area Top
    A_cable_03 = (np.pi * Cable_Dia_03 **2) / 4  # [mm^2] Vertical Longitudinal Cable Area Top

    # Define model builder
    ops.wipe()
    ops.model('basic', '-ndm', 2, '-ndf', 2)
    KE = 1;
    dx = L / (num_nodes - 1)
    MASS = TOTAL_MASS / num_nodes
    # Create nodes
    for i in range(num_nodes):
        X = i * dx
        Y1 = H1 - arc_depth * np.sin(np.pi * X / L)
        Y2 = Y1 - H2
        ops.node(i + 1, X, Y1)
        ops.node(num_nodes + i + 1, i * dx, Y2)
        # Define mass
        ops.mass(num_nodes + i + 1, MASS, MASS, 0)

    # Define boundary conditions (fixed at both ends)
    ops.fix(1, 1, 1)              # TOP CABLE
    ops.fix(num_nodes, 1, 1)      # TOP CABLE
    ops.fix(num_nodes + 1, 1, 1)  # BOTTOM CABLE
    ops.fix(2 * num_nodes, 1, 1)  # BOTTOM CABLE

    # Define material properties
    if LINEAR == True:
        ops.uniaxialMaterial('Elastic', 1, E_cable)
        #                                 TENSION    COMPRESSION
        #ops.uniaxialMaterial('Elastic', 1, E_cable, 0, 0.5 * E_cable)
    if LINEAR == False:
        Fy_cable = 355   # [N/mm^2] Yield strength of the cable
        b0 = 0.01
        ops.uniaxialMaterial('Steel01', 1, Fy_cable, E_cable, b0)

    # Define truss elements
    for i in range(num_nodes - 1):
        ops.element('corotTruss', i + 1, i + 1, i + 2, A_cable_02, 1)  # Top Cable element
        ops.element('corotTruss', num_nodes + i + 1, num_nodes + i + 1, num_nodes + i + 2, A_cable_01, 1)  # Bottom Cable Element

    # Connect each node of the cable with the corresponding node of the deck using truss elements
    for i in range(num_nodes):
        ops.element('corotTruss', 2 * num_nodes + i + 1, i + 1, num_nodes + i + 1, A_cable_03, 1) # Vertical Cable Element

    #mid_node = num_nodes // 2 + 1
    mid_node = int(num_nodes + 0.5 * num_nodes)
    # Define load pattern with displacement at the middle span
    ops.timeSeries('Linear', 1)
    ops.pattern('Plain', 1, 1)
    ops.load(mid_node, 0.0, -1)  # Apply a horizontal load at node
    ops.constraints('Transformation')
    ops.numberer('RCM')
    ops.system('BandGeneral')
    ops.algorithm('Newton')
    ops.test('EnergyIncr', TOLERANCE, MAX_ITERATIONS)
    ops.integrator('DisplacementControl', mid_node, 2, u0)
    ops.analysis('Static')
    ops.analyze(1)

    ops.setTime(0.0)

    # Wipe analysis and reset time
    ops.wipeAnalysis()
    ops.remove('loadPattern', 1)
    ops.system('UmfPack')

    # Dynamic analysis
    ops.constraints('Transformation')
    ops.numberer('RCM')
    ops.system('UmfPack')
    ops.test('EnergyIncr', TOLERANCE, MAX_ITERATIONS)
    ops.integrator('Newmark', 0.5, 0.25)
    ops.algorithm('Newton')

    if damping == True:
        # Calculate Rayleigh damping factors
        omega1 = np.sqrt(KE / TOTAL_MASS)
        omega2 = 2 * omega1  # Just an assumption for two modes
        a0 = damping_ratio * (2 * omega1 * omega2) / (omega1 + omega2)
        a1 = damping_ratio * 2 / (omega1 + omega2)
        # Apply Rayleigh damping
        ops.rayleigh(a0, a1, 0, 0)

    ops.analysis('Transient')

    # OUTPUT DATA
    ops.recorder('Node', '-file', f"{SALAR_DIR}DTH_DYN.txt",'-time', '-node', mid_node, '-dof', 1,2, 'disp')# Displacement Time History
    ops.recorder('Node', '-file', f"{SALAR_DIR}VTH_DYN.txt",'-time', '-node', mid_node, '-dof', 1,2,3, 'vel')  # Velocity Time History
    ops.recorder('Node', '-file', f"{SALAR_DIR}ATH_DYN.txt",'-time', '-node', mid_node, '-dof', 1,2,3, 'accel') # Acceleration Time History
    ops.recorder('Node', '-file', f"{SALAR_DIR}BTH_DYN_01.txt",'-time', '-node', 1, '-dof', 1,2, 'reaction')# Base Shear Time History
    ops.recorder('Node', '-file', f"{SALAR_DIR}BTH_DYN_02.txt",'-time', '-node', num_nodes, '-dof', 1,2, 'reaction')# Base Shear Time History
    ops.recorder('Node', '-file', f"{SALAR_DIR}BTH_DYN_03.txt",'-time', '-node', num_nodes + 1, '-dof', 1,2, 'reaction')# Base Shear Time History
    ops.recorder('Node', '-file', f"{SALAR_DIR}BTH_DYN_04.txt",'-time', '-node', 2*num_nodes, '-dof', 1,2, 'reaction')# Base Shear Time History

    DISPLACEMENTS = []
    DISP_X , DISP_Y, VELOCITY, ACCELERATION = [], [], [], []
    BASE_REACTION_X, BASE_REACTION_Y = [], []

    stable = 0
    current_time = 0.0
```

```
    # Perform the analysis with increments
    while stable == 0 and current_time < duration:
        OK = ops.analyze(1, dt)
        ANALYSIS(OK, 1, TOLERANCE, MAX_ITERATIONS)
        current_time = ops.getTime()
        DISPLACEMENTS.append([ops.nodeDisp(j + 1) for j in range(num_nodes * 2)])
        DISP_X.append(ops.nodeDisp(mid_node, 1))
        DISP_Y.append(ops.nodeDisp(mid_node, 2))
        VELOCITY.append(ops.nodeVel(mid_node, 2))
        ACCELERATION.append(ops.nodeAccel(mid_node, 2))
        x1 = ops.nodeResponse(1, 1, 6)
        x2 = ops.nodeResponse(num_nodes, 1, 6)
        x3 = ops.nodeResponse(num_nodes + 1, 1, 6)
        x4 = ops.nodeResponse(2*num_nodes, 1, 6)
        y1 = ops.nodeResponse(1, 2, 6)
        y2 = ops.nodeResponse(num_nodes, 2, 6)
        y3 = ops.nodeResponse(num_nodes + 1, 2, 6)
        y4 = ops.nodeResponse(2*num_nodes, 2, 6)
        BASE_REACTION_X.append(x1+x2+x3+x4) # CABLE REACION-X
        BASE_REACTION_Y.append(y1+y2+y3+y4) # CABLE REACION-Y
        KE =  BASE_REACTION_Y[-1] / DISP_Y[-1] # Effective Lateral Stiffness
        #print(f'STEP: {i+1}')

    # Get initial and final node coordinates for plotting
    initial_coords = np.array([ops.nodeCoord(i + 1) for i in range(num_nodes * 2)])
    deformed_coords = initial_coords + np.array(DISPLACEMENTS[-1])

    # Output all unconstrained node displacements
    for i in range(num_nodes * 2):
        disp = ops.nodeDisp(i + 1)
        print(f"Node {i + 1}: X Displacement = {disp[0]}, Y Displacement = {disp[1]}")

    return initial_coords, deformed_coords, DISPLACEMENTS, VELOCITY, ACCELERATION, DISP_X, DISP_Y, BASE_REACTION_X, BASE_REACTION_Y
```

In [17]:
```
# Define the plotting function
def plot_time_history(TIME, DISP_X_undamped, DISP_X_damped, DISP_Y_undamped, DISP_Y_damped,
                      VELOCITY_undamped, VELOCITY_damped, ACCELERATION_undamped, ACCELERATION_damped,
                      BASE_REACTION_X_undamped, BASE_REACTION_X_damped, BASE_REACTION_Y_undamped, BASE_REACTION_Y_damped):
    import matplotlib.pyplot as plt
    import numpy as np

    plt.figure(figsize=(14, 26))

    # Displacement X
    plt.subplot(6, 1, 1)
    P1 = np.max(np.abs(np.array(DISP_X_undamped)))
    P2 = np.max(np.abs(np.array(DISP_X_damped)))
    plt.plot(TIME, DISP_X_undamped, color='black', label=f'Undamped: {P1:.4e}')
    plt.plot(TIME, DISP_X_damped, color='red', label=f'Damped: {P2:.4e}')
    plt.xlabel('Time [s]')
    plt.ylabel('Displacement X [mm]')
    plt.legend()

    # Displacement Y
    plt.subplot(6, 1, 2)
    P1 = np.max(np.abs(np.array(DISP_Y_undamped)))
    P2 = np.max(np.abs(np.array(DISP_Y_damped)))
    plt.plot(TIME, DISP_Y_undamped, color='black', label=f'Undamped: {P1:.4e}')
    plt.plot(TIME, DISP_Y_damped, color='red', label=f'Damped: {P2:.4e}')
    plt.xlabel('Time [s]')
    plt.ylabel('Displacement Y [mm]')
    plt.legend()

    # Velocity
    plt.subplot(6, 1, 3)
    P1 = np.max(np.abs(np.array(VELOCITY_damped)))
    P2 = np.max(np.abs(np.array(VELOCITY_undamped)))
    plt.plot(TIME, VELOCITY_undamped, color='black', label=f'Undamped: {P1:.4e}')
    plt.plot(TIME, VELOCITY_damped, color='red', label=f'Damped: {P2:.4e}')
    plt.xlabel('Time [s]')
    plt.ylabel('Velocity [mm/s]')
    plt.legend()

    # Acceleration
    plt.subplot(6, 1, 4)
    P1 = np.max(np.abs(np.array(ACCELERATION_undamped)))
    P2 = np.max(np.abs(np.array(ACCELERATION_damped)))
    plt.plot(TIME, ACCELERATION_undamped, color='black', label=f'Undamped: {P1:.4e}')
    plt.plot(TIME, ACCELERATION_damped, color='red', label=f'Damped: {P2:.4e}')
    plt.xlabel('Time [s]')
    plt.ylabel('Acceleration [mm/s^2]')
    plt.legend()

    # Base Reaction X
    plt.subplot(6, 1, 5)
    P1 = np.max(np.abs(np.array(BASE_REACTION_X_undamped)))
    P2 = np.max(np.abs(np.array(BASE_REACTION_X_damped)))
    plt.plot(TIME, BASE_REACTION_X_undamped, color='black', label=f'Undamped: {P1:.4e}')
    plt.plot(TIME, BASE_REACTION_X_damped, color='red', label=f'Damped: {P2:.4e}')
    plt.xlabel('Time [s]')
    plt.ylabel('Base Reaction X [N]')
    plt.legend()

    # Base Reaction Y
    plt.subplot(6, 1, 6)
    P1 = np.max(np.abs(np.array(BASE_REACTION_Y_undamped)))
    P2 = np.max(np.abs(np.array(BASE_REACTION_Y_damped)))
    plt.plot(TIME, BASE_REACTION_Y_undamped, color='black', label=f'Undamped: {P1:.4e}')
    plt.plot(TIME, BASE_REACTION_Y_damped, color='red', label=f'Damped: {P2:.4e}')
    plt.xlabel('Time [s]')
    plt.ylabel('Base Reaction Y [N]')
    plt.legend()

    plt.suptitle('Time History of Free Vibration Analysis: Damped vs Undamped')
    plt.tight_layout(rect=[0, 0.03, 1, 0.95])
    plt.show()
```

In [18]:
```
# -------------------------
#   FREE VIBRATION ANALYSIS
# -------------------------
# Parameters for the analysis
L = 50000.0   # [mm] Bridge span lentgh
```

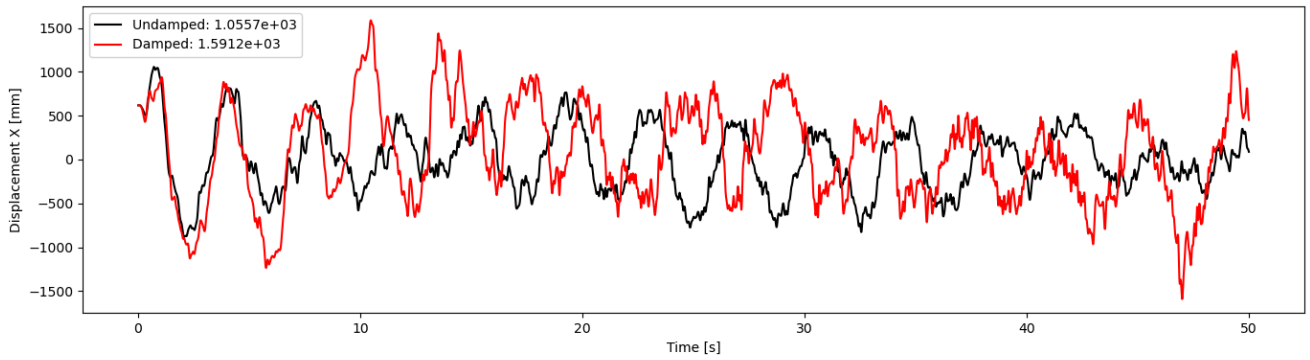
```python
H1 = 12000.0  # [mm] Height of Top Cable
H2 = 2000     # [mm] Height of Bottom Cable
arc_depth = 1000.0 # [mm]
E_cable = 210e5    # [N/mm^2] Modulus of elasticity Cable
Cable_Dia_01 = 25  # [mm] Horizontal Longitudinal Cable Diameter Bottom
Cable_Dia_02 = 18  # [mm] Horizontal Longitudinal Cable Diameter Top
Cable_Dia_03 = 10  # [mm] Vertical Longitudinal Cable Diameter Top
num_nodes = 51 # Cable Arc Number of nodes

TOTAL_MASS = 500000.0  # [kg] Total Mass of Structure
u0 = 2.0  # [mm] Initial displacement
damping_ratio = 0.05  # Damping ratio
duration = 50.0  # [s] Duration of the analysis in seconds
dt = 0.01  # Time step in seconds

MAX_ITERATIONS = 10000  # Maximum number of iterations
TOLERANCE = 1.0e-14     # Tolerance for convergence

import time
starttime = time.process_time()

# Run the undamped analysis
damping = False
LINEAR = True # False: Cable Nonlinear Materials Properties
results_undamped = FREE_VIBRATION_ANALYSIS(damping, damping_ratio, LINEAR, duration, dt, TOTAL_MASS, u0, L, H1, H2, arc_depth, E_cable, Cable_Dia_01, Cable_Dia_02, Cable_Dia_03, num_no
initial_coords, deformed_coords, DISPLACEMENTS_undamped, VELOCITY_undamped, ACCELERATION_undamped, DISP_X_undamped, DISP_Y_undamped, BASE_REACTION_X_undamped, BASE_REACTION_Y_undamped

# Run the damped analysis
damping = True
LINEAR = True # False: Cable Nonlinear Materials Properties
results_damped = FREE_VIBRATION_ANALYSIS(damping, damping_ratio,  LINEAR, duration, dt, TOTAL_MASS, u0, L, H1, H2, arc_depth, E_cable, Cable_Dia_01, Cable_Dia_02, Cable_Dia_03, num_noc
initial_coords, deformed_coords, DISPLACEMENTS_damped, VELOCITY_damped, ACCELERATION_damped, DISP_X_damped, DISP_Y_damped, BASE_REACTION_X_damped, BASE_REACTION_Y_damped = results_damp

totaltime = time.process_time() - starttime
print(f'\nTotal time (s): {totaltime:.4f} \n\n')
```

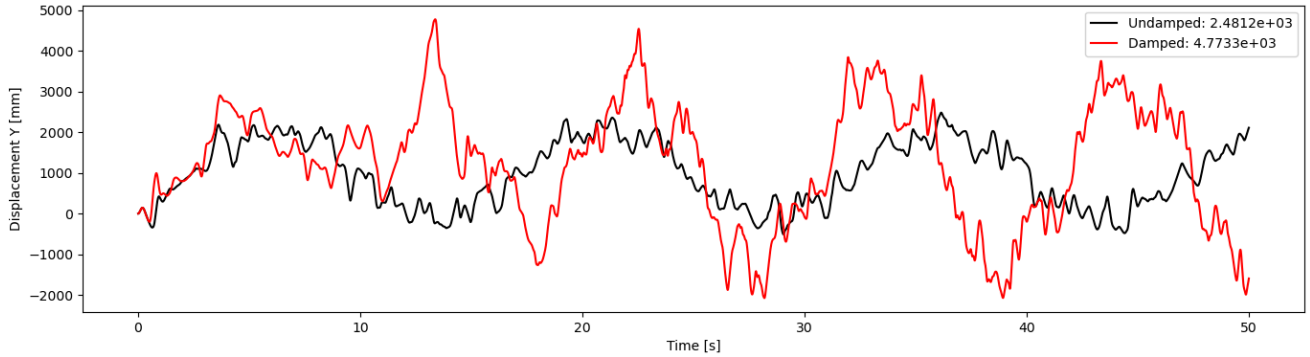
Node 1: X Displacement = 0.0, Y Displacement = 0.0
Node 2: X Displacement = -30.718668519346693, Y Displacement = -266.15342691477963
Node 3: X Displacement = -20.586939650564606, Y Displacement = -362.7793805073048
Node 4: X Displacement = 1.6191300734020357, Y Displacement = -290.89475908844895
Node 5: X Displacement = -109.92970300030427, Y Displacement = 280.5076592375267
Node 6: X Displacement = -89.64066716735655, Y Displacement = 377.18029364874707
Node 7: X Displacement = -91.01187381974067, Y Displacement = 650.9705333436825
Node 8: X Displacement = -118.90527498512695, Y Displacement = 395.8699507217365
Node 9: X Displacement = -103.09535238512199, Y Displacement = 353.85176683903217
Node 10: X Displacement = -84.57948723170868, Y Displacement = 470.8833250254912
Node 11: X Displacement = -122.97982777416968, Y Displacement = 867.4623019382296
Node 12: X Displacement = -185.80988732747971, Y Displacement = 1324.5220803802215
Node 13: X Displacement = -172.72701627058728, Y Displacement = 1491.0296640115528
Node 14: X Displacement = -167.1623945990054, Y Displacement = 1707.1930002946845
Node 15: X Displacement = -150.4831765379583, Y Displacement = 1668.4246031041064
Node 16: X Displacement = -145.0633483593968, Y Displacement = 1877.3225201785483
Node 17: X Displacement = -131.37896781417155, Y Displacement = 2021.854574400206
Node 18: X Displacement = -142.91040345969785, Y Displacement = 1799.6868707758529
Node 19: X Displacement = -134.92356369205248, Y Displacement = 1672.3754290535942
Node 20: X Displacement = -136.52208178292722, Y Displacement = 1487.8489336231883
Node 21: X Displacement = -148.20715520278804, Y Displacement = 1759.1219626994773
Node 22: X Displacement = -136.04648923383792, Y Displacement = 1895.8813389132176
Node 23: X Displacement = -121.9360489984197, Y Displacement = 2010.1095723162234
Node 24: X Displacement = -124.97090818488262, Y Displacement = 1805.0766440608857
Node 25: X Displacement = -134.74852284478783, Y Displacement = 2051.882567989894
Node 26: X Displacement = -122.93940719977329, Y Displacement = 1929.0043294030888
Node 27: X Displacement = -118.94172691360146, Y Displacement = 1748.2120953539206
Node 28: X Displacement = -112.52316477872172, Y Displacement = 1578.153304923999
Node 29: X Displacement = -123.16166878985467, Y Displacement = 1811.3356297504668
Node 30: X Displacement = -109.69290208880224, Y Displacement = 1900.8966058027106
Node 31: X Displacement = -96.30804685524357, Y Displacement = 1773.5373061679295
Node 32: X Displacement = -87.62166239597641, Y Displacement = 1603.5371645196437
Node 33: X Displacement = -74.67863860098434, Y Displacement = 1463.071757520174
Node 34: X Displacement = -62.83749031973655, Y Displacement = 1308.4812689994678
Node 35: X Displacement = -103.61232977829513, Y Displacement = 1621.3536312400804
Node 36: X Displacement = -84.91705994434169, Y Displacement = 1617.4545349722675
Node 37: X Displacement = -66.51980924154029, Y Displacement = 1621.494975952147
Node 38: X Displacement = -66.02721221489952, Y Displacement = 1380.8232063498767
Node 39: X Displacement = -47.94056746327443, Y Displacement = 1394.0491017187715
Node 40: X Displacement = -30.180382003535907, Y Displacement = 1412.0732677488052
Node 41: X Displacement = -32.972150956192586, Y Displacement = 1578.2749045099993
Node 42: X Displacement = -19.32724017997512, Y Displacement = 1641.3332294878971
Node 43: X Displacement = -0.09640278685113915, Y Displacement = 1541.06342449442455
Node 44: X Displacement = -63.83333869657605, Y Displacement = 1073.5973065829692
Node 45: X Displacement = -89.33978405838833, Y Displacement = 708.2682175135725
Node 46: X Displacement = -68.5419395045329, Y Displacement = 679.418364375998
Node 47: X Displacement = -50.195706086044815, Y Displacement = 698.0677357593512
Node 48: X Displacement = -28.86890656669278, Y Displacement = 626.3679306573932
Node 49: X Displacement = -26.201147325953333, Y Displacement = 367.35849226064994
Node 50: X Displacement = -19.115599602160145, Y Displacement = 132.2410259151241
Node 51: X Displacement = 0.0, Y Displacement = 0.0
Node 52: X Displacement = 0.0, Y Displacement = 0.0
Node 53: X Displacement = 8.17839864940458, Y Displacement = -289.91455506481725
Node 54: X Displacement = 14.829622948820022, Y Displacement = -384.5164200642993
Node 55: X Displacement = 138.9884632094206, Y Displacement = -357.9882984285013
Node 56: X Displacement = -198.77023006982475, Y Displacement = 350.98236800593156
Node 57: X Displacement = -179.65928167047284, Y Displacement = 356.78302173793196
Node 58: X Displacement = -220.06521697250815, Y Displacement = 720.8810650574907
Node 59: X Displacement = -261.7538748792876, Y Displacement = 373.9488604464692
Node 60: X Displacement = -165.27032777012826, Y Displacement = 335.4364544652151
Node 61: X Displacement = -83.65279870227114, Y Displacement = 434.50061300600106
Node 62: X Displacement = -121.66201672852998, Y Displacement = 858.0981875453017
Node 63: X Displacement = -218.34258066440026, Y Displacement = 1363.7488905305709
Node 64: X Displacement = -203.66336919127798, Y Displacement = 1484.7889777013054
Node 65: X Displacement = -141.55121424672524, Y Displacement = 1737.900088971842
Node 66: X Displacement = -74.71106141093715, Y Displacement = 1639.564856433624
Node 67: X Displacement = -17.99428849587402, Y Displacement = 1888.8360902454165
Node 68: X Displacement = 50.06529426058855, Y Displacement = 2076.052443031832
Node 69: X Displacement = 2.3633871586452524, Y Displacement = 1791.82576302661
Node 70: X Displacement = 17.284945500826122, Y Displacement = 1685.2412052864888
Node 71: X Displacement = 44.36247540254964, Y Displacement = 1438.6910553848402
Node 72: X Displacement = 47.933988793578145, Y Displacement = 1785.0439994155774
Node 73: X Displacement = -22.54037999182316, Y Displacement = 1901.3831689243748
Node 74: X Displacement = 29.754845841636193, Y Displacement = 2055.2778355234886
Node 75: X Displacement = 67.7393864945806, Y Displacement = 1757.605338607494
Node 76: X Displacement = 87.83067997485387, Y Displacement = 2109.5793208994005
Node 77: X Displacement = 158.33301001562123, Y Displacement = 1956.05769671271
Node 78: X Displacement = 190.59373303466344, Y Displacement = 1770.3177950303889
Node 79: X Displacement = 221.98251230924163, Y Displacement = 1555.4714897914776
Node 80: X Displacement = 140.15455592380445, Y Displacement = 1845.7474002809638
Node 81: X Displacement = 151.09696170635712, Y Displacement = 1944.0354002408358
Node 82: X Displacement = 169.0561458550683, Y Displacement = 1796.2275733400088
Node 83: X Displacement = 173.9444009830184, Y Displacement = 1616.4336659520643
Node 84: X Displacement = 259.5437497499354, Y Displacement = 1492.5662576798952
Node 85: X Displacement = 262.8397107955778, Y Displacement = 1277.0389362286678
Node 86: X Displacement = 176.3637332278191, Y Displacement = 1679.3523518053596
Node 87: X Displacement = 124.11704439630944, Y Displacement = 1627.085588792253
Node 88: X Displacement = 86.32033227449638, Y Displacement = 1656.8647304383421
Node 89: X Displacement = 11.587517989343601, Y Displacement = 1351.0958528876674
Node 90: X Displacement = 74.91599941034187, Y Displacement = 1396.9319108084114
Node 91: X Displacement = 12.932390475022972, Y Displacement = 1394.168365219348
Node 92: X Displacement = 10.21674107178017, Y Displacement = 1591.1373434752309
Node 93: X Displacement = 56.81993205985844, Y Displacement = 1662.0413511139209
Node 94: X Displacement = 139.4373519482723, Y Displacement = 1595.6917577098093
Node 95: X Displacement = -30.88959957802895, Y Displacement = 1058.2899473606774
Node 96: X Displacement = -144.5263774111846, Y Displacement = 665.8187106980207
Node 97: X Displacement = -140.73360738131763, Y Displacement = 674.655633585349
Node 98: X Displacement = -82.18626305937983, Y Displacement = 709.4053970789278
Node 99: X Displacement = -11.725127700868715, Y Displacement = 649.9766169532958
Node 100: X Displacement = 5.500429708451393, Y Displacement = 364.4322965310505
Node 101: X Displacement = 42.49694021892959, Y Displacement = 120.11663818536213
Node 102: X Displacement = 0.0, Y Displacement = 0.0
Node 1: X Displacement = 0.0, Y Displacement = 0.0
Node 2: X Displacement = 37.509778763165315, Y Displacement = 130.9869707545326
Node 3: X Displacement = 32.72633666926651, Y Displacement = -116.64624923125233
Node 4: X Displacement = 71.97969993544537, Y Displacement = -110.37728800544416
Node 5: X Displacement = 99.04587238540182, Y Displacement = -216.64089694728003
Node 6: X Displacement = 140.01166942279923, Y Displacement = -125.99977452574853
Node 7: X Displacement = 178.987979311878, Y Displacement = -129.89155890982258
Node 8: X Displacement = 216.69112676647987, Y Displacement = 14.237916299798194
Node 9: X Displacement = 253.71285533064776, Y Displacement = 163.69637496299444
Node 10: X Displacement = 251.01098269604063, Y Displacement = -82.4207112443212

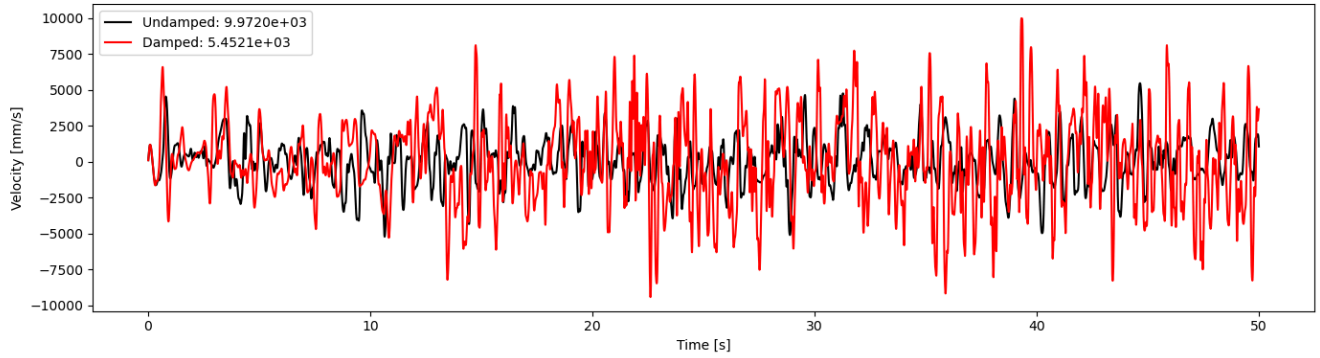
```
Node 11: X Displacement = 206.73687065161187, Y Displacement = -447.53813193632396
Node 12: X Displacement = 151.94436270063082, Y Displacement = -839.0979644456779
Node 13: X Displacement = 189.0333431688936, Y Displacement = -848.8536019123545
Node 14: X Displacement = 215.2896623090379, Y Displacement = -966.2223216830554
Node 15: X Displacement = 248.7611520887261, Y Displacement = -818.3888457208029
Node 16: X Displacement = 207.1533779622164, Y Displacement = -1187.6724574831492
Node 17: X Displacement = 202.77345581353737, Y Displacement = -1454.0291277899855
Node 18: X Displacement = 220.02399190256634, Y Displacement = -1636.5341594418212
Node 19: X Displacement = 253.4507669709644, Y Displacement = -1719.2739722075792
Node 20: X Displacement = 260.4399830833829, Y Displacement = -1952.5611034409424
Node 21: X Displacement = 293.5215603565254, Y Displacement = -1812.0085951724395
Node 22: X Displacement = 318.39489674391825, Y Displacement = -1968.799321455473
Node 23: X Displacement = 355.3897488516623, Y Displacement = -1886.612694538629
Node 24: X Displacement = 390.68730490012973, Y Displacement = -1786.743895496302
Node 25: X Displacement = 412.64659868370876, Y Displacement = -1590.391124791765
Node 26: X Displacement = 433.3467453858738, Y Displacement = -1391.1895496389602
Node 27: X Displacement = 255.20981879019502, Y Displacement = -742.8458589431895
Node 28: X Displacement = 271.3924036965326, Y Displacement = -560.9758583588434
Node 29: X Displacement = 265.3005141791632, Y Displacement = -282.4577044672797
Node 30: X Displacement = 264.7588187222104, Y Displacement = -28.49952802562378
Node 31: X Displacement = 294.85086932570863, Y Displacement = 35.15018121688349
Node 32: X Displacement = 325.6234460342326, Y Displacement = 86.5932870056365
Node 33: X Displacement = 257.1982496801466, Y Displacement = 518.2334471540621
Node 34: X Displacement = 266.838138817401, Y Displacement = 715.4880918858539
Node 35: X Displacement = 293.62918357392164, Y Displacement = 551.8511945034468
Node 36: X Displacement = 327.1756019300102, Y Displacement = 461.4698248803895
Node 37: X Displacement = 259.5797366517786, Y Displacement = -38.01346028371049
Node 38: X Displacement = 242.3245371865781, Y Displacement = -411.3005375467922
Node 39: X Displacement = 274.55466048152175, Y Displacement = -526.154326002239
Node 40: X Displacement = 295.00013051723374, Y Displacement = -752.0684950133375
Node 41: X Displacement = 328.5719093672402, Y Displacement = -758.4749433985866
Node 42: X Displacement = 308.0763761502307, Y Displacement = -1155.1499782404196
Node 43: X Displacement = 300.03184062277876, Y Displacement = -1514.0416827770684
Node 44: X Displacement = 298.96180622644346, Y Displacement = -1849.781322729485
Node 45: X Displacement = 304.49554499766936, Y Displacement = -1663.7972749096084
Node 46: X Displacement = 82.28103507139873, Y Displacement = -1031.4151892413815
Node 47: X Displacement = -20.013266461044996, Y Displacement = -573.6269324545004
Node 48: X Displacement = -20.441446474360202, Y Displacement = -368.5271395844961
Node 49: X Displacement = -11.772802236094444, Y Displacement = -201.31047713863617
Node 50: X Displacement = -33.82486616643464, Y Displacement = 75.15173703325165
Node 51: X Displacement = 0.0, Y Displacement = 0.0
Node 52: X Displacement = 0.0, Y Displacement = 0.0
Node 53: X Displacement = 161.6132389274258, Y Displacement = 229.15219728112137
Node 54: X Displacement = 6.529060370495443, Y Displacement = -180.92219684418473
Node 55: X Displacement = -165.8517165627427, Y Displacement = -68.87214029886498
Node 56: X Displacement = -258.692604188559, Y Displacement = -233.74176863547748
Node 57: X Displacement = -138.97830545306326, Y Displacement = -83.20761557803098
Node 58: X Displacement = 21.7737740279371, Y Displacement = -160.5439432268825
Node 59: X Displacement = 196.95954612625553, Y Displacement = 12.59224231352236
Node 60: X Displacement = 108.10906580010462, Y Displacement = 267.2729221472187
Node 61: X Displacement = 105.77650305475645, Y Displacement = -44.42805956519879
Node 62: X Displacement = 187.8749160086995, Y Displacement = -440.01890537833043
Node 63: X Displacement = 213.65098141597673, Y Displacement = -939.5637262841173
Node 64: X Displacement = 191.71605700029548, Y Displacement = -823.9044564303459
Node 65: X Displacement = 134.7271466877104, Y Displacement = -1028.1609279990412
Node 66: X Displacement = 219.2170780111714, Y Displacement = -689.49527664164
Node 67: X Displacement = 2.6473188767041043, Y Displacement = -1206.3792381956114
Node 68: X Displacement = 39.78647482612344, Y Displacement = -1469.8137228557632
Node 69: X Displacement = -42.198753796929225, Y Displacement = -1644.9192485053861
Node 70: X Displacement = -2.1499254528073193, Y Displacement = -1666.111886683028
Node 71: X Displacement = 52.72057075358244, Y Displacement = -2035.3800828230662
Node 72: X Displacement = 194.85657999432053, Y Displacement = -1737.0296699246944
Node 73: X Displacement = 274.3241896116723, Y Displacement = -2027.9069803240795
Node 74: X Displacement = 509.7578267854542, Y Displacement = -1885.9420697735973
Node 75: X Displacement = 466.0740626682962, Y Displacement = -1810.304175661454
Node 76: X Displacement = 450.87520463397743, Y Displacement = -1591.7526278576688
Node 77: X Displacement = 548.1220042400096, Y Displacement = -1532.7910631799014
Node 78: X Displacement = -113.79078921415676, Y Displacement = -548.5163410578319
Node 79: X Displacement = -82.16449823804763, Y Displacement = -553.8347987371693
Node 80: X Displacement = 119.95063766115649, Y Displacement = -272.2099551893817
Node 81: X Displacement = 147.6890565730815, Y Displacement = 16.18106323322106
Node 82: X Displacement = 252.5269032015602, Y Displacement = 37.36761022002151
Node 83: X Displacement = 277.763214277118, Y Displacement = -3.6745198999403414
Node 84: X Displacement = 255.54337018292944, Y Displacement = 575.8215685149551
Node 85: X Displacement = 437.15843850749695, Y Displacement = 800.2211674910591
Node 86: X Displacement = 346.5896167450358, Y Displacement = 535.8188331693766
Node 87: X Displacement = 444.6260875438782, Y Displacement = 566.500451578369
Node 88: X Displacement = 558.0403509168955, Y Displacement = -54.201152481803284
Node 89: X Displacement = 438.36515484101653, Y Displacement = -462.70562897788307
Node 90: X Displacement = 682.3253641857192, Y Displacement = -458.93903241368145
Node 91: X Displacement = 608.0633636554305, Y Displacement = -776.5312255312842
Node 92: X Displacement = 567.2845728155096, Y Displacement = -654.2342185414137
Node 93: X Displacement = 639.0102521855162, Y Displacement = -1138.517090951418
Node 94: X Displacement = 695.1718189990152, Y Displacement = -1481.352876023817
Node 95: X Displacement = 591.7719352236707, Y Displacement = -1943.1922124176863
Node 96: X Displacement = 640.527478503479, Y Displacement = -1755.794458047475
Node 97: X Displacement = 285.19004641258687, Y Displacement = -965.0762051734044
Node 98: X Displacement = 291.49341289024125, Y Displacement = -489.8766331344124
Node 99: X Displacement = 269.2689244339139, Y Displacement = -339.5352973263397
Node 100: X Displacement = 212.34281242772423, Y Displacement = -212.5421192838872
Node 101: X Displacement = 80.40649882898971, Y Displacement = 151.21975754033687
Node 102: X Displacement = 0.0, Y Displacement = 0.0

Total time (s): 20.7188
```

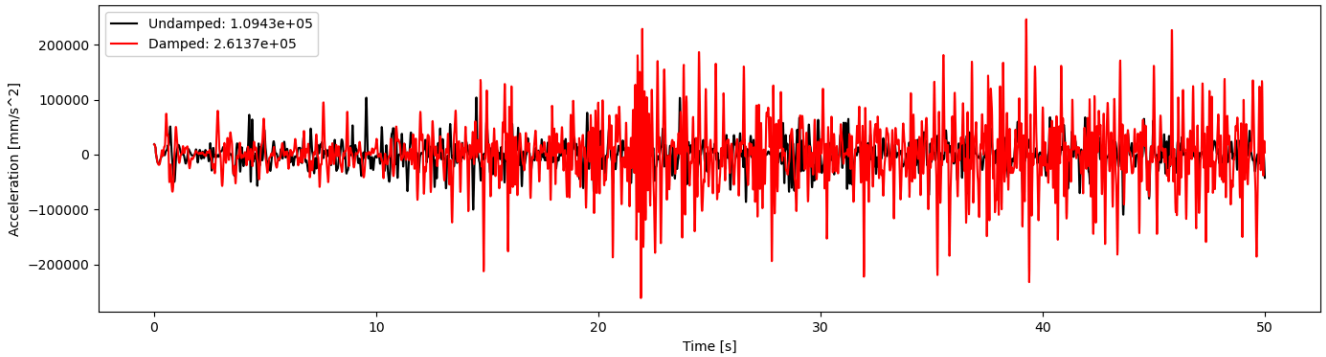
In [19]:
```python
# Plotting the time history
TIME = np.arange(dt, duration+2*dt, dt)

plot_time_history(TIME, DISP_X_undamped, DISP_X_damped, DISP_Y_undamped, DISP_Y_damped,
                  VELOCITY_undamped, VELOCITY_damped, ACCELERATION_undamped, ACCELERATION_damped,
                  BASE_REACTION_X_undamped, BASE_REACTION_X_damped, BASE_REACTION_Y_undamped, BASE_REACTION_Y_damped)
```

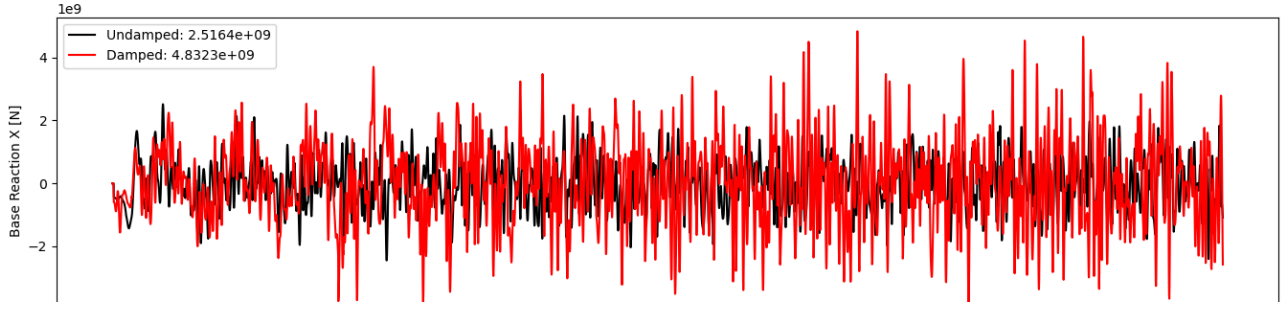
Time History of Free Vibration Analysis: Damped vs Undamped

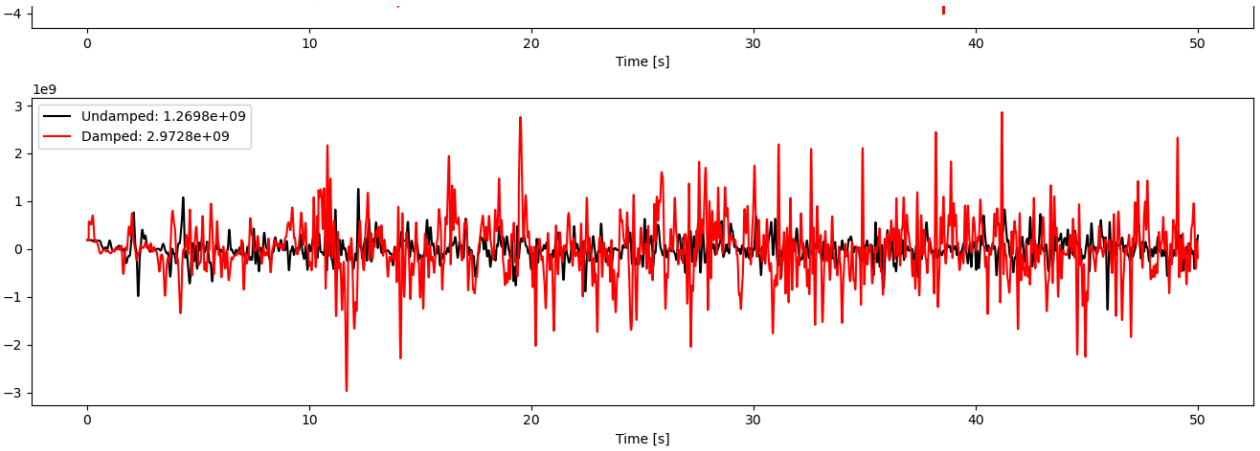




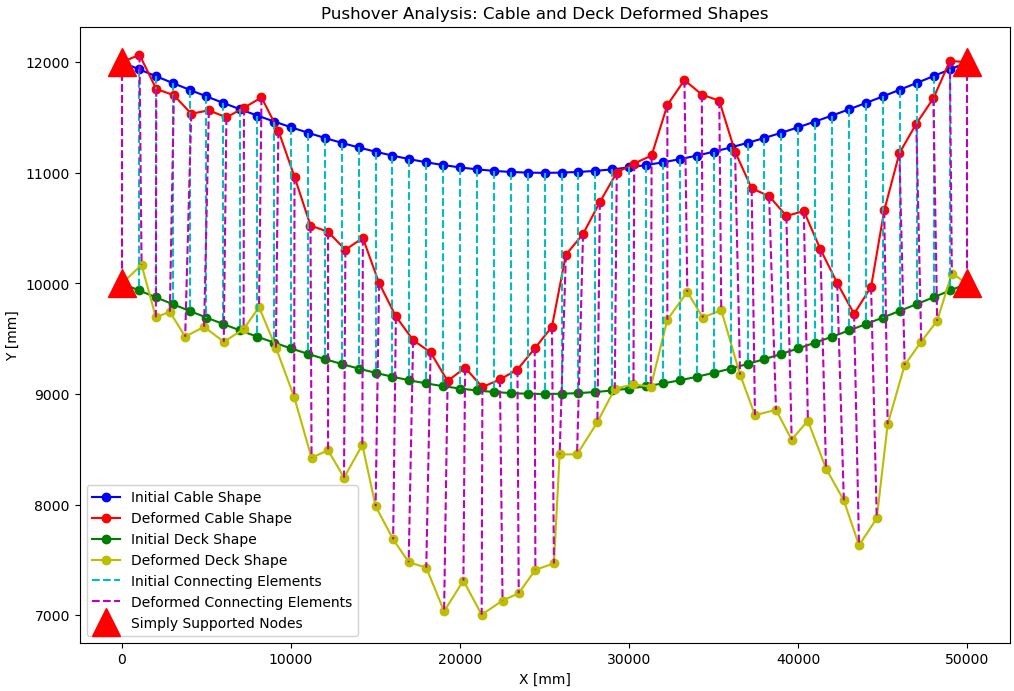


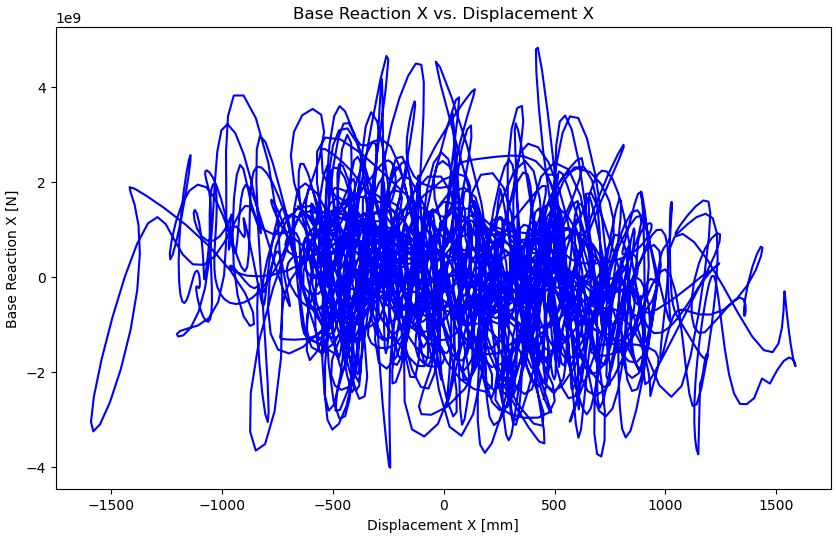


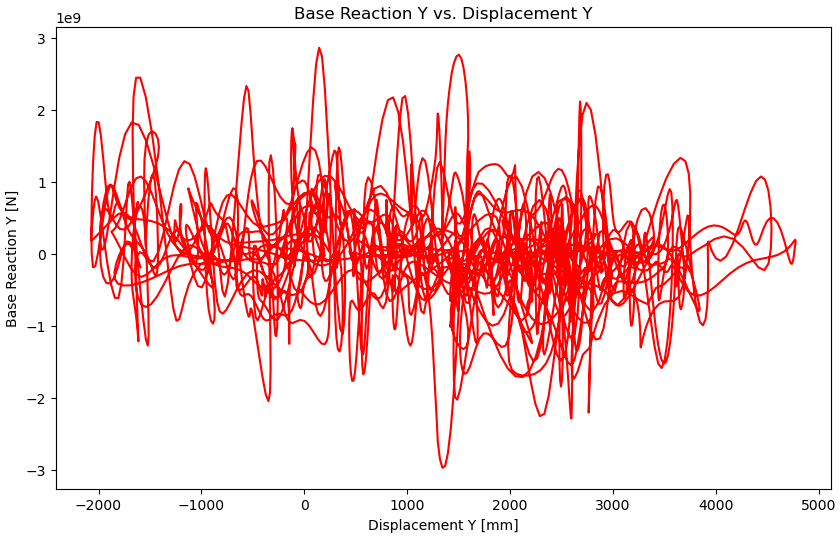




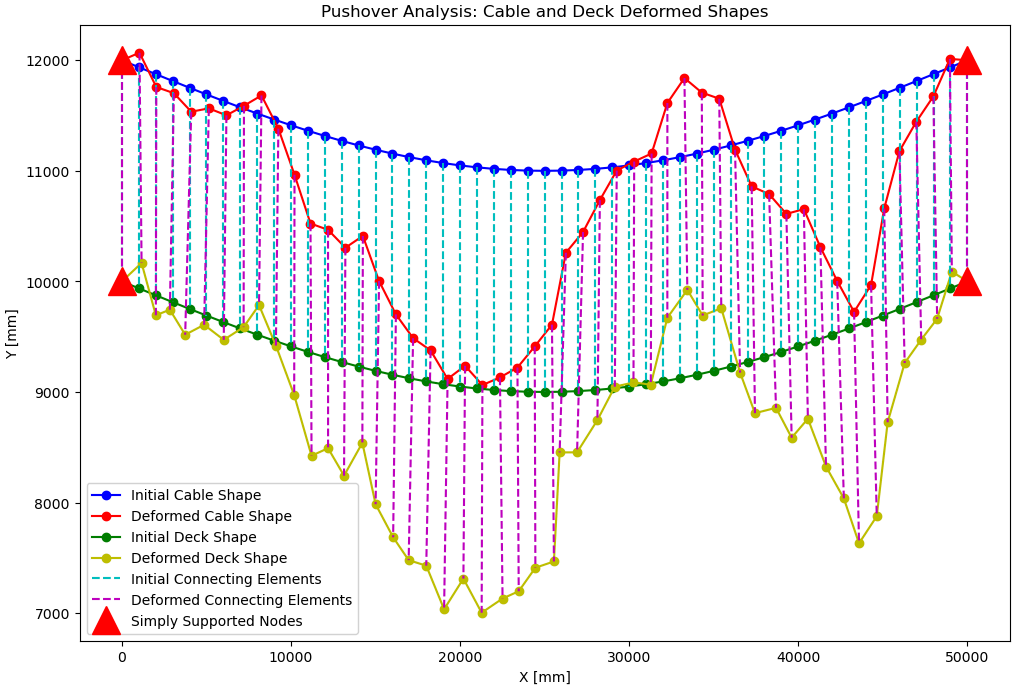
In [20]:
```
# Plot damped results
plot_shapes(initial_coords, DISPLACEMENTS_damped)
plot_reactions(DISP_X_damped, BASE_REACTION_X_damped, DISP_Y_damped, BASE_REACTION_Y_damped)
```

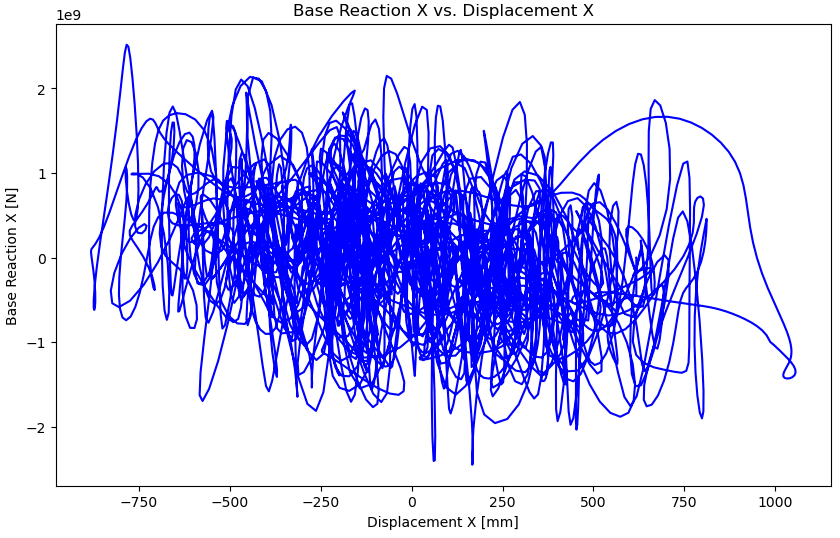


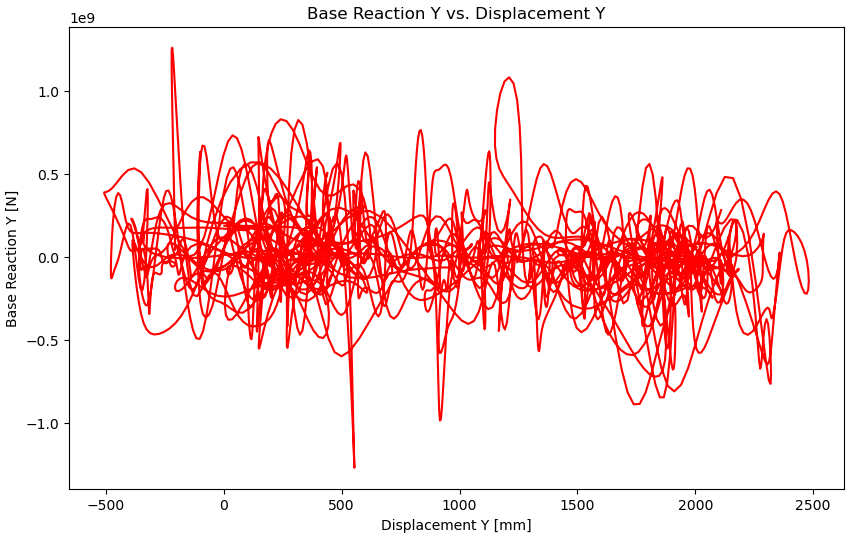




In [21]:
```
# Plot undamped results
plot_shapes(initial_coords, DISPLACEMENTS_undamped)
plot_reactions(DISP_X_undamped, BASE_REACTION_X_undamped, DISP_Y_undamped, BASE_REACTION_Y_undamped)
```







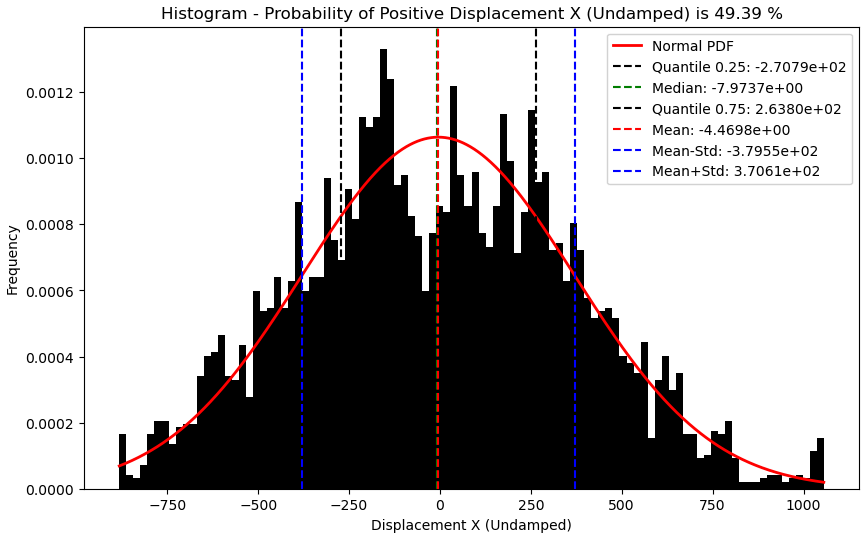
In [22]:

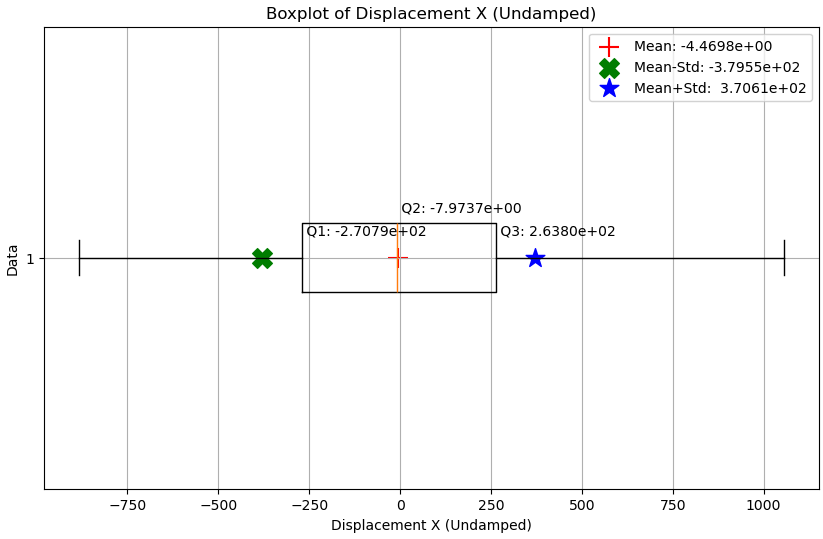
```python
def HISROGRAM_BOXPLOT(X, HISTO_COLOR, LABEL):
    import numpy as np
    import matplotlib.pyplot as plt
    X = np.array(X)
    print("-------------------------")
    from scipy.stats import skew, kurtosis
    MINIMUM = np.min(X)
    MAXIMUM = np.max(X)
    #MODE = max(set(X), key=list(X).count)
    MEDIAN = np.quantile(X, .50)#q2
    MEAN = np.mean(X)
    STD = np.std(X)
    q1 = np.quantile(X, .25)
    q3 = np.quantile(X, .75)
    SKEW = skew(X)
    KURT = kurtosis(X)
    #SKEW = (MEAN - MODE) / STD
    #KURT = (np.mean((X - MEAN)**4) / STD**4)
    # Estimate confidence intervals of the output variable
    lower_bound = np.quantile(X, .05)
    upper_bound = np.quantile(X, .95)
    print("Box-Chart Datas: ")
    print(f'Minimum: {MINIMUM:.4e}')
    print(f'First quartile: {q1:.4e}')
    #print(f'Mode: {MODE:.4e}')
    print(f'Median: {MEDIAN:.4e}')
    print(f'Mean: {MEAN:.4e}')
    print(f'Std: {STD:.4e}')
    print(f'Third quartile: {q3:.4e}')
    print(f'Maximum: {MAXIMUM :.4e}')
    print(f'Skewness: {skew(X) :.4e}')
    print(f'kurtosis: {kurtosis(X) :.4e}')
    print(f"90% Confidence Interval: ({lower_bound:.4e}, {upper_bound:.4e})")
    print("-------------------------")

    plt.figure(figsize=(10,6))
    # Plot histogram of data
    count, bins, ignored = plt.hist(X, bins=100, color=HISTO_COLOR, density=True, align='mid')#, edgecolor="black"

    # Plot lognormal PDF
    x = np.linspace(min(bins), max(bins), 10000)
    pdf = (np.exp(-(x - MEAN)**2 / (2 * STD**2)) / (STD * np.sqrt(2 * np.pi)))
    plt.plot(x, pdf, linewidth=2, color='r', label="Normal PDF")

    # Plot vertical lines for risk measures
    plt.axvline(q1, color="black", linestyle="--", label=f"Quantile 0.25: {q1:.4e}")
    plt.axvline(MEDIAN, color="green", linestyle="--", label=f"Median: {MEDIAN:.4e}")
    plt.axvline(q3, color="black", linestyle="--", label=f"Quantile 0.75: {q3:.4e}")
    #plt.axvline(MODE, color="purple", linestyle="--", label=f"Mode: {MODE:.4e}")
    plt.axvline(MEAN, color="red", linestyle="--", label=f"Mean: {MEAN:.4e}")
    plt.axvline(MEAN-STD, color="blue", linestyle="--", label=f"Mean-Std: {MEAN-STD:.4e}")
    plt.axvline(MEAN+STD, color="blue", linestyle="--", label=f"Mean+Std: {MEAN+STD:.4e}")
    plt.xlabel(LABEL)
    plt.ylabel("Frequency")
    prob = np.sum(X > 0) / len(X)
    plt.title(f"Histogram - Probability of Positive {LABEL} is {100*prob:.2f} %")
    plt.legend()
    #plt.grid()
    plt.show()

    #Plot boxplot with outliers
    plt.figure(figsize=(10,6))
    plt.boxplot(X, vert=0)
    # Write the quartile data on the chart
    plt.text(q1, 1.05, f" Q1: {q1:.4e}")
    plt.text(MEDIAN, 1.1, f" Q2: {MEDIAN:.4e}")
    plt.text(q3, 1.05, f" Q3: {q3:.4e}")
    #plt.text(MODE, 1.15, f" Mode: {MODE:.4e}")

    #plt.text(MEAN, 0.9, f" Mean: {MEAN:.4e}")
    #plt.text(MEAN-STD, 0.9, f" Mean-Std: {MEAN-STD:.4e}")
    #plt.text(MEAN+STD, 0.9, f" Mean+Std: {MEAN+STD:.4e}")
    plt.scatter(MEAN, 1, color="red", marker="+", s=200, label=f"Mean: {MEAN:.4e}")
    plt.scatter(MEAN-STD, 1, color="green", marker="X", s=200, label=f"Mean-Std: {MEAN-STD:.4e}")
    plt.scatter(MEAN+STD, 1, color="blue", marker="*", s=200, label=f"Mean+Std:  {MEAN+STD:.4e}")
    plt.xlabel(LABEL)
    plt.ylabel("Data")
    plt.title(f"Boxplot of {LABEL}")
    plt.legend()
```

```
    plt.grid()
    plt.show()
# ------------------------------------------------
def HISTOGRAM_BOXPLOT_PLOTLY( DATA, XLABEL='X', TITLE='A', COLOR='cyan'):
    # Plotting histogram and boxplot
    import plotly.express as px
    fig = px.histogram(x=DATA, marginal="box", color_discrete_sequence=[COLOR])
    fig.update_layout(title=TITLE, xaxis_title=XLABEL, yaxis_title="Frequency")
    fig.show()
```

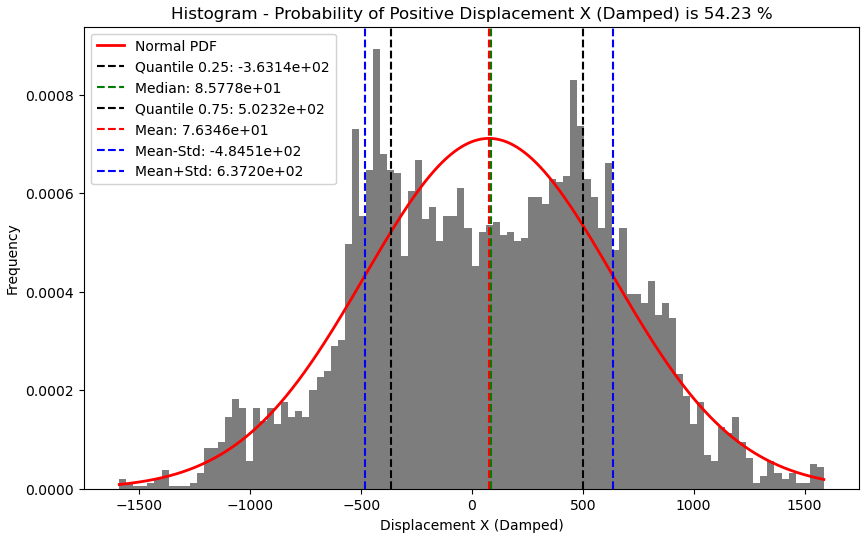
In [31]:
```
HISROGRAM_BOXPLOT(DISP_X_undamped, HISTO_COLOR='black', LABEL='Displacement X (Undamped)')
HISROGRAM_BOXPLOT(DISP_X_damped, HISTO_COLOR='grey', LABEL='Displacement X (Damped)')
#HISTOGRAM_BOXPLOT_PLOTLY(DISP_X_undamped, XLABEL='Displacement X (Undamped)', TITLE='Displacement X (Undamped)', COLOR='black')
#HISTOGRAM_BOXPLOT_PLOTLY(DISP_X_damped, XLABEL='Displacement X (Damped)', TITLE='Displacement X (Damped)', COLOR='grey')
```

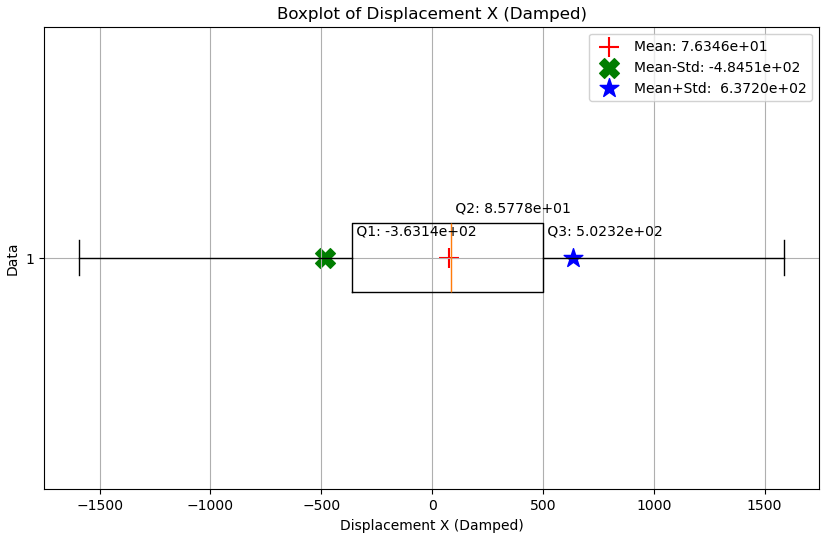
```
-------------------------
Box-Chart Datas:
Minimum: -8.8200e+02
First quartile: -2.7079e+02
Median: -7.9737e+00
Mean: -4.4698e+00
Std: 3.7508e+02
Third quartile: 2.6380e+02
Maximum: 1.0557e+03
Skewness: 1.1076e-01
kurtosis: -4.2244e-01
90% Confidence Interval: (-6.1978e+02, 6.1751e+02)
-------------------------
```





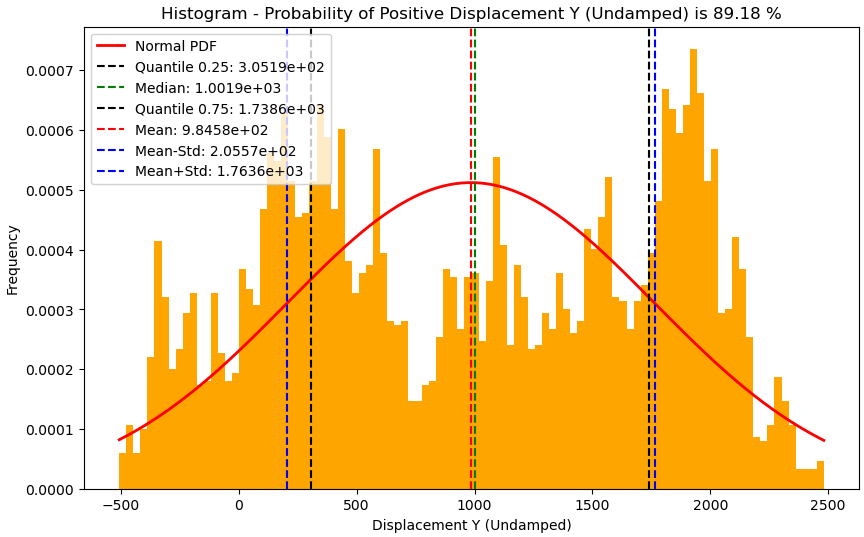

```
-------------------------
Box-Chart Datas:
Minimum: -1.5912e+03
First quartile: -3.6314e+02
Median: 8.5778e+01
Mean: 7.6346e+01
Std: 5.6086e+02
Third quartile: 5.0232e+02
Maximum: 1.5870e+03
Skewness: -6.3644e-02
kurtosis: -5.1354e-01
90% Confidence Interval: (-8.6030e+02, 9.3185e+02)
-------------------------
```

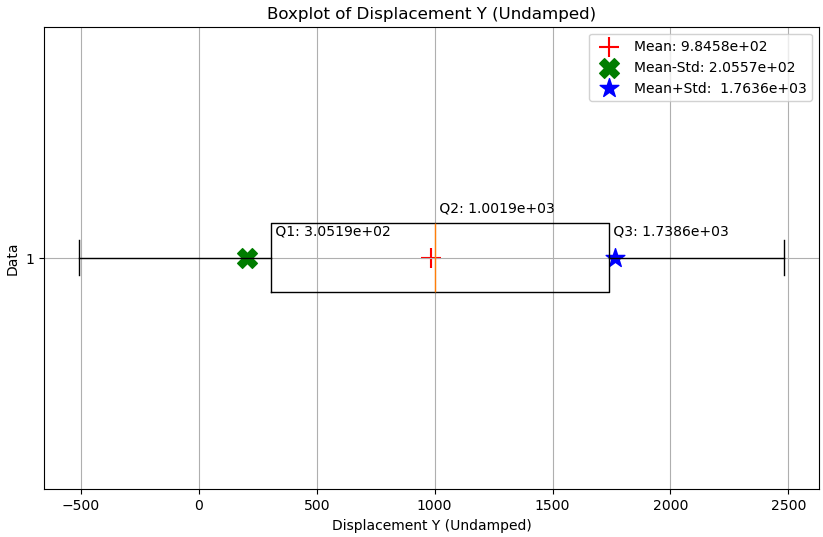




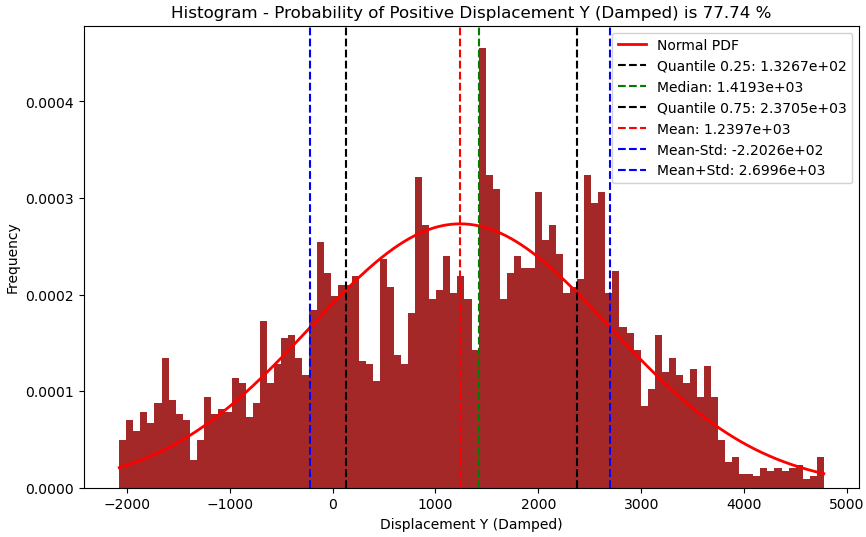
In [32]:
```
HISROGRAM_BOXPLOT(DISP_Y_undamped, HISTO_COLOR='orange', LABEL='Displacement Y (Undamped)')
HISROGRAM_BOXPLOT(DISP_Y_damped, HISTO_COLOR='brown', LABEL='Displacement Y (Damped)')
#HISTOGRAM_BOXPLOT_PLOTLY(DISP_Y_undamped, XLABEL='Displacement Y (Undamped)', TITLE='Displacement Y (Undamped)', COLOR='orange')
#HISTOGRAM_BOXPLOT_PLOTLY(DISP_Y_damped, XLABEL='Displacement Y (Damped)', TITLE='Displacement Y (Damped)', COLOR='brown')
```

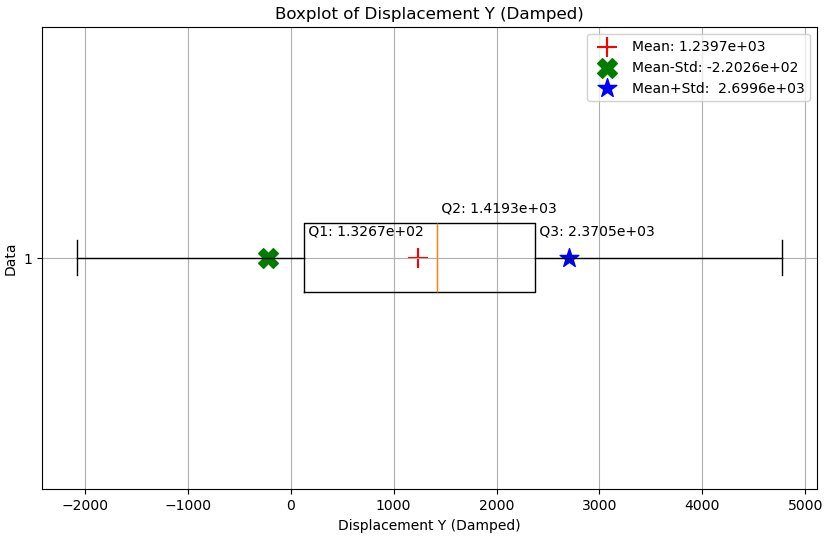
```
-------------------------
Box-Chart Datas:
Minimum: -5.0680e+02
First quartile: 3.0519e+02
Median: 1.0019e+03
Mean: 9.8458e+02
Std: 7.7901e+02
Third quartile: 1.7386e+03
Maximum: 2.4812e+03
Skewness: -3.5038e-02
kurtosis: -1.2825e+00
90% Confidence Interval: (-2.4386e+02, 2.1099e+03)
-------------------------
```





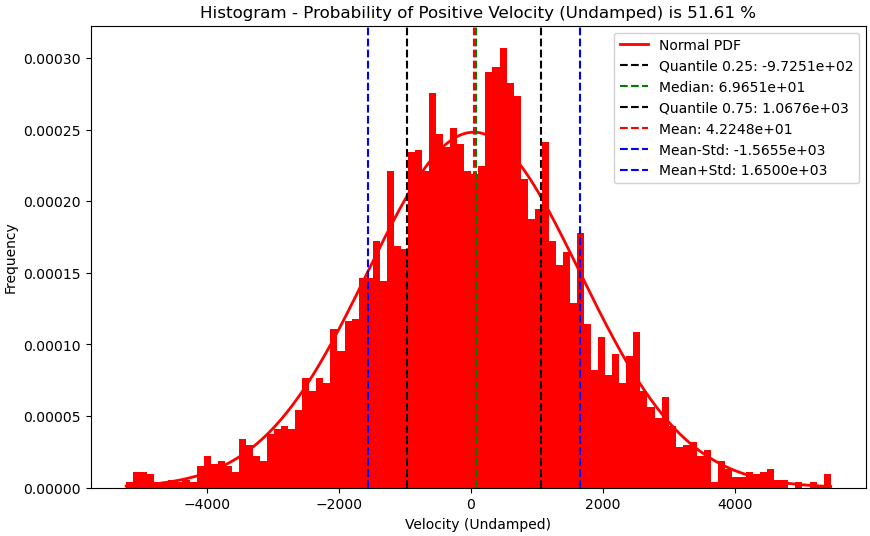

```
-------------------------
Box-Chart Datas:
Minimum: -2.0731e+03
First quartile: 1.3267e+02
Median: 1.4193e+03
Mean: 1.2397e+03
Std: 1.4599e+03
Third quartile: 2.3705e+03
Maximum: 4.7733e+03
Skewness: -2.0349e-01
kurtosis: -6.2274e-01
90% Confidence Interval: (-1.4367e+03, 3.4910e+03)
-------------------------
```

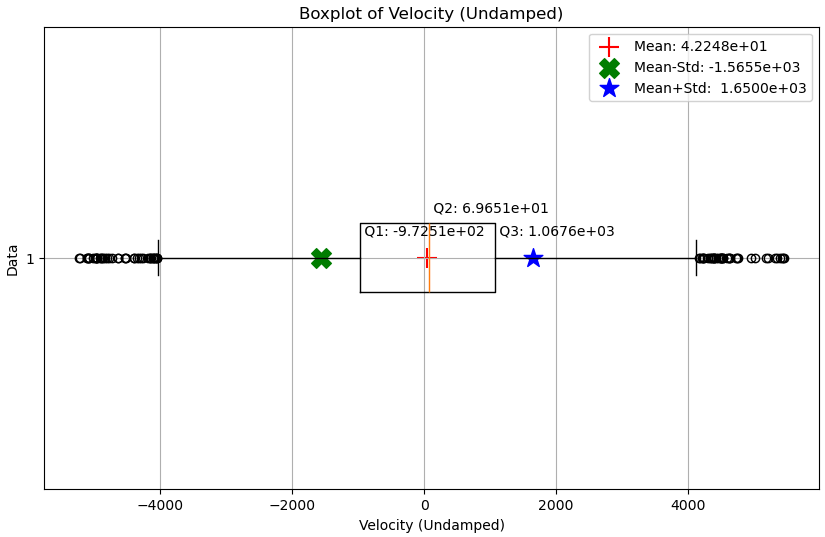




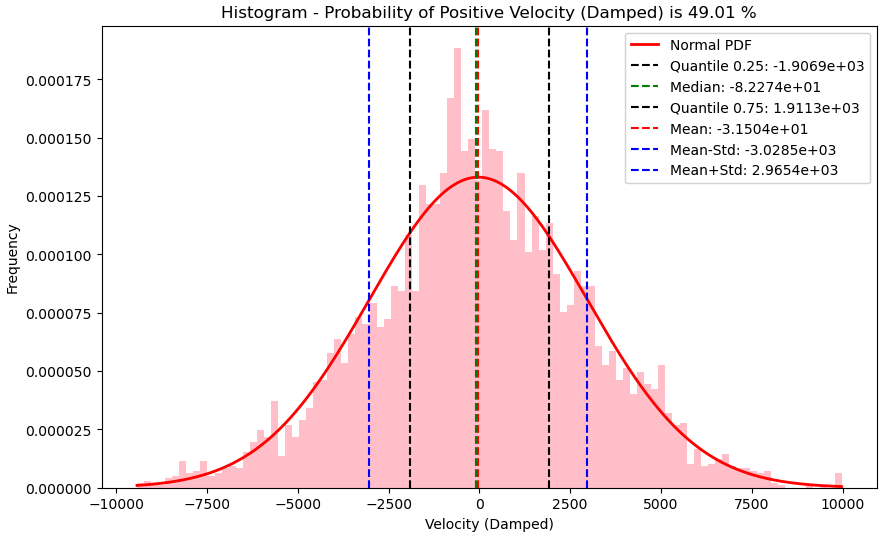
In [33]:
```
HISROGRAM_BOXPLOT(VELOCITY_undamped, HISTO_COLOR='red', LABEL='Velocity (Undamped)')
HISROGRAM_BOXPLOT(VELOCITY_damped, HISTO_COLOR='pink', LABEL='Velocity (Damped)')
#HISTOGRAM_BOXPLOT_PLOTLY(VELOCITY_undamped, XLABEL='Velocity (Undamped)', TITLE='Velocity (Undamped)', COLOR='red')
#HISTOGRAM_BOXPLOT_PLOTLY(VELOCITY_damped, XLABEL='Velocity (Damped)', TITLE='Velocity (Damped)', COLOR='pink')
```

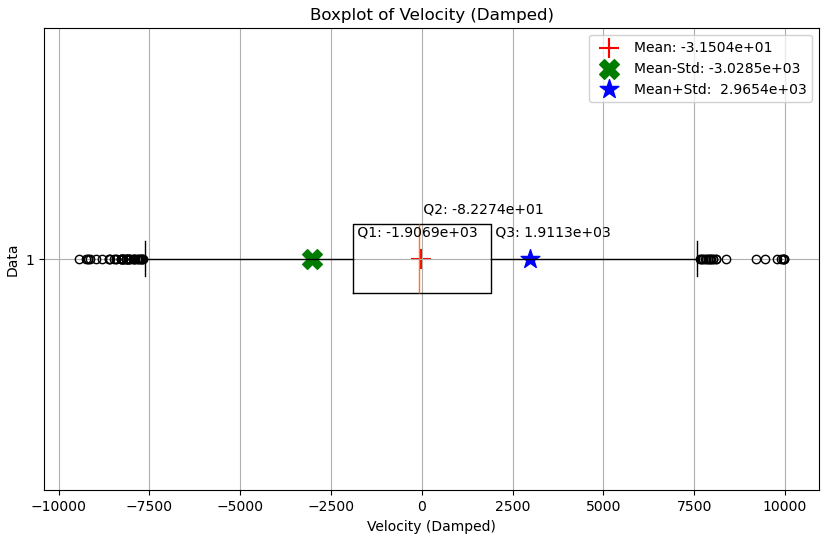
```
-------------------------
Box-Chart Datas:
Minimum: -5.2261e+03
First quartile: -9.7251e+02
Median: 6.9651e+01
Mean: 4.2248e+01
Std: 1.6077e+03
Third quartile: 1.0676e+03
Maximum: 5.4521e+03
Skewness: -2.7785e-02
kurtosis: 3.0273e-01
90% Confidence Interval: (-2.5773e+03, 2.6789e+03)
-------------------------
```






```
-------------------------
Box-Chart Datas:
Minimum: -9.4305e+03
First quartile: -1.9069e+03
Median: -8.2274e+01
Mean: -3.1504e+01
Std: 2.9969e+03
Third quartile: 1.9113e+03
Maximum: 9.9720e+03
Skewness: -2.8378e-02
kurtosis: 8.9536e-02
90% Confidence Interval: (-5.0629e+03, 4.9644e+03)
-------------------------
```

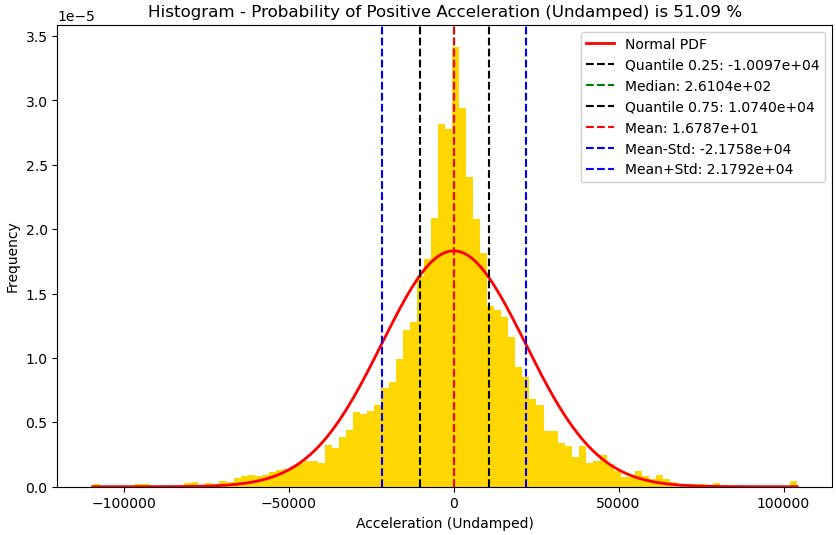


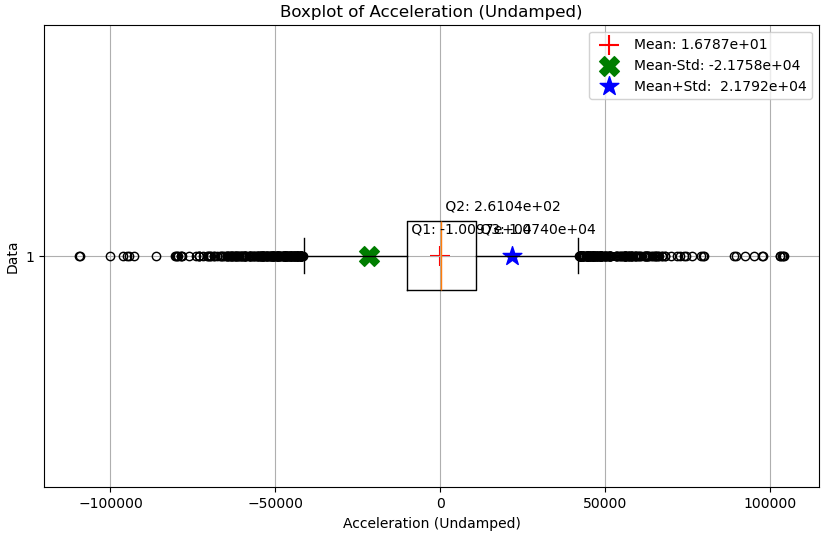


In [34]:
```
HISROGRAM_BOXPLOT(ACCELERATION_undamped, HISTO_COLOR='gold', LABEL='Acceleration (Undamped)')
HISROGRAM_BOXPLOT(ACCELERATION_damped, HISTO_COLOR='brown', LABEL='Acceleration (Damped)')
```

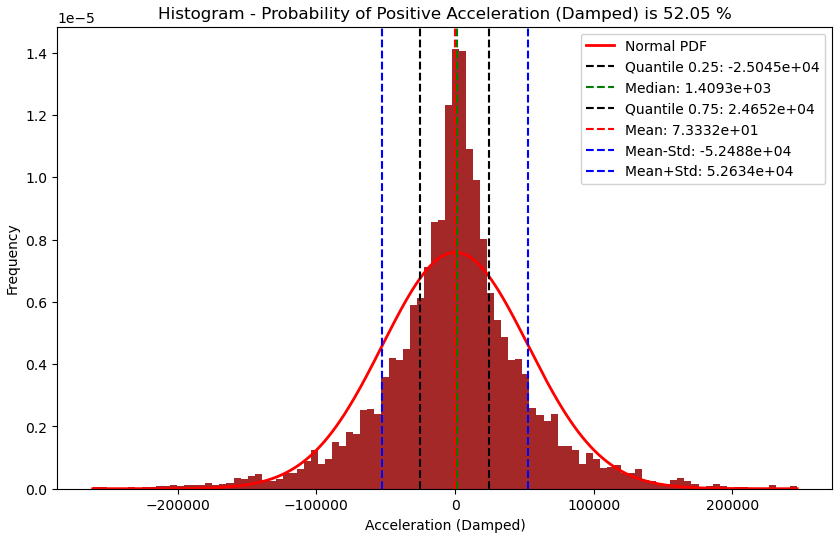
```
#HISTOGRAM_BOXPLOT_PLOTLY(ACCELERATION_undamped, XLABEL='Acceleration (Undamped)', TITLE='Acceleration (Undamped)', COLOR='gold')
#HISTOGRAM_BOXPLOT_PLOTLY(ACCELERATION_damped, XLABEL='Acceleration (Damped)', TITLE='Acceleration (Damped)', COLOR='brown')
```

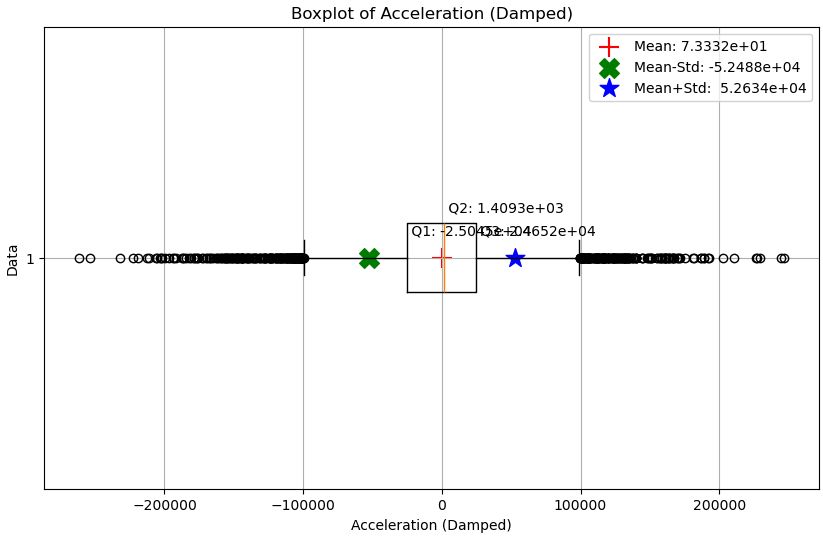
```
-------------------------
Box-Chart Datas:
Minimum: -1.0943e+05
First quartile: -1.0097e+04
Median: 2.6104e+02
Mean: 1.6787e+01
Std: 2.1775e+04
Third quartile: 1.0740e+04
Maximum: 1.0414e+05
Skewness: -7.8594e-02
kurtosis: 2.4479e+00
90% Confidence Interval: (-3.6803e+04, 3.5566e+04)
-------------------------
```





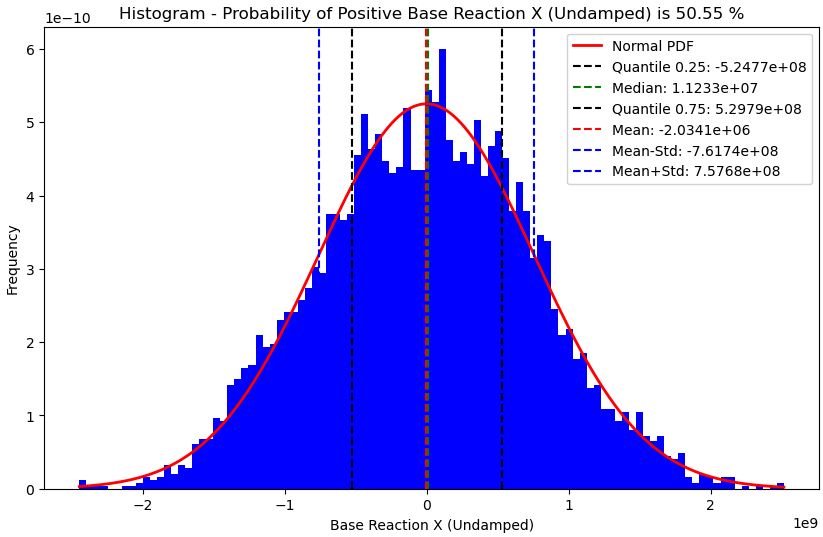

```
-------------------------
Box-Chart Datas:
Minimum: -2.6137e+05
First quartile: -2.5045e+04
Median: 1.4093e+03
Mean: 7.3332e+01
Std: 5.2561e+04
Third quartile: 2.4652e+04
Maximum: 2.4665e+05
Skewness: -1.2018e-01
kurtosis: 2.3742e+00
90% Confidence Interval: (-8.7129e+04, 8.6345e+04)
-------------------------
```

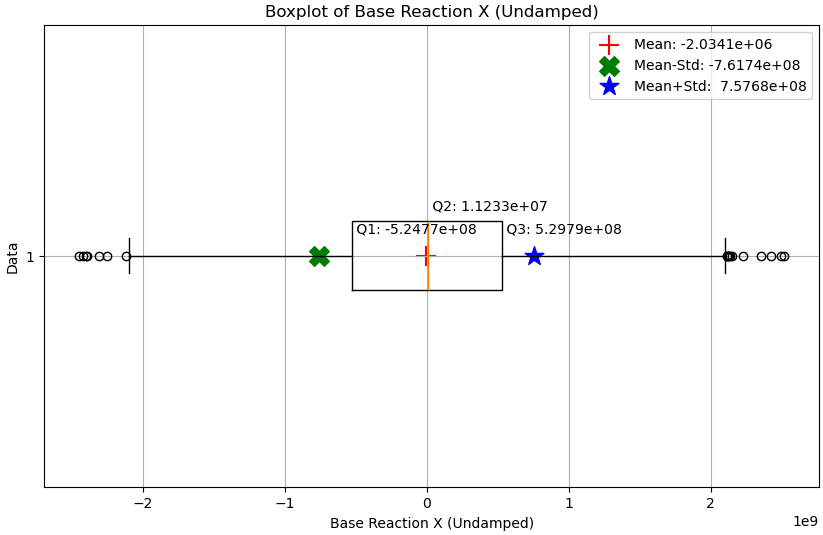




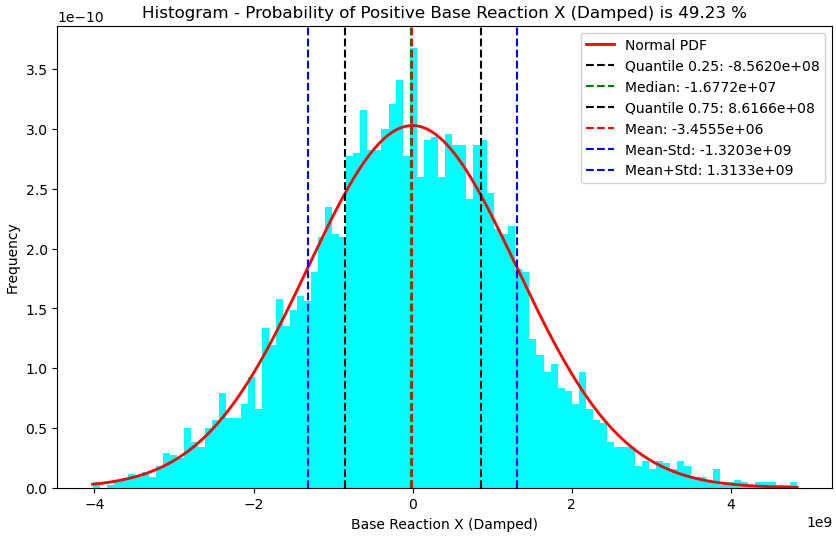
In [35]:
```
HISROGRAM_BOXPLOT(BASE_REACTION_X_undamped, HISTO_COLOR='blue', LABEL='Base Reaction X (Undamped)')
HISROGRAM_BOXPLOT(BASE_REACTION_X_damped, HISTO_COLOR='cyan', LABEL='Base Reaction X (Damped)')
#HISTOGRAM_BOXPLOT_PLOTLY(BASE_REACTION_X_undamped, XLABEL='Base Reaction X (Undamped)', TITLE='Base Reaction X (Undamped)', COLOR='blue')
#HISTOGRAM_BOXPLOT_PLOTLY(BASE_REACTION_X_damped, XLABEL='Base Reaction X (Damped)', TITLE='Base Reaction X (Damped)', COLOR='cyan')
```

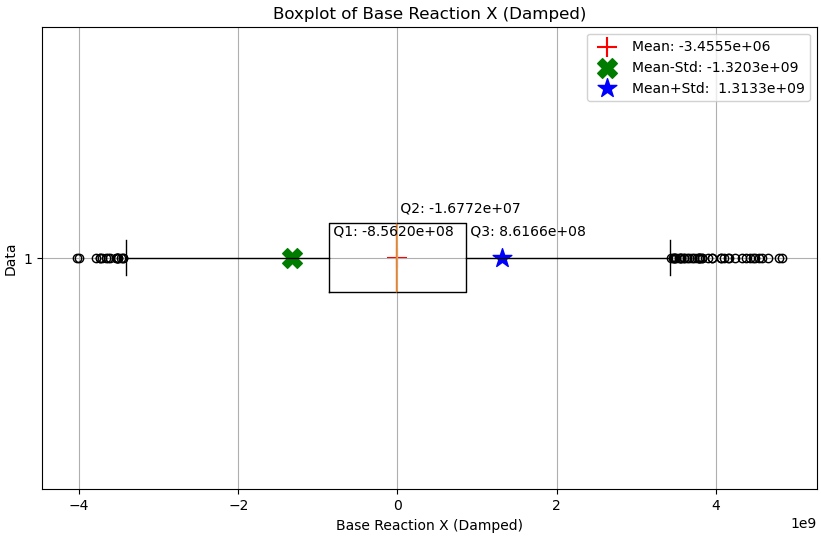
```
-------------------------
Box-Chart Datas:
Minimum: -2.4467e+09
First quartile: -5.2477e+08
Median: 1.1233e+07
Mean: -2.0341e+06
Std: 7.5971e+08
Third quartile: 5.2979e+08
Maximum: 2.5164e+09
Skewness: 1.2706e-02
kurtosis: -2.3883e-01
90% Confidence Interval: (-1.2687e+09, 1.2521e+09)
-------------------------
```





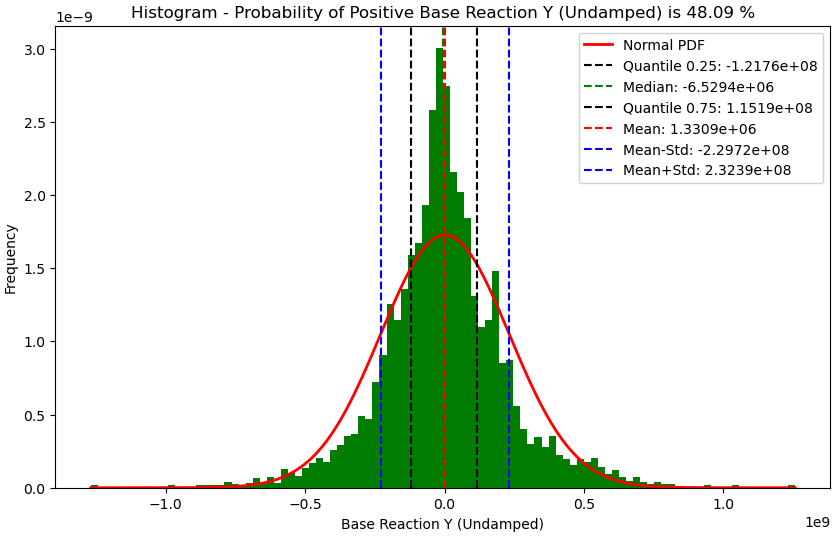

```
-------------------------
Box-Chart Datas:
Minimum: -4.0228e+09
First quartile: -8.5620e+08
Median: -1.6772e+07
Mean: -3.4555e+06
Std: 1.3168e+09
Third quartile: 8.6166e+08
Maximum: 4.8323e+09
Skewness: 1.0752e-01
kurtosis: 2.1594e-01
90% Confidence Interval: (-2.2036e+09, 2.1727e+09)
-------------------------
```

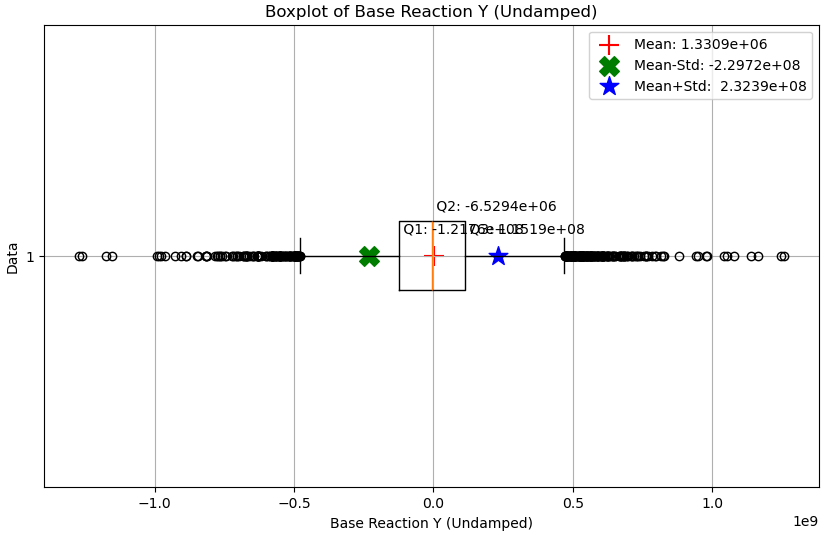




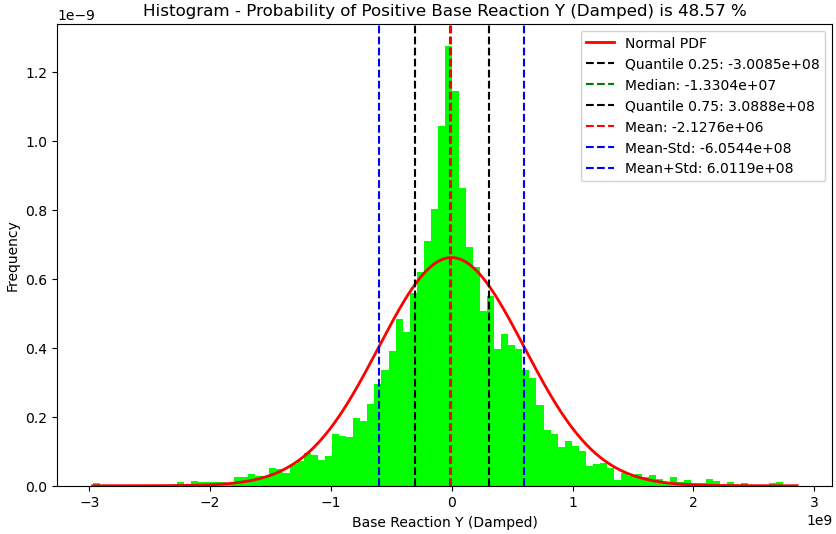
In [36]:
```
HISROGRAM_BOXPLOT(BASE_REACTION_Y_undamped, HISTO_COLOR='green', LABEL='Base Reaction Y (Undamped)')
HISROGRAM_BOXPLOT(BASE_REACTION_Y_damped, HISTO_COLOR='lime', LABEL='Base Reaction Y (Damped)')
#HISTOGRAM_BOXPLOT_PLOTLY(BASE_REACTION_Y_undamped, XLABEL='Base Reaction Y (Undamped)', TITLE='Base Reaction Y (Undamped)', COLOR='green')
#HISTOGRAM_BOXPLOT_PLOTLY(BASE_REACTION_Y_damped, XLABEL='Base Reaction Y (Damped)', TITLE='Base Reaction Y (Damped)', COLOR='lime')
```

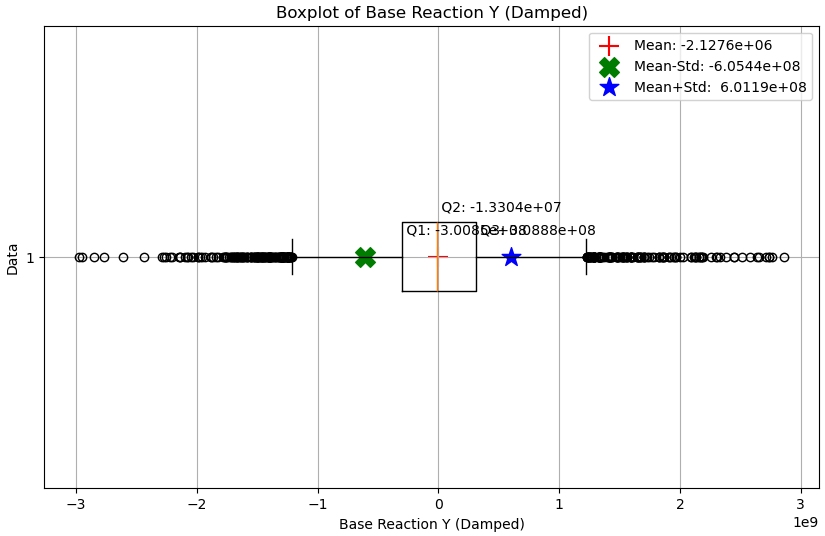
```
------------------------
Box-Chart Datas:
Minimum: -1.2698e+09
First quartile: -1.2176e+08
Median: -6.5294e+06
Mean: 1.3309e+06
Std: 2.3106e+08
Third quartile: 1.1519e+08
Maximum: 1.2578e+09
Skewness: 1.2928e-01
kurtosis: 2.8227e+00
90% Confidence Interval: (-3.6076e+08, 3.9602e+08)
------------------------
```






```
-------------------------
Box-Chart Datas:
Minimum: -2.9728e+09
First quartile: -3.0085e+08
Median: -1.3304e+07
Mean: -2.1276e+06
Std: 6.0332e+08
Third quartile: 3.0888e+08
Maximum: 2.8597e+09
Skewness: 8.0655e-02
kurtosis: 2.4432e+00
90% Confidence Interval: (-9.8636e+08, 9.7949e+08)
-------------------------
```





In [10]:
```
#         ##########################################################################################
#         #                                    IN THE NAME OF ALLAH                                  #
```

```
#         #                         DYNAMIC ANALYSIS OF CABLE SUSPENSION BRIDGE 02                 #
#         #--------------------------------------------------------------------------------------#
#         #                  THIS PROGRAM WRITTEN BY SALAR DELAVAR GHASHGHAEI (QASHQAI)           #
#         #                           EMAIL: salar.d.ghashghaei@gmail.com                        #
#         ###############################################################################################
```

In [14]:
```
# -------------------
#   DYNAMIC ANALYSIS
# -------------------
def DYNAMIC_ANALYSIS(damping, damping_ratio, LINEAR, periodTF, duration, dt, TOTAL_MASS, L, H1, H2, arc_depth, E_cable, Cable_Dia_01, Cable_Dia_02, Cable_Dia_03, num_nodes, MAX_ITERATI
    import openseespy.opensees as ops
    import numpy as np
    import matplotlib.pyplot as plt

    #GMfact = 9810              # standard acceleration of gravity or standard acceleration
    GMfact = 1;
    iv0 = 0.0                  # [mm/s] Initial velocity applied to the node
    st_iv0 = 0.0               # [s] Initial velocity applied starting time
    KE = 1000;                 # [N/mm] Lateral column Effective Stiffness
    A_cable_01 = (np.pi * Cable_Dia_01 **2) / 4  # [mm^2] Horizontal Longitudinal Cable Area Bottom
    A_cable_02 = (np.pi * Cable_Dia_02 **2) / 4  # [mm^2] Horizontal Longitudinal Cable Area Top
    A_cable_03 = (np.pi * Cable_Dia_03 **2) / 4  # [mm^2] Vertical Longitudinal Cable Area Top

    # Define model builder
    ops.wipe()
    ops.model('basic', '-ndm', 2, '-ndf', 2)
    dx = L / (num_nodes - 1)
    MASS = TOTAL_MASS / num_nodes
    # Create nodes
    # Create nodes
    for i in range(num_nodes):
        X = i * dx
        Y1 = H1 - arc_depth * np.sin(np.pi * X / L)
        Y2 = Y1 - H2
        ops.node(i + 1, X, Y1)
        ops.node(num_nodes + i + 1, i * dx, Y2)
        # Define mass
        ops.mass(num_nodes + i + 1, MASS, MASS, 0)

    # Define boundary conditions (fixed at both ends)
    ops.fix(1, 1, 1)              # TOP CABLE
    ops.fix(num_nodes, 1, 1)      # TOP CABLE
    ops.fix(num_nodes + 1, 1, 1)  # BOTTOM CABLE
    ops.fix(2 * num_nodes, 1, 1)  # BOTTOM CABLE

    # Define material properties
    if LINEAR == True:
        ops.uniaxialMaterial('Elastic', 1, E_cable)
        #                                 TENSION    COMPRESSION
        #ops.uniaxialMaterial('Elastic', 1, E_cable, 0, 0.5 * E_cable)
    if LINEAR == False:
        Fy_cable = 355   # [N/mm^2] Yield strength of the cable
        b0 = 0.01
        ops.uniaxialMaterial('Steel01', 1, Fy_cable, E_cable, b0)

    # Define truss elements
    for i in range(num_nodes - 1):
        ops.element('corotTruss', i + 1, i + 1, i + 2, A_cable_02, 1)  # Top Cable element
        ops.element('corotTruss', num_nodes + i + 1, num_nodes + i + 1, num_nodes + i + 2, A_cable_01, 1)  # Bottom Cable Element

    # Connect each node of the cable with the corresponding node of the deck using truss elements
    for i in range(num_nodes):
        ops.element('corotTruss', 2 * num_nodes + i + 1, i + 1, num_nodes + i + 1, A_cable_03, 1) # Vertical Cable Element

    #mid_node = num_nodes // 2 + 1
    mid_node = int(num_nodes + 0.5 * num_nodes)

    # Dynamic analysis setup
    ops.constraints('Transformation')
    ops.numberer('RCM')
    ops.system('UmfPack')
    ops.test('EnergyIncr', TOLERANCE, MAX_ITERATIONS)
    #ops.integrator('CentralDifference')
    ops.integrator('HHT', 0.9)
    #ops.integrator('Newmark', 0.5, 0.25)
    ops.algorithm('ModifiedNewton')

    # Define analysis type
    ops.analysis('Transient')

    # Define time series for input motion (Acceleration time history)
    ops.timeSeries('Path', 1, '-dt', 0.01, '-filePath', 'OPENSEES_SPRING_SEISMIC_01.txt', '-factor', GMfact, '-startTime', st_iv0) # SEISMIC-X
    ops.timeSeries('Path', 2, '-dt', 0.01, '-filePath', 'OPENSEES_SPRING_SEISMIC_02.txt', '-factor', GMfact) # SEISMIC-Y

    # Define load patterns
    # pattern UniformExcitation $patternTag $dof -accel $tsTag <-vel0 $vel0> <-fact $cFact>
    ops.pattern('UniformExcitation', 1, 1, '-accel', 1, '-vel0', iv0, '-fact', 1.0) # SEISMIC-X
    ops.pattern('UniformExcitation', 2, 2, '-accel', 2)                             # SEISMIC-Y

    # Perform eigenvalue analysis to determine modal periods
    if periodTF == True:
        eigenvalues01 = ops.eigen('-genBandArpack', 1) # eigenvalue mode 1
        #eigenvalues01 = ops.eigen('-fullGenLapack', 1) # eigenvalue mode 1
        #eigenvalues01 = ops.eigen('-symmBandLapack', 1) # eigenvalue mode 1
        #Omega = np.power(eigenvalues01, 0.5)
        #Omega = np.power(max(min(eigenvalues01), min(eigenvalues02), min(eigenvalues03)), 0.5)
        Omega = np.power(min(eigenvalues01), 0.5)
        modal_period = 2 * np.pi / np.sqrt(Omega)  # [Second]
        frequency = 1 / modal_period               # [Hertz]

    if periodTF == False:
        modal_period = 0.0           # [Second]
        frequency = 0.0              # [Hertz]

    if damping == True:
        # Calculate Rayleigh damping factors
        omega1 = np.sqrt(KE / TOTAL_MASS)
        omega2 = 2 * omega1  # Just an assumption for two modes
        a0 = damping_ratio * (2 * omega1 * omega2) / (omega1 + omega2)
        a1 = damping_ratio * 2 / (omega1 + omega2)
        # Apply Rayleigh damping
        ops.rayleigh(a0, a1, 0, 0)
```

```
    # OUTPUT DATA
    ops.recorder('Node', '-file', f"{SALAR_DIR}DTH_DYN.txt",'-time', '-node', mid_node, '-dof', 1,2, 'disp')# Displacement Time History
    ops.recorder('Node', '-file', f"{SALAR_DIR}VTH_DYN.txt",'-time', '-node', mid_node, '-dof', 1,2,3, 'vel')  # Velocity Time History
    ops.recorder('Node', '-file', f"{SALAR_DIR}ATH_DYN.txt",'-time', '-node', mid_node, '-dof', 1,2,3, 'accel') # Acceleration Time History
    ops.recorder('Node', '-file', f"{SALAR_DIR}BTH_DYN_01.txt",'-time', '-node', 1, '-dof', 1,2, 'reaction')# Base Shear Time History
    ops.recorder('Node', '-file', f"{SALAR_DIR}BTH_DYN_02.txt",'-time', '-node', num_nodes, '-dof', 1,2, 'reaction')# Base Shear Time History
    ops.recorder('Node', '-file', f"{SALAR_DIR}BTH_DYN_03.txt",'-time', '-node', num_nodes + 1, '-dof', 1,2, 'reaction')# Base Shear Time History
    ops.recorder('Node', '-file', f"{SALAR_DIR}BTH_DYN_04.txt",'-time', '-node', 2*num_nodes, '-dof', 1,2, 'reaction')# Base Shear

    DISPLACEMENTS = []
    DISP_X , DISP_Y, VELOCITY, ACCELERATION = [], [], [], []
    BASE_REACTION_X, BASE_REACTION_Y = [], []

    stable = 0
    current_time = 0.0
     # Perform the analysis with increments
    while stable == 0 and current_time < duration:
        OK = ops.analyze(1, dt)
        ANALYSIS(OK, 1, TOLERANCE, MAX_ITERATIONS)
        current_time = ops.getTime()
        DISPLACEMENTS.append([ops.nodeDisp(j + 1) for j in range(num_nodes * 2)])
        DISP_X.append(ops.nodeDisp(mid_node, 1))
        DISP_Y.append(ops.nodeDisp(mid_node, 2))
        VELOCITY.append(ops.nodeVel(mid_node, 2))
        ACCELERATION.append(ops.nodeAccel(mid_node, 2))
        x1 = ops.nodeResponse(1, 1, 6)
        x2 = ops.nodeResponse(num_nodes, 1, 6)
        x3 = ops.nodeResponse(num_nodes + 1, 1, 6)
        x4 = ops.nodeResponse(2*num_nodes, 1, 6)
        y1 = ops.nodeResponse(1, 2, 6)
        y2 = ops.nodeResponse(num_nodes, 2, 6)
        y3 = ops.nodeResponse(num_nodes + 1, 2, 6)
        y4 = ops.nodeResponse(2*num_nodes, 2, 6)
        BASE_REACTION_X.append(x1+x2+x3+x4) # CABLE REACION-X
        BASE_REACTION_Y.append(y1+y2+y3+y4) # CABLE REACION-Y
        KE =  BASE_REACTION_Y[-1] / DISP_Y[-1] # Effective Lateral Stiffness
        #print(f'STEP: {i+1}')

    # Get initial and final node coordinates for plotting
    initial_coords = np.array([ops.nodeCoord(i + 1) for i in range(num_nodes * 2)])
    deformed_coords = initial_coords + np.array(DISPLACEMENTS[-1])

    # Output all unconstrained node displacements
    for i in range(num_nodes * 2):
        disp = ops.nodeDisp(i + 1)
        print(f"Node {i + 1}: X Displacement = {disp[0]}, Y Displacement = {disp[1]}")

    ops.wipe()
    print("Period: ", modal_period)
    print("Frequency: ", frequency)
    if damping == False:
        print(f' Undamping Structure Dynamic Analysis Done.')
    if damping == True:
        print(f' Damping Structure Dynamic Analysis Done.')

    return initial_coords, deformed_coords, DISPLACEMENTS, VELOCITY, ACCELERATION, DISP_X, DISP_Y, BASE_REACTION_X, BASE_REACTION_Y
```

In [15]:

```
def plot_time_history(DISP_X_undamped, DISP_X_damped, DISP_Y_undamped, DISP_Y_damped,
                      VELOCITY_undamped, VELOCITY_damped, ACCELERATION_undamped, ACCELERATION_damped,
                      BASE_REACTION_X_undamped, BASE_REACTION_X_damped, BASE_REACTION_Y_undamped, BASE_REACTION_Y_damped):
    import matplotlib.pyplot as plt
    import numpy as np
    # Assuming you have 'time' array representing the time steps
    time = [i * dt for i in range(len(DISPLACEMENTS_undamped))]
    plt.figure(figsize=(14, 20))

    # Plot Displacements in X direction
    plt.subplot(6, 1, 1)
    P1 = DISP_X_undamped
    P2 = DISP_X_damped
    P11 = np.max(np.abs(P1))
    P22 = np.max(np.abs(P2))
    P3 = P11 / P22
    plt.plot(time, P1, color='black', label=f'Undamped X: {P11:.5e}')
    plt.plot(time, P2, color='red', label=f'Damped X: {P22:.5e}')
    plt.title(f'Displacement Time History (X direction) - Amplification Factor: {P3:.3f}')
    plt.xlabel('Time [s]')
    plt.ylabel('Displacement [m] (X direction)')
    plt.legend()

    # Plot Displacements in Y direction
    plt.subplot(6, 1, 2)
    P1 = DISP_Y_undamped
    P2 = DISP_Y_damped
    P11 = np.max(np.abs(P1))
    P22 = np.max(np.abs(P2))
    P3 = P11 / P22
    plt.plot(time, P1, color='black', label=f'Undamped Y: {P11:.5e}')
    plt.plot(time, P2, color='red', label=f'Damped Y: {P22:.5e}')
    plt.title(f'Displacement Time History (Y direction) - Amplification Factor: {P3:.3f}')
    plt.xlabel('Time [s]')
    plt.ylabel('Displacement [m] (Y direction)')
    plt.legend()

    # Plot Velocities
    plt.subplot(6, 1, 3)
    P1 = VELOCITY_undamped
    P2 = VELOCITY_damped
    P11 = np.max(np.abs(P1))
    P22 = np.max(np.abs(P2))
    P3 = P11 / P22
    plt.plot(time, P1, color='black', label=f'Undamped: {P11:.5e}')
    plt.plot(time, P2, color='red', label=f'Damped: {P22:.5e}')
    plt.title(f'Velocity Time History - Amplification Factor: {P3:.3f}')
    plt.xlabel('Time [s]')
    plt.ylabel('Velocity [m/s] (Y direction)')
    plt.legend()

    # Plot Accelerations
    plt.subplot(6, 1, 4)
    P1 = ACCELERATION_undamped
```

```
    P2 = ACCELERATION_damped
    P11 = np.max(np.abs(P1))
    P22 = np.max(np.abs(P2))
    P3 = P11 / P22
    plt.plot(time, P1, color='black', label=f'Undamped: {P11:.5e}')
    plt.plot(time, P2, color='red', label=f'Damped: {P22:.5e}')
    plt.title(f'Acceleration Time History - Amplification Factor: {P3:.3f}')
    plt.xlabel('Time [s]')
    plt.ylabel('Acceleration [m/s²] (Y direction)')
    plt.legend()

    # Plot Base Reactions in X direction
    plt.subplot(6, 1, 5)
    P1 = BASE_REACTION_X_undamped
    P2 = BASE_REACTION_X_damped
    P11 = np.max(np.abs(P1))
    P22 = np.max(np.abs(P2))
    P3 = P11 / P22
    plt.plot(time, P1, color='black', label=f'Undamped: {P11:.5e}')
    plt.plot(time, P2, color='red', label=f'Damped: {P22:.5e}')
    plt.title(f'Base Reaction Time History (X direction) - Amplification Factor: {P3:.3f}')
    plt.xlabel('Time [s]')
    plt.ylabel('Base Reaction [N] (X direction)')
    plt.legend()

    # Plot Base Reactions in Y direction
    plt.subplot(6, 1, 6)
    P1 = BASE_REACTION_Y_undamped
    P2 = BASE_REACTION_Y_damped
    P11 = np.max(np.abs(P1))
    P22 = np.max(np.abs(P2))
    P3 = P11 / P22
    plt.plot(time, P1, color='black', label=f'Undamped: {P11:.5e}')
    plt.plot(time, P2, color='red', label=f'Damped: {P22:.5e}')
    plt.title(f'Base Reaction Time History (Y direction) - Amplification Factor: {P3:.3f}')
    plt.xlabel('Time [s]')
    plt.ylabel('Base Reaction [N] (Y direction)')
    plt.legend()

    plt.tight_layout()
    plt.show()
```

DYNAMIC ANALYSIS

Parameters for the analysis

L = 50000.0 # [mm] Bridge span lentgh H1 = 3500.0 # [mm] Height of Top Cable H2 = 2000 # [mm] Height of Bottom Cable arc_depth = 1000.0 # [mm] E_cable = 210e5 # [N/mm^2] Modulus of elasticity Cable Cable_Dia_01 = 25 # [mm] Horizontal Longitudinal Cable Diameter Bottom Cable_Dia_02 = 18 # [mm] Horizontal Longitudinal Cable Diameter Top Cable_Dia_03 = 10 # [mm] Vertical Longitudinal Cable Diameter Top num_nodes = 51 # Cable Arc Number of nodes

TOTAL_MASS = 500000.0 # [kg] Total Mass of Structure damping_ratio = 0.05 # Damping ratio duration = 50.0 # [s] Duration of the analysis in seconds dt = 0.01 # Time step in seconds

MAX_ITERATIONS = 10000 # Maximum number of iterations TOLERANCE = 1.0e-14 # Tolerance for convergence

import time starttime = time.process_time()

Run the undamped analysis

damping = False LINEAR = True # False: Cable Nonlinear Materials Properties periodTF = False results_undamped = DYNAMIC_ANALYSIS(damping, damping_ratio, LINEAR, periodTF, duration, dt, TOTAL_MASS, L, H1, H2, arc_depth, E_cable, Cable_Dia_01, Cable_Dia_02, Cable_Dia_03, num_nodes, MAX_ITERATIONS, TOLERANCE) initial_coords, deformed_coords, DISPLACEMENTS_undamped, VELOCITY_undamped, ACCELERATION_undamped, DISP_X_undamped, DISP_Y_undamped, BASE_REACTION_X_undamped, BASE_REACTION_Y_undamped = results_undamped

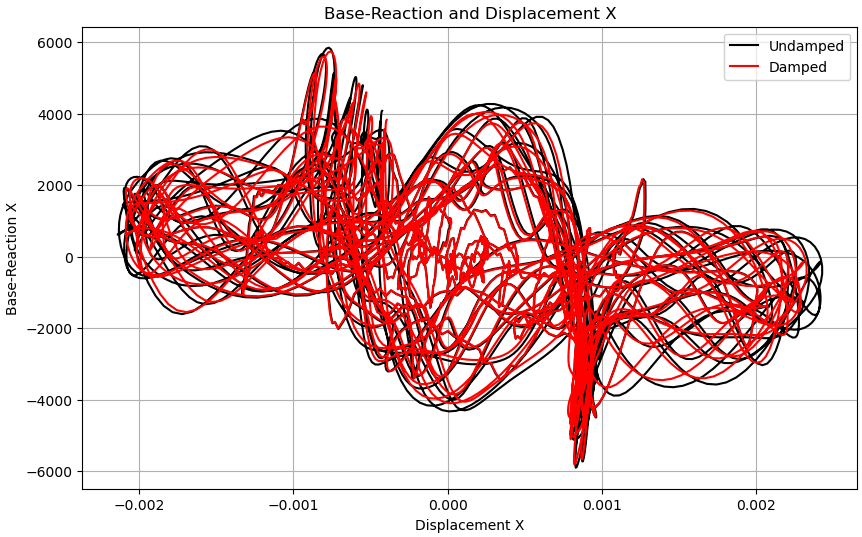
Run the damped analysis

damping = True LINEAR = True # False: Cable Nonlinear Materials Properties periodTF = False results_damped = DYNAMIC_ANALYSIS(damping, damping_ratio, LINEAR, periodTF, duration, dt, TOTAL_MASS, L, H1, H2, arc_depth, E_cable, Cable_Dia_01, Cable_Dia_02, Cable_Dia_03, num_nodes, MAX_ITERATIONS, TOLERANCE) initial_coords, deformed_coords, DISPLACEMENTS_damped, VELOCITY_damped, ACCELERATION_damped, DISP_X_damped, DISP_Y_damped, BASE_REACTION_X_damped, BASE_REACTION_Y_damped = results_damped

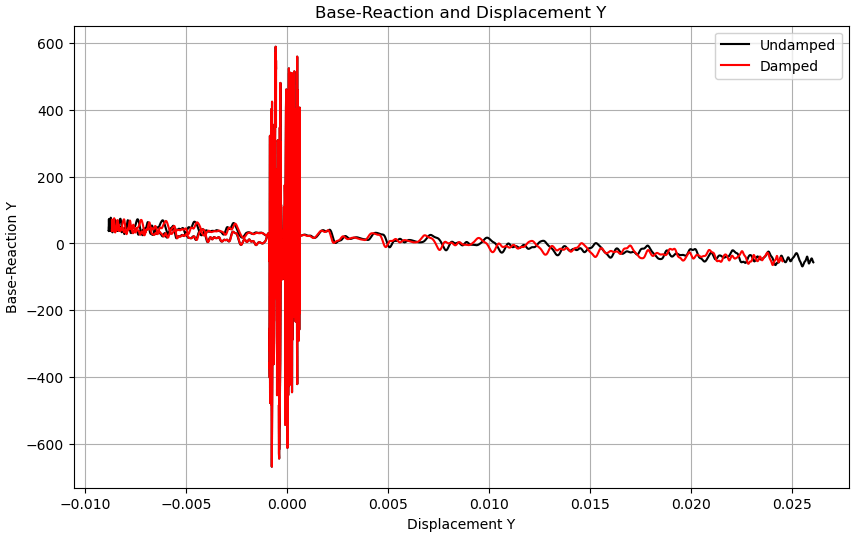
totaltime = time.process_time() - starttime print(f'\nTotal time (s): {totaltime:.4f} \n\n')

In [17]:
```
def PLOT_2D(X1, Y1, X2, Y2, XLABEL, YLABEL, TITLE):
    plt.figure(figsize=(10, 6))
    plt.plot(X1, Y1, label='Undamped', color='black')
    plt.plot(X2, Y2, label='Damped', color='red')
    plt.xlabel(XLABEL)
    plt.ylabel(YLABEL)
    plt.title(TITLE)
    plt.grid(True)
    #plt.semilogy()
    plt.legend()
    plt.show()
```

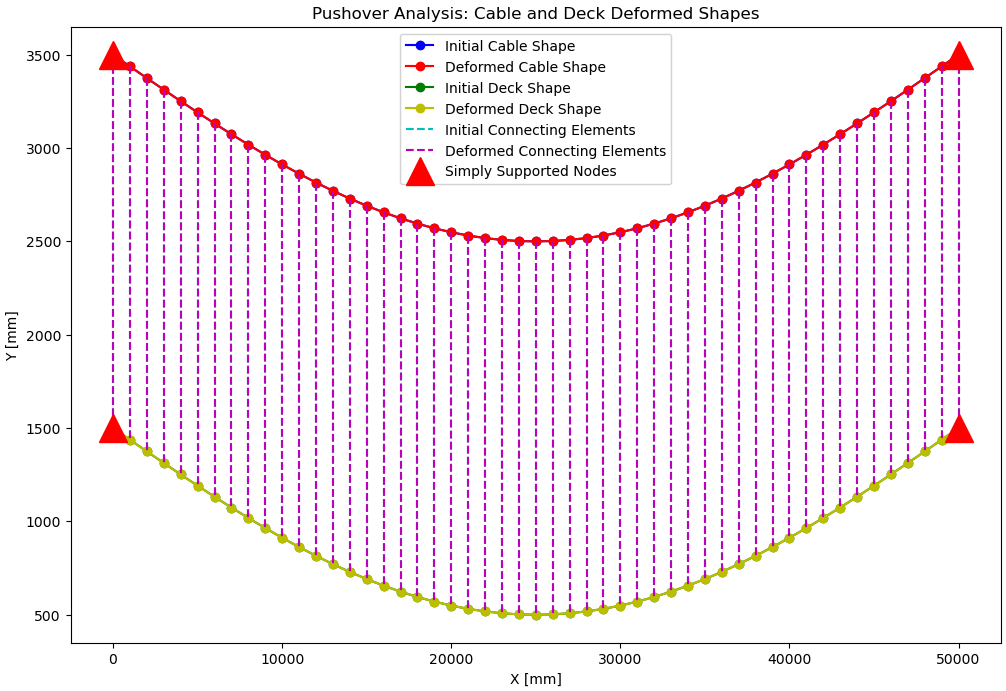
In [18]:
```
### BASE REACTION & DISPALCEMENT IN X:
X1 =  DISP_X_undamped
X2 =  DISP_X_damped
Y1 = BASE_REACTION_X_undamped
Y2 = BASE_REACTION_X_damped
XLABEL = 'Displacement X'
YLABEL = 'Base-Reaction X'
TITLE = 'Base-Reaction and Displacement X'

PLOT_2D(X1, Y1, X2, Y2, XLABEL, YLABEL, TITLE)
```



In [19]:
```
### BASE REACTION & DISPALCEMENT IN Y:
X1 =  DISP_Y_undamped
X2 =  DISP_Y_damped
Y1 = BASE_REACTION_Y_undamped
Y2 = BASE_REACTION_Y_damped
XLABEL = 'Displacement Y'
YLABEL = 'Base-Reaction Y'
TITLE = 'Base-Reaction and Displacement Y'

PLOT_2D(X1, Y1, X2, Y2, XLABEL, YLABEL, TITLE)
```



In [20]:
```
# Plot damped results
plot_shapes(initial_coords, DISPLACEMENTS_damped)
```



Plotting a histogram for the results of dynamic analysis of a structure, such as displacements, accelerations, and support reactions, helps analyze the statistical distribution and trends in the data. The following insights can be drawn from these histograms:

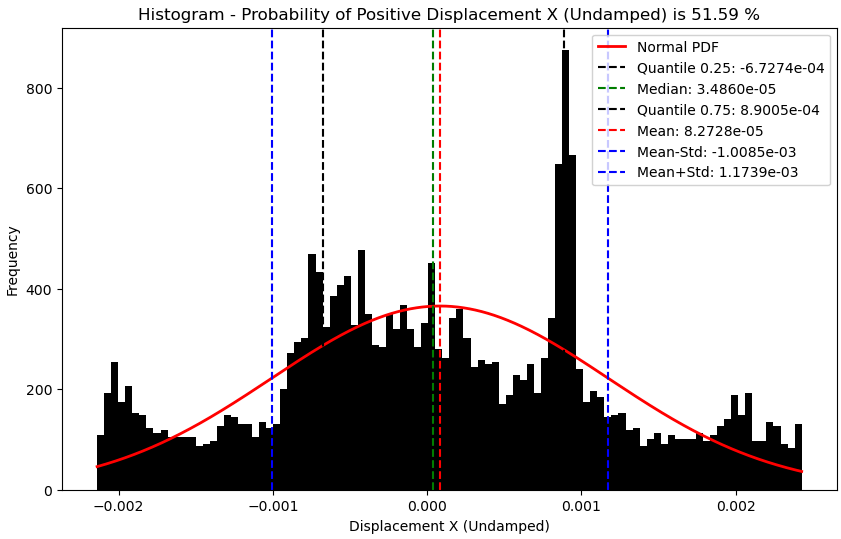
1. Range Identification

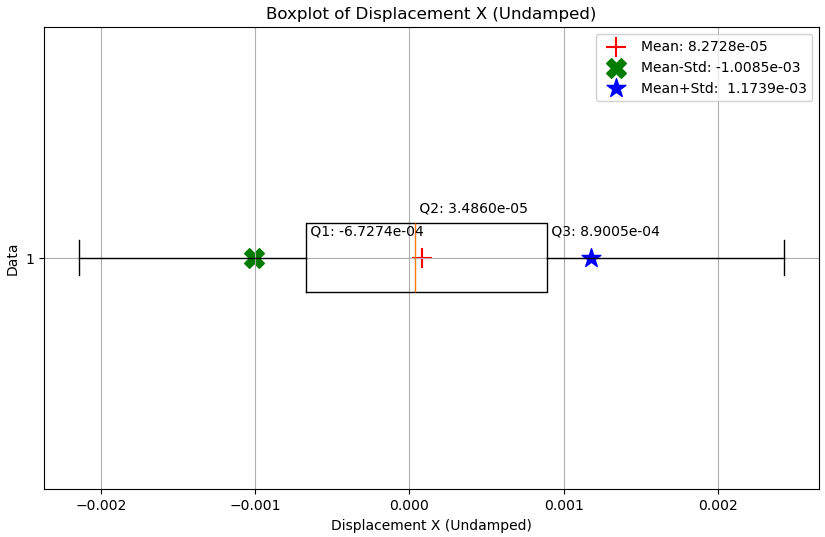
The histogram shows the range within which displacements, accelerations, or support reactions vary. This is crucial for assessing the stability of the structure and determining the allowable design limits. 2. Central Tendency From the histogram, measures like mean, median, or mode can be extracted. This helps identify the most likely values and provides an overview of the dominant behavior of the structure. 3. Data Dispersion The histogram illustrates the spread of the data (e.g., through its width and shape). A wide histogram may indicate nonlinear behavior or unusual effects in the structure. 4. Anomaly Detection Unexpected peaks or asymmetric distributions may indicate abnormal structural behavior (e.g., weaknesses or inelastic behavior). It can also reveal specific phenomena like resonances or amplification effects. 5. Probability of Specific Values The histogram's shape can indicate the probability distribution of the data (e.g., normal, uniform, or Poisson). This is useful for simulations and predicting future structural behavior. 6. Behavior Under Specific Conditions Analyzing histograms of acceleration or displacement can reveal dominant modes of the structure and their impact on overall response. These insights are helpful in designing vibration control systems (e.g., dampers). 7. Impact of External Forces or Loading An unusual histogram shape might suggest the influence of extraordinary dynamic loads (e.g., severe earthquakes or impact forces). Conclusion: Histograms are a simple yet effective tool for analyzing dynamic data. They help engineers and designers:

Understand the overall behavior of a structure. Assess the likelihood of extreme or unexpected responses. Define appropriate design or retrofitting criteria. For deeper analysis, histograms can be combined with other statistical tools, such as standard deviation or spectral analysis.

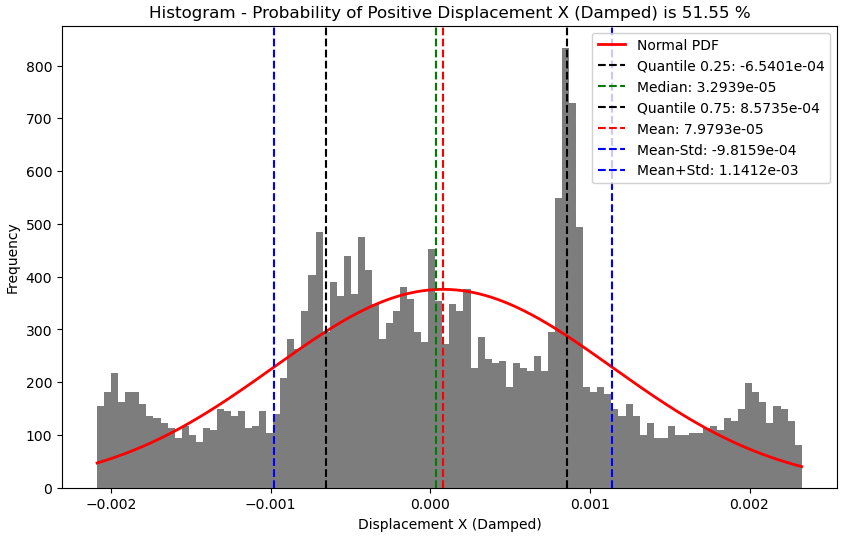
In [23]:
```
HISROGRAM_BOXPLOT(DISP_X_undamped, HISTO_COLOR='black', LABEL='Displacement X (Undamped)')
HISROGRAM_BOXPLOT(DISP_X_damped, HISTO_COLOR='grey', LABEL='Displacement X (Damped)')
#HISTOGRAM_BOXPLOT_PLOTLY(DISP_X_undamped, XLABEL='Displacement X (Undamped)', TITLE='Displacement X (Undamped)', COLOR='black')
#HISTOGRAM_BOXPLOT_PLOTLY(DISP_X_damped, XLABEL='Displacement X (Damped)', TITLE='Displacement X (Damped)', COLOR='grey')
```

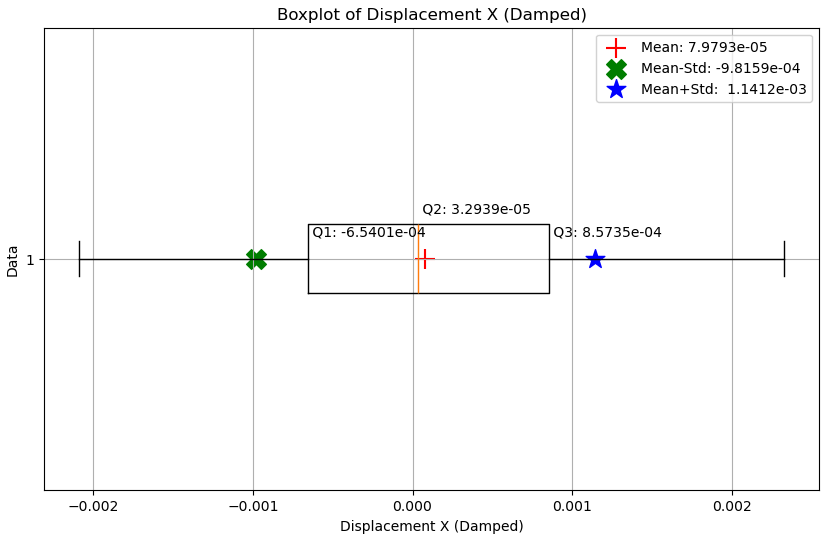
```
-------------------------
Box-Chart Datas:
Minimum: -2.1394e-03
First quartile: -6.7274e-04
Median: 3.4860e-05
Mean: 8.2728e-05
Std: 1.0912e-03
Third quartile: 8.9005e-04
Maximum: 2.4275e-03
Skewness: 3.5074e-02
kurtosis: -5.9403e-01
90% Confidence Interval: (-1.8625e-03, 2.0207e-03)
-------------------------
```






```
-------------------------
Box-Chart Datas:
Minimum: -2.0867e-03
First quartile: -6.5401e-04
Median: 3.2939e-05
Mean: 7.9793e-05
Std: 1.0614e-03
Third quartile: 8.5735e-04
Maximum: 2.3266e-03
Skewness: 2.9756e-02
kurtosis: -6.0405e-01
90% Confidence Interval: (-1.8069e-03, 1.9820e-03)
-------------------------
```

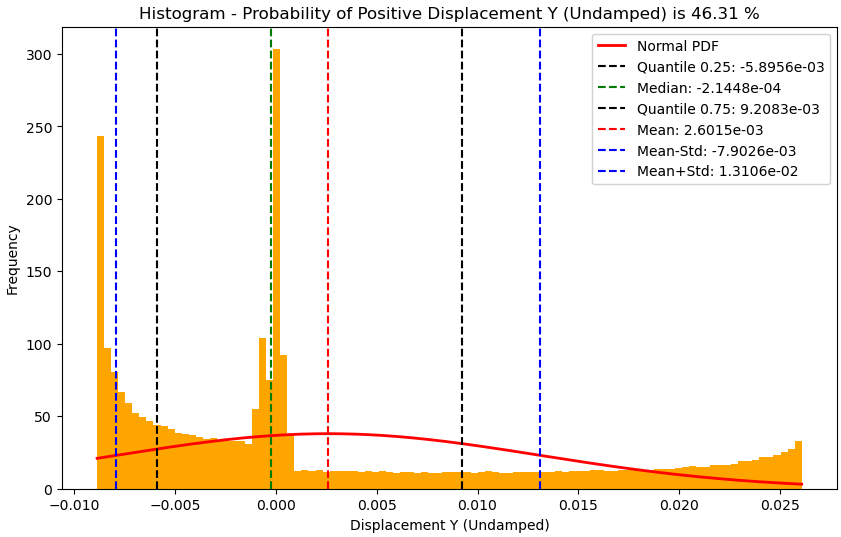


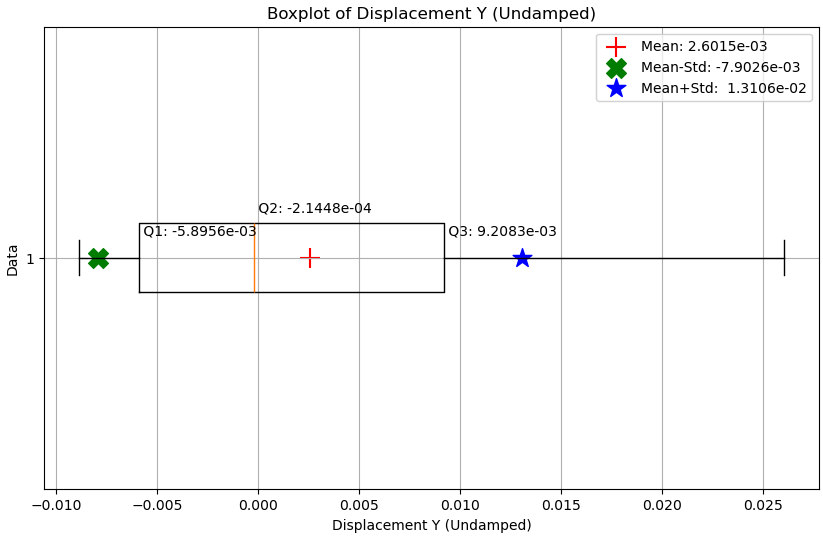


In [24]:
```
HISROGRAM_BOXPLOT(DISP_Y_undamped, HISTO_COLOR='orange', LABEL='Displacement Y (Undamped)')
HISROGRAM_BOXPLOT(DISP_Y_damped, HISTO_COLOR='brown', LABEL='Displacement Y (Damped)')
```

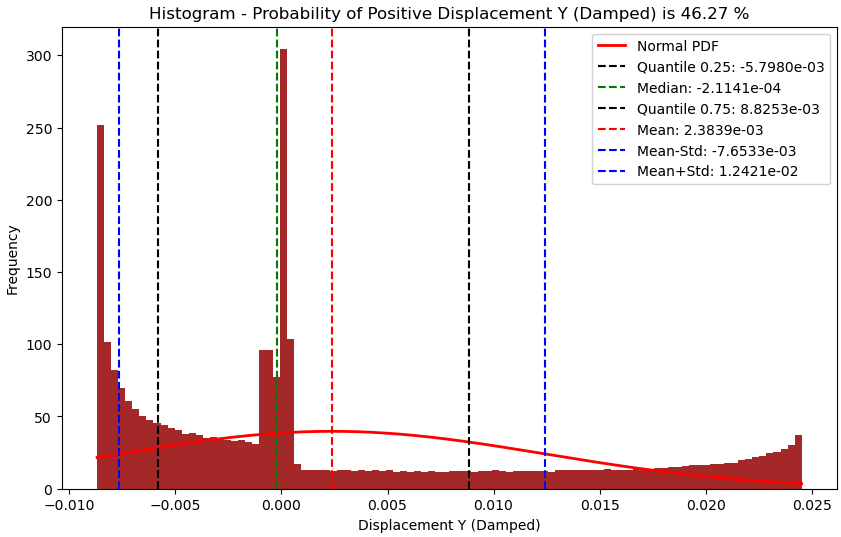
```
#HISTOGRAM_BOXPLOT_PLOTLY(DISP_Y_undamped, XLABEL='Displacement Y (Undamped)', TITLE='Displacement Y (Undamped)', COLOR='orange')
#HISTOGRAM_BOXPLOT_PLOTLY(DISP_Y_damped, XLABEL='Displacement Y (Damped)', TITLE='Displacement Y (Damped)', COLOR='brown')
```

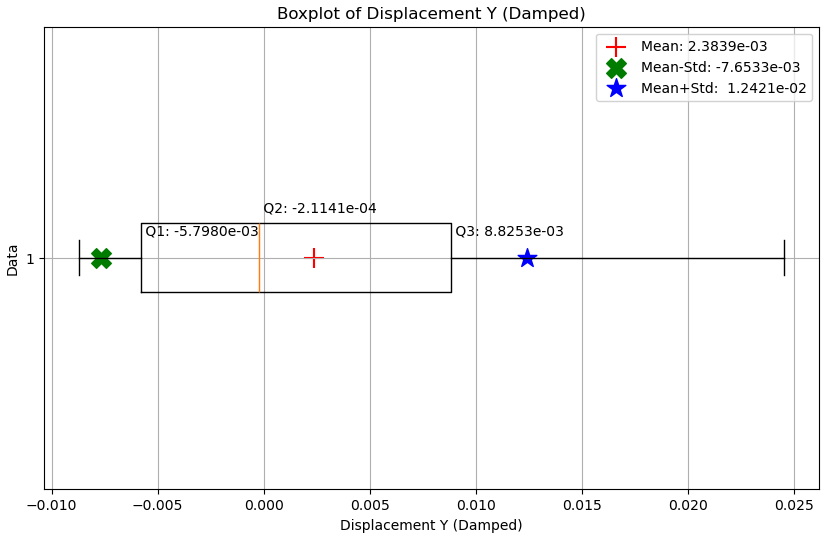
```
-------------------------
Box-Chart Datas:
Minimum: -8.8475e-03
First quartile: -5.8956e-03
Median: -2.1448e-04
Mean: 2.6015e-03
Std: 1.0504e-02
Third quartile: 9.2083e-03
Maximum: 2.6058e-02
Skewness: 9.0460e-01
kurtosis: -4.5237e-01
90% Confidence Interval: (-8.7370e-03, 2.4118e-02)
-------------------------
```





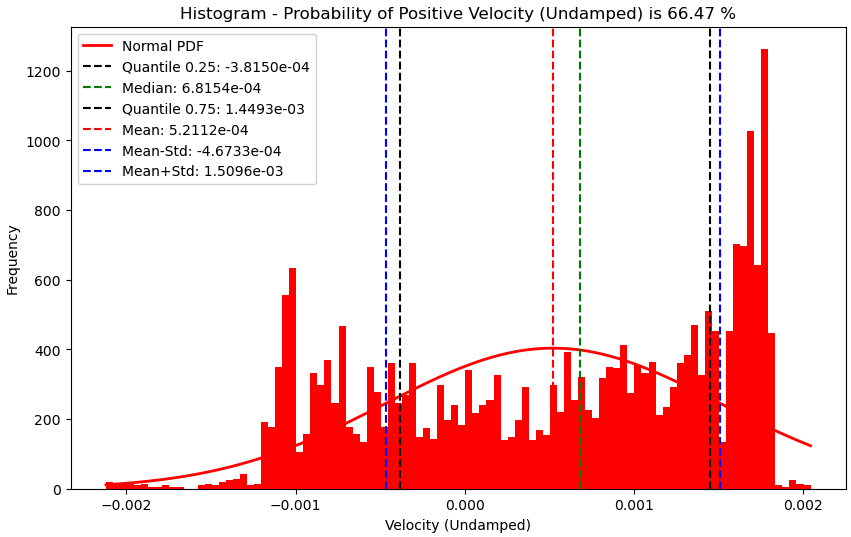

```
-------------------------
Box-Chart Datas:
Minimum: -8.6910e-03
First quartile: -5.7980e-03
Median: -2.1141e-04
Mean: 2.3839e-03
Std: 1.0037e-02
Third quartile: 8.8253e-03
Maximum: 2.4528e-02
Skewness: 8.7794e-01
kurtosis: -4.9601e-01
90% Confidence Interval: (-8.5794e-03, 2.2784e-02)
-------------------------
```

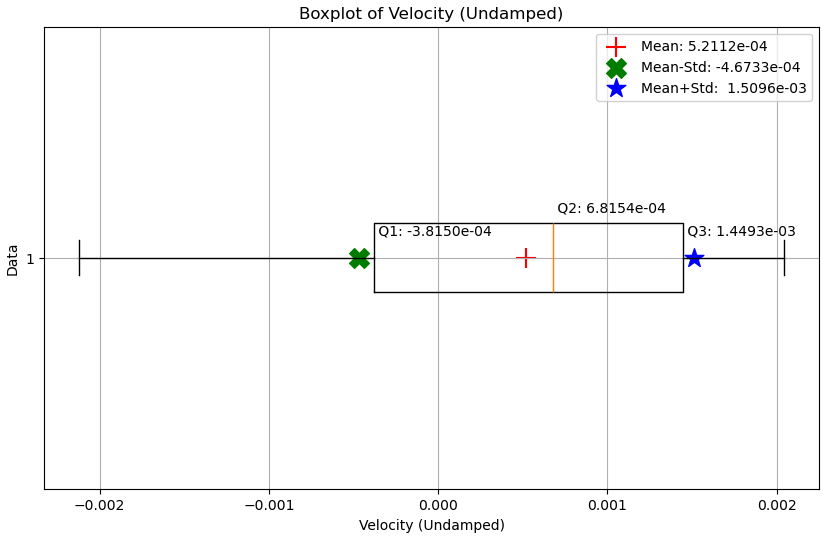




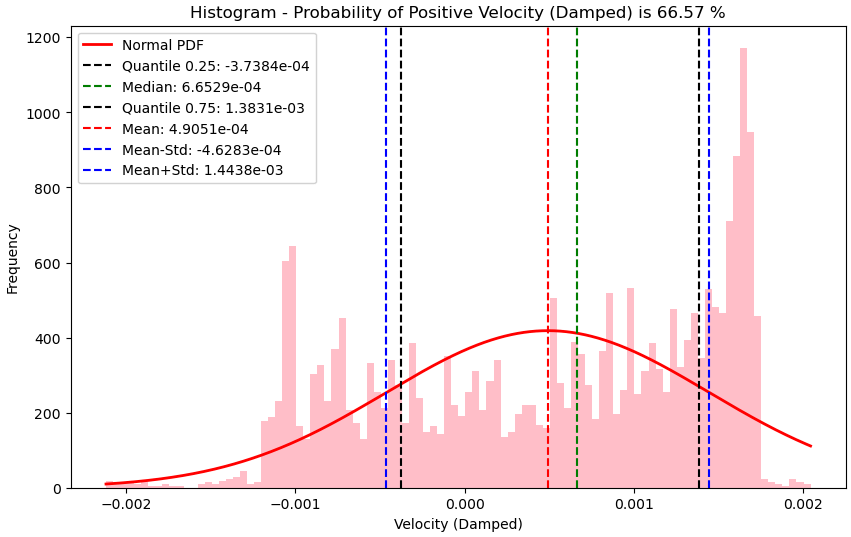
In [25]:
```
HISROGRAM_BOXPLOT(VELOCITY_undamped, HISTO_COLOR='red', LABEL='Velocity (Undamped)')
HISROGRAM_BOXPLOT(VELOCITY_damped, HISTO_COLOR='pink', LABEL='Velocity (Damped)')
#HISTOGRAM_BOXPLOT_PLOTLY(VELOCITY_undamped, XLABEL='Velocity (Undamped)', TITLE='Velocity (Undamped)', COLOR='red')
#HISTOGRAM_BOXPLOT_PLOTLY(VELOCITY_damped, XLABEL='Velocity (Damped)', TITLE='Velocity (Damped)', COLOR='pink')
```

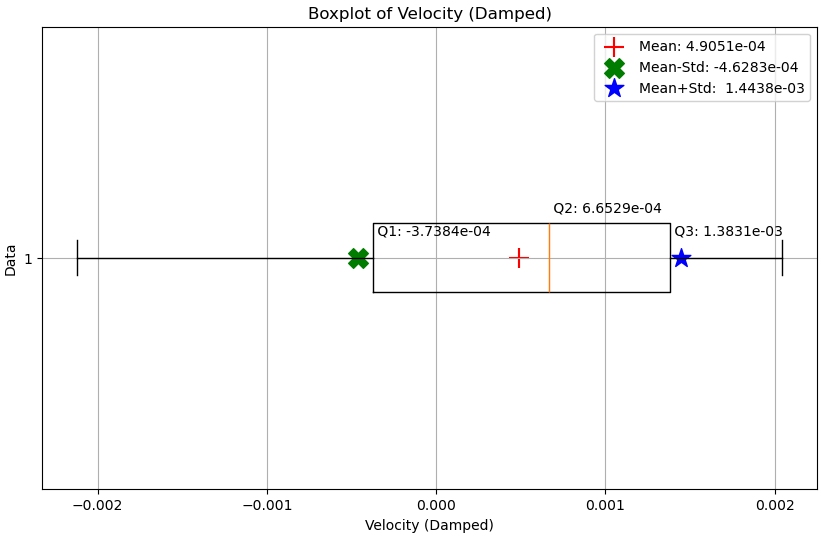
```
-------------------------
Box-Chart Datas:
Minimum: -2.1197e-03
First quartile: -3.8150e-04
Median: 6.8154e-04
Mean: 5.2112e-04
Std: 9.8845e-04
Third quartile: 1.4493e-03
Maximum: 2.0433e-03
Skewness: -3.2128e-01
kurtosis: -1.2248e+00
90% Confidence Interval: (-1.0664e-03, 1.7672e-03)
-------------------------
```





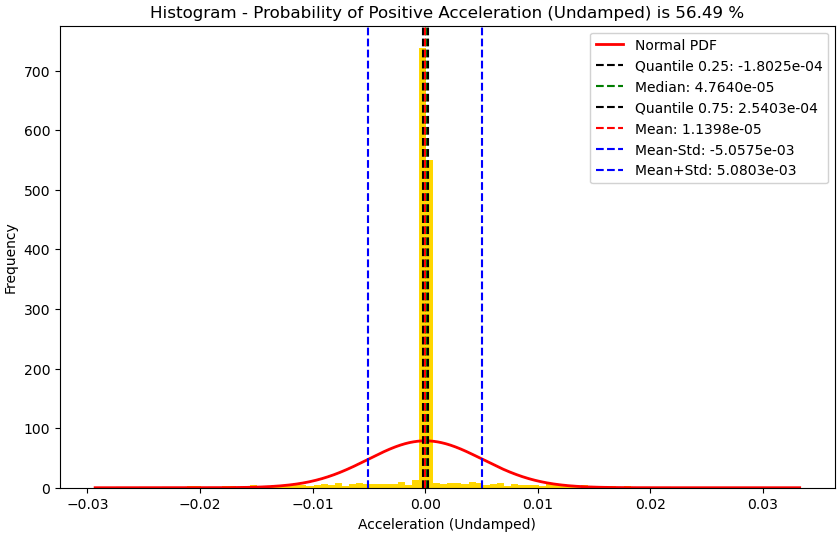

```
-------------------------
Box-Chart Datas:
Minimum: -2.1192e-03
First quartile: -3.7384e-04
Median: 6.6529e-04
Mean: 4.9051e-04
Std: 9.5334e-04
Third quartile: 1.3831e-03
Maximum: 2.0425e-03
Skewness: -3.4912e-01
kurtosis: -1.1814e+00
90% Confidence Interval: (-1.0584e-03, 1.6824e-03)
-------------------------
```

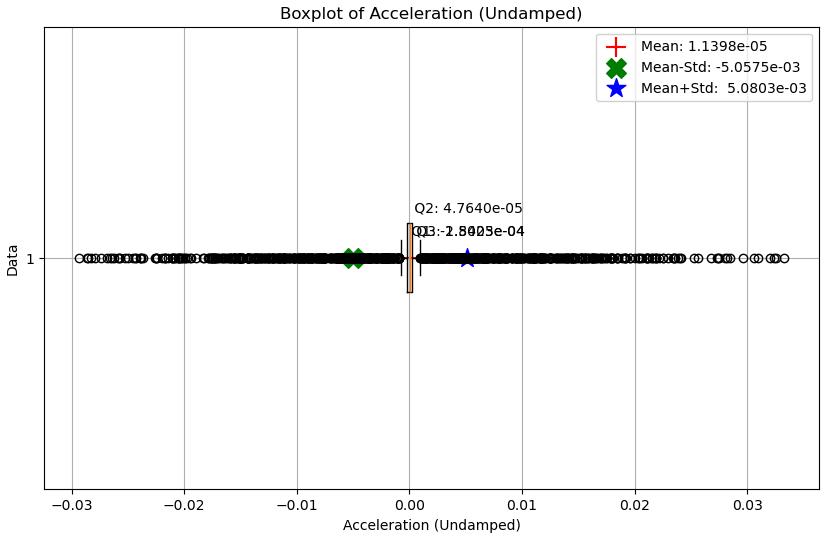




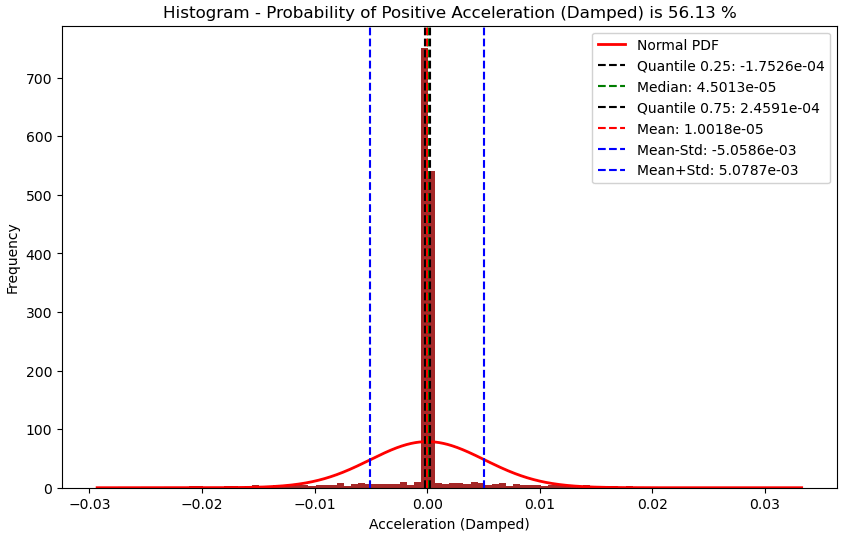
In [26]:
```
HISROGRAM_BOXPLOT(ACCELERATION_undamped, HISTO_COLOR='gold', LABEL='Acceleration (Undamped)')
HISROGRAM_BOXPLOT(ACCELERATION_damped, HISTO_COLOR='brown', LABEL='Acceleration (Damped)')
#HISTOGRAM_BOXPLOT_PLOTLY(ACCELERATION_undamped, XLABEL='Acceleration (Undamped)', TITLE='Acceleration (Undamped)', COLOR='gold')
#HISTOGRAM_BOXPLOT_PLOTLY(ACCELERATION_damped, XLABEL='Acceleration (Damped)', TITLE='Acceleration (Damped)', COLOR='brown')
```

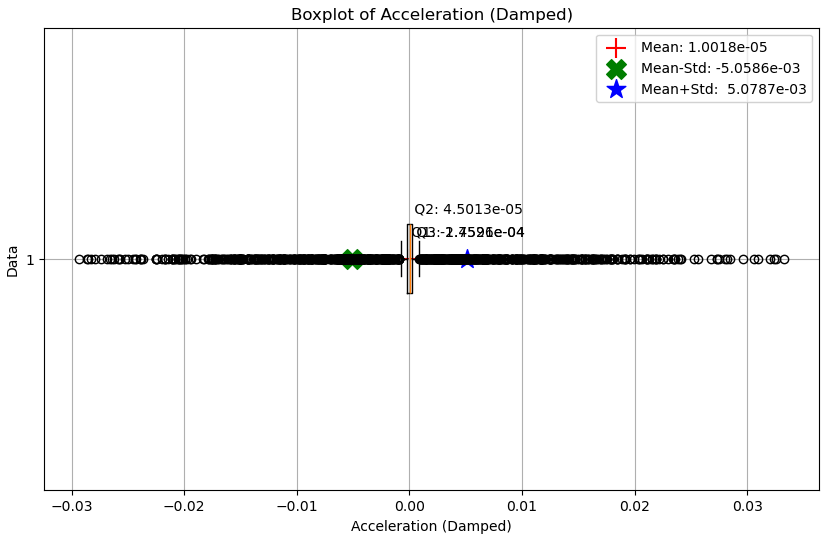
```
-------------------------
Box-Chart Datas:
Minimum: -2.9307e-02
First quartile: -1.8025e-04
Median: 4.7640e-05
Mean: 1.1398e-05
Std: 5.0689e-03
Third quartile: 2.5403e-04
Maximum: 3.3247e-02
Skewness: 8.0167e-02
kurtosis: 1.1514e+01
90% Confidence Interval: (-7.6501e-03, 6.9218e-03)
-------------------------
```






```
-------------------------
Box-Chart Datas:
Minimum: -2.9310e-02
First quartile: -1.7526e-04
Median: 4.5013e-05
Mean: 1.0018e-05
Std: 5.0686e-03
Third quartile: 2.4591e-04
Maximum: 3.3251e-02
Skewness: 8.1075e-02
kurtosis: 1.1518e+01
90% Confidence Interval: (-7.6482e-03, 6.9205e-03)
-------------------------
```

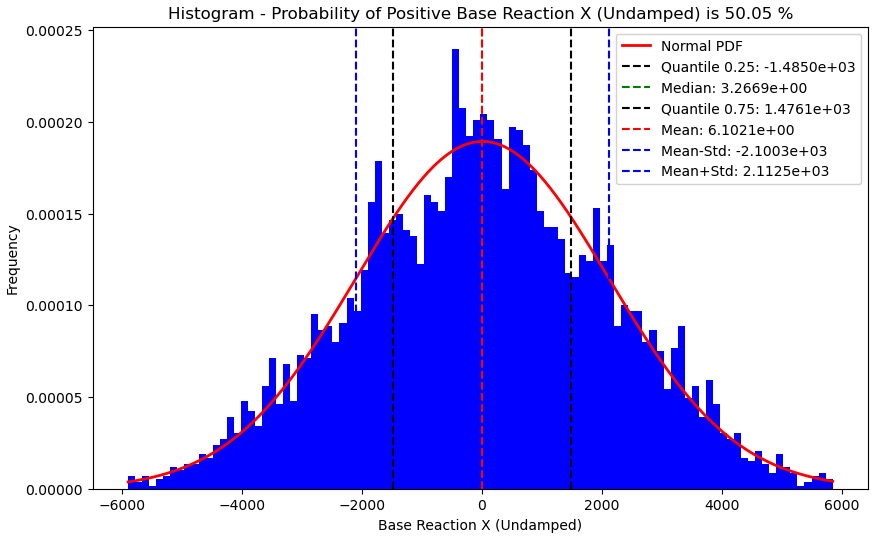


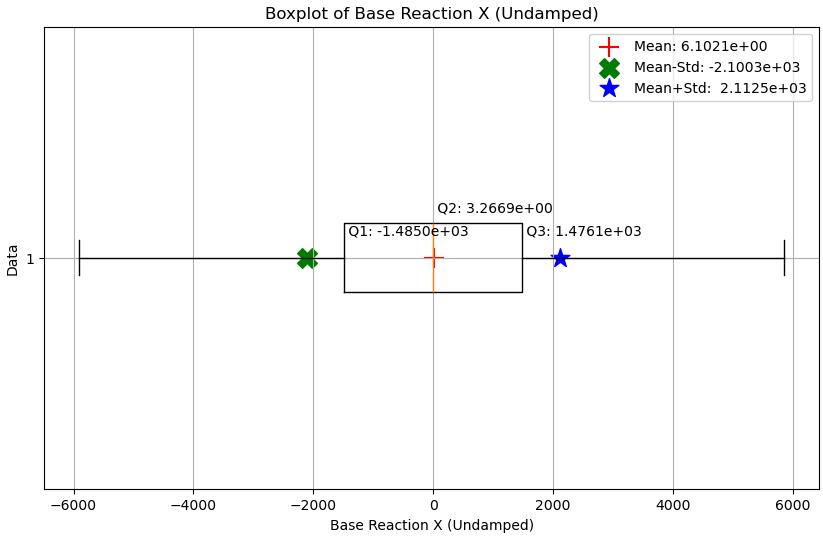


In [27]:
```
HISROGRAM_BOXPLOT(BASE_REACTION_X_undamped, HISTO_COLOR='blue', LABEL='Base Reaction X (Undamped)')
HISROGRAM_BOXPLOT(BASE_REACTION_X_damped, HISTO_COLOR='cyan', LABEL='Base Reaction X (Damped)')
```

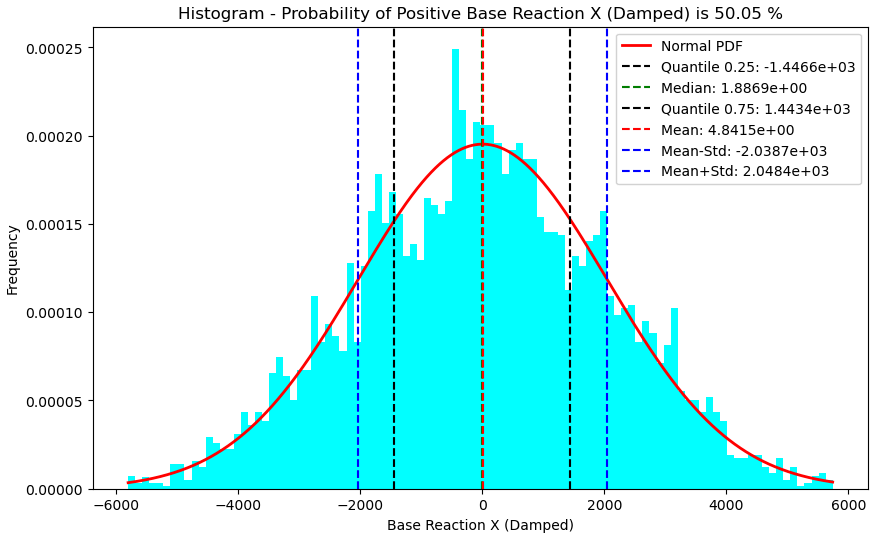
```
#HISTOGRAM_BOXPLOT_PLOTLY(BASE_REACTION_X_undamped, XLABEL='Base Reaction X (Undamped)', TITLE='Base Reaction X (Undamped)', COLOR='blue')
#HISTOGRAM_BOXPLOT_PLOTLY(BASE_REACTION_X_damped, XLABEL='Base Reaction X (Damped)', TITLE='Base Reaction X (Damped)', COLOR='cyan')
```

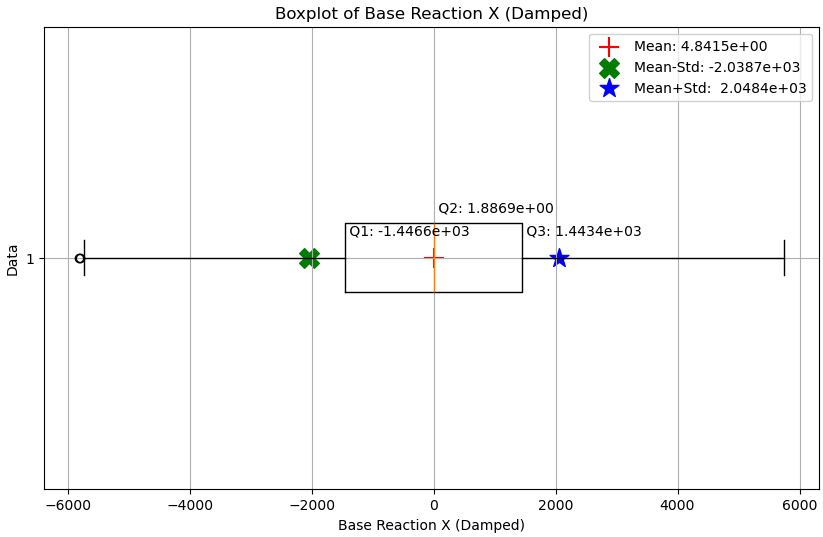
```
-------------------------
Box-Chart Datas:
Minimum: -5.9023e+03
First quartile: -1.4850e+03
Median: 3.2669e+00
Mean: 6.1021e+00
Std: 2.1064e+03
Third quartile: 1.4761e+03
Maximum: 5.8480e+03
Skewness: -1.0769e-02
kurtosis: -3.3829e-01
90% Confidence Interval: (-3.5393e+03, 3.4993e+03)
-------------------------
```





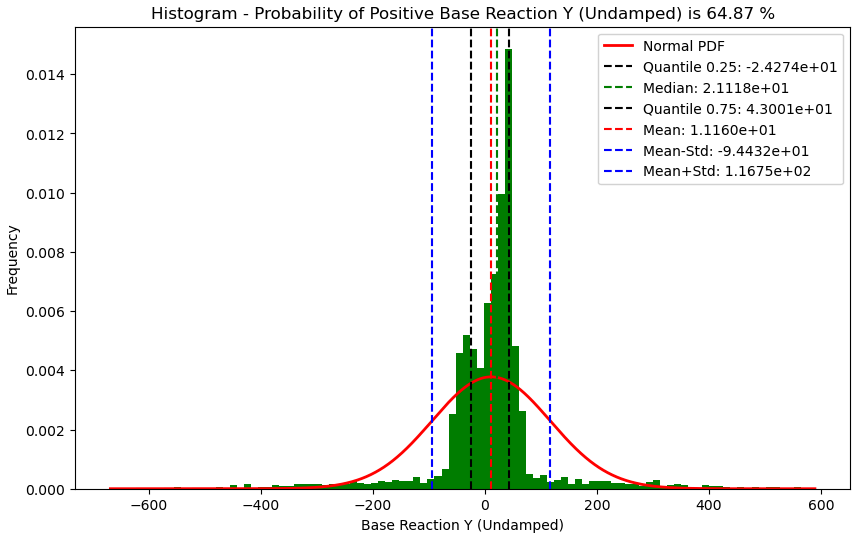

```
-------------------------
Box-Chart Datas:
Minimum: -5.8070e+03
First quartile: -1.4466e+03
Median: 1.8869e+00
Mean: 4.8415e+00
Std: 2.0436e+03
Third quartile: 1.4434e+03
Maximum: 5.7438e+03
Skewness: -1.2608e-02
kurtosis: -3.3154e-01
90% Confidence Interval: (-3.3994e+03, 3.3421e+03)
-------------------------
```

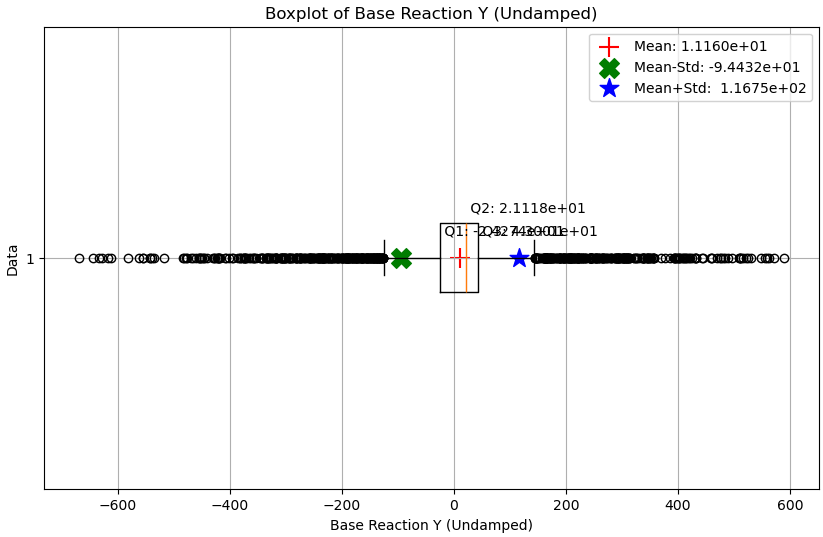




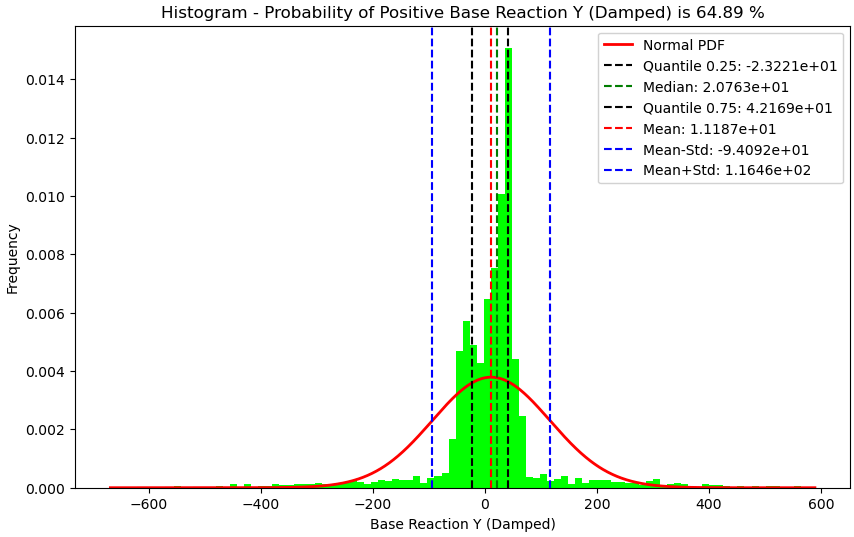
In [28]:
```
HISROGRAM_BOXPLOT(BASE_REACTION_Y_undamped, HISTO_COLOR='green', LABEL='Base Reaction Y (Undamped)')
HISROGRAM_BOXPLOT(BASE_REACTION_Y_damped, HISTO_COLOR='lime', LABEL='Base Reaction Y (Damped)')
#HISTOGRAM_BOXPLOT_PLOTLY(BASE_REACTION_Y_undamped, XLABEL='Base Reaction Y (Undamped)', TITLE='Base Reaction Y (Undamped)', COLOR='green')
#HISTOGRAM_BOXPLOT_PLOTLY(BASE_REACTION_Y_damped, XLABEL='Base Reaction Y (Damped)', TITLE='Base Reaction Y (Damped)', COLOR='lime')
```

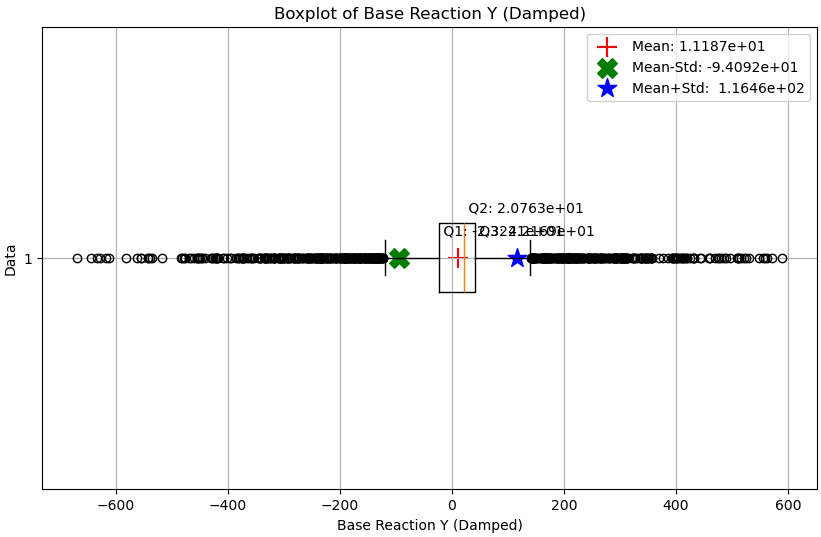
```
-------------------------
Box-Chart Datas:
Minimum: -6.6879e+02
First quartile: -2.4274e+01
Median: 2.1118e+01
Mean: 1.1160e+01
Std: 1.0559e+02
Third quartile: 4.3001e+01
Maximum: 5.8937e+02
Skewness: -3.7615e-01
kurtosis: 9.1616e+00
90% Confidence Interval: (-1.4260e+02, 1.5412e+02)
-------------------------
```






```
-------------------------
Box-Chart Datas:
Minimum: -6.6872e+02
First quartile: -2.3221e+01
Median: 2.0763e+01
Mean: 1.1187e+01
Std: 1.0528e+02
Third quartile: 4.2169e+01
Maximum: 5.8934e+02
Skewness: -3.7837e-01
kurtosis: 9.3045e+00
90% Confidence Interval: (-1.4266e+02, 1.5409e+02)
-------------------------
```




Boxplot of Base Reaction Y (Damped)
Mean: 1.1187e+01
Mean-Std: -9.4092e+01
Mean+Std: 1.1646e+02
Q2: 2.0763e+01
Data
1
−600
−400
−200
0
200
400
600
Base Reaction Y (Damped)

In [ ]: