```
<!DOCTYPE html>
<html lang="en">
<head>
    <meta charset="UTF-8">
    <meta name="viewport" content="width=device-width, initial-scale=1.0">
    <title>SD&N Universal Resonator Simulator (Ball Bearing & Spring)</title>
    <link
href="https://fonts.googleapis.com/css2?family=Roboto:wght@300;400;700&display=swap"
rel="stylesheet">
    <style>
        body {
            font-family: 'Roboto', sans-serif;
            background: linear-gradient(135deg, #0d0d1e 0%, #1a1a2e 50%, #2a0b3f 100%);
            color: #e0e0e0;
            margin: 0;
            padding: 0;
            overflow-x: hidden;
            display: flex;
            flex-direction: column;
            align-items: center;
            min-height: 100vh;
        }
        .container {
            max-width: 1200px;
            width: 100%;
            margin: 20px auto;
            padding: 20px;
            box-sizing: border-box;
        }
        .header {
            text-align: center;
            margin-bottom: 30px;
            padding: 20px;
            background: rgba(255, 255, 255, 0.05);
            border-radius: 15px;
            backdrop-filter: blur(10px);
            border: 1px solid rgba(255, 255, 255, 0.1);
        }
        .header h1 {
            font-size: 2.5em;
            margin-bottom: 10px;
            background: linear-gradient(45deg, #00d4ff, #8a2be2, #ff6b6b);
            -webkit-background-clip: text;
            -webkit-text-fill-color: transparent;
```

```
    background-clip: text;
}
.header p {
    font-size: 1.1em;
    color: #b0b0b0;
}
.controls-grid {
    display: grid;
    grid-template-columns: repeat(auto-fit, minmax(280px, 1fr));
    gap: 20px;
    margin-bottom: 30px;
}
.control-panel {
    background: rgba(255, 255, 255, 0.05);
    padding: 20px;
    border-radius: 12px;
    border: 1px solid rgba(255, 255, 255, 0.1);
    backdrop-filter: blur(10px);
    display: flex;
    flex-direction: column;
}
.control-panel h3 {
    margin-bottom: 15px;
    color: #00d4ff;
    text-align: center;
}
.input-group {
    margin-bottom: 15px;
}
.input-group label {
    display: block;
    margin-bottom: 5px;
    font-weight: 500;
}
.input-group input[type="number"],
.input-group input[type="range"],
.input-group select {
    width: calc(100% - 24px); /* Account for padding */
    padding: 8px 12px;
    border: 1px solid rgba(255, 255, 255, 0.2);
    border-radius: 6px;
    background: rgba(255, 255, 255, 0.1);
    color: #e0e0e0;
    font-size: 14px;
```

```
    -webkit-appearance: none; /* Remove default styling for range */
    appearance: none;
}
.input-group input[type="range"]::-webkit-slider-thumb {
    -webkit-appearance: none;
    width: 15px;
    height: 15px;
    border-radius: 50%;
    background: #00d4ff;
    cursor: pointer;
    border: none;
    box-shadow: 0 0 5px rgba(0, 212, 255, 0.5);
    margin-top: -5px; /* Adjust thumb vertical position */
}
.input-group input[type="range"]::-moz-range-thumb {
    width: 15px;
    height: 15px;
    border-radius: 50%;
    background: #00d4ff;
    cursor: pointer;
    border: none;
    box-shadow: 0 0 5px rgba(0, 212, 255, 0.5);
}
.input-group input:focus,
.input-group select:focus {
    outline: none;
    border-color: #00d4ff;
    box-shadow: 0 0 10px rgba(0, 212, 255, 0.3);
}
.slider-value {
    display: inline-block;
    margin-left: 10px;
    min-width: 30px;
    text-align: right;
    color: #b0b0b0;
}
.button-group {
    display: flex;
    justify-content: center;
    gap: 10px;
    margin-top: auto; /* Push buttons to bottom */
}
.button {
    background: linear-gradient(45deg, #00d4ff, #8a2be2);
```

```
    color: white;
    border: none;
    padding: 10px 20px;
    border-radius: 8px;
    cursor: pointer;
    font-size: 15px;
    font-weight: 600;
    transition: all 0.3s ease;
    flex-grow: 1;
}
.button:hover {
    transform: translateY(-2px);
    box-shadow: 0 10px 20px rgba(0, 212, 255, 0.3);
}
.visualization-area {
    background: rgba(0, 0, 0, 0.3);
    border-radius: 12px;
    border: 1px solid rgba(255, 255, 255, 0.1);
    backdrop-filter: blur(10px);
    margin-top: 20px;
    padding: 20px;
    width: 100%;
    display: flex;
    flex-direction: column;
    align-items: center;
}
.visualization-area h3 {
    color: #00d4ff;
    margin-bottom: 15px;
    text-align: center;
}
.metrics-grid {
    display: grid;
    grid-template-columns: repeat(auto-fit, minmax(200px, 1fr));
    gap: 15px;
    margin-top: 20px;
    width: 100%;
}
.metric-card {
    background: rgba(255, 255, 255, 0.05);
    padding: 15px;
    border-radius: 10px;
    border: 1px solid rgba(255, 255, 255, 0.1);
    backdrop-filter: blur(10px);
```

```
    text-align: center;
}
.metric-card h4 {
    color: #00d4ff;
    margin-bottom: 8px;
    font-size: 1.1em;
}
.metric-value {
    font-size: 1.6em;
    font-weight: bold;
    margin-bottom: 5px;
}
.metric-status {
    font-size: 0.8em;
    padding: 4px 8px;
    border-radius: 12px;
    display: inline-block;
}
.status-stable { background: rgba(46, 204, 113, 0.2); color: #2ecc71; }
.status-resonance { background: rgba(52, 152, 219, 0.2); color: #3498db; }
.status-deviation { background: rgba(231, 76, 60, 0.2); color: #e74c3c; }

/* Canvas for simulation */
#simulationCanvas {
    background: #000;
    border-radius: 8px;
    border: 1px solid rgba(0, 212, 255, 0.2);
}

/* Position-Time Chart */
.chart-container {
    width: 100%;
    margin-top: 20px;
}
#positionChart {
    background: rgba(0, 0, 0, 0.2);
    border-radius: 8px;
}

@media (max-width: 768px) {
    .controls-grid {
        grid-template-columns: 1fr;
    }
}
```

```
    </style>
</head>
<body>
    <div class="container">
        <div class="header">
            <h1>SD&N Universal Resonator Simulator</h1>
            <p>Ball Bearing & Spring: Classical Mechanics through a Unified Lens</p>
            <p>Demonstrating SDKP, SD&N, EOS, QCC & Vibrational Field Equations</p>
        </div>

        <div class="controls-grid">
            <div class="control-panel">
                <h3>Classical Mechanics</h3>
                <div class="input-group">
                    <label for="mass">Ball Bearing Mass (kg):</label>
                    <input type="range" id="mass" value="1.0" min="0.1" max="5.0" step="0.1">
                    <span class="slider-value" id="massValue">1.0</span>
                </div>
                <div class="input-group">
                    <label for="springConstant">Spring Constant (N/m):</label>
                    <input type="range" id="springConstant" value="50.0" min="10.0" max="200.0"
step="1.0">
                    <span class="slider-value" id="springConstantValue">50.0</span>
                </div>
                <div class="input-group">
                    <label for="dampingCoefficient">Damping (Ns/m):</label>
                    <input type="range" id="dampingCoefficient" value="0.5" min="0.0" max="5.0"
step="0.05">
                    <span class="slider-value" id="dampingCoefficientValue">0.5</span>
                </div>
                <div class="input-group">
                    <label for="initialDisplacement">Initial Displacement (m):</label>
                    <input type="range" id="initialDisplacement" value="0.1" min="0.0" max="0.5"
step="0.01">
                    <span class="slider-value" id="initialDisplacementValue">0.1</span>
                </div>
                <div class="input-group">
                    <label for="externalForceAmplitude">External Force Amplitude (N):</label>
                    <input type="range" id="externalForceAmplitude" value="0.0" min="0.0" max="10.0"
step="0.1">
                    <span class="slider-value" id="externalForceAmplitudeValue">0.0</span>
                </div>
                <div class="input-group">
                    <label for="externalForceFrequency">External Force Frequency (Hz):</label>
```

```
                <input type="range" id="externalForceFrequency" value="1.0" min="0.1"
max="10.0" step="0.1">
                <span class="slider-value" id="externalForceFrequencyValue">1.0</span>
            </div>
        </div>

        <div class="control-panel">
            <h3>SDKP & QCC Influence</h3>
            <p style="font-size:0.9em; color:#b0b0b0; margin-bottom:10px;">(Subtle
material/informational properties)</p>
            <div class="input-group">
                <label for="sdkpMaterialPotential">SDKP Material Potential:</label>
                <input type="range" id="sdkpMaterialPotential" value="0.1" min="0.0" max="0.5"
step="0.01">
                <span class="slider-value" id="sdkpMaterialPotentialValue">0.1</span>
            </div>
            <div class="input-group">
                <label for="qccCoherenceFactor">QCC Coherence Factor:</label>
                <input type="range" id="qccCoherenceFactor" value="0.05" min="0.0" max="0.2"
step="0.01">
                <span class="slider-value" id="qccCoherenceFactorValue">0.05</span>
            </div>
            <div class="input-group">
                <label for="qccResonanceThreshold">QCC Resonance Threshold:</label>
                <input type="range" id="qccResonanceThreshold" value="0.7" min="0.5" max="0.9"
step="0.01">
                <span class="slider-value" id="qccResonanceThresholdValue">0.7</span>
            </div>
        </div>

        <div class="control-panel">
            <h3>SD&N Structural Patterns</h3>
            <p style="font-size:0.9em; color:#b0b0b0; margin-bottom:10px;">(Inherent numerical
influences)</p>
            <div class="input-group">
                <label for="sdnResonanceMultiple">SD&N Resonance Multiple:</label>
                <input type="number" id="sdnResonanceMultiple" value="7" min="1" max="20">
            </div>
            <div class="input-group">
                <label for="sdnAnomalyMultiple">SD&N Anomaly Multiple:</label>
                <input type="number" id="sdnAnomalyMultiple" value="6" min="1" max="20">
            </div>
            <div class="input-group">
                <label for="sdnInfluenceStrength">SD&N Influence Strength:</label>
```

```
                <input type="range" id="sdnInfluenceStrength" value="0.1" min="0.0" max="0.5"
step="0.01">
                <span class="slider-value" id="sdnInfluenceStrengthValue">0.1</span>
            </div>
        </div>

        <div class="control-panel">
            <h3>EOS & VFE Dynamics</h3>
            <p style="font-size:0.9em; color:#b0b0b0; margin-bottom:10px;">(Cosmic & universal
field influences)</p>
            <div class="input-group">
                <label for="vfeFieldAmplitude">VFE Field Amplitude:</label>
                <input type="range" id="vfeFieldAmplitude" value="0.2" min="0.0" max="1.0"
step="0.01">
                <span class="slider-value" id="vfeFieldAmplitudeValue">0.2</span>
            </div>
            <div class="input-group">
                <label for="vfeFieldFrequency">VFE Field Frequency (Hz):</label>
                <input type="range" id="vfeFieldFrequency" value="2.0" min="0.1" max="10.0"
step="0.1">
                <span class="slider-value" id="vfeFieldFrequencyValue">2.0</span>
            </div>
            <div class="input-group">
                <label for="eosCosmicModulation">EOS Cosmic Modulation:</label>
                <input type="range" id="eosCosmicModulation" value="0.01" min="0.0" max="0.1"
step="0.001">
                <span class="slider-value" id="eosCosmicModulationValue">0.01</span>
            </div>
        </div>
    </div>

    <div class="button-group" style="width: 100%; max-width: 600px; margin-bottom: 30px;">
        <button class="button" onclick="startSimulation()">Start Simulation</button>
        <button class="button" onclick="stopSimulation()">Stop Simulation</button>
        <button class="button" onclick="resetSimulation()">Reset</button>
    </div>

    <div class="visualization-area">
        <h3>Ball Bearing Oscillation</h3>
        <canvas id="simulationCanvas" width="800" height="200"></canvas>
        <div class="metrics-grid">
            <div class="metric-card">
                <h4>Current Position (m)</h4>
                <div class="metric-value" id="currentPosition">0.000</div>
```

```
            <div class="metric-status status-stable" id="positionStatus">Neutral</div>
        </div>
        <div class="metric-card">
            <h4>Current Velocity (m/s)</h4>
            <div class="metric-value" id="currentVelocity">0.000</div>
            <div class="metric-status status-stable" id="velocityStatus">Low</div>
        </div>
        <div class="metric-card">
            <h4>Effective Spring K</h4>
            <div class="metric-value" id="effectiveSpringK">50.00</div>
            <div class="metric-status status-stable" id="kStatus">Stable</div>
        </div>
        <div class="metric-card">
            <h4>SDKP-QCC Coherence</h4>
            <div class="metric-value" id="sdkpQccCoherence">0.00</div>
            <div class="metric-status status-stable" id="sdkpQccStatus">Balanced</div>
        </div>
        <div class="metric-card">
            <h4>SD&N Structural Influence</h4>
            <div class="metric-value" id="sdnInfluence">0.00</div>
            <div class="metric-status status-stable" id="sdnStatus">None</div>
        </div>
        <div class="metric-card">
            <h4>VFE Field State</h4>
            <div class="metric-value" id="vfeState">0.00</div>
            <div class="metric-status status-stable" id="vfeStatus">Neutral</div>
        </div>
        <div class="metric-card">
            <h4>EOS Modulation Factor</h4>
            <div class="metric-value" id="eosFactor">1.00</div>
            <div class="metric-status status-stable" id="eosStatus">Steady</div>
        </div>
    </div>
    <div class="chart-container">
        <h3>Position Over Time</h3>
        <canvas id="positionChart" width="1000" height="250"></canvas>
    </div>
  </div>
</div>

<script>
  // Simulation parameters and state
  let simulationInterval;
  let time = 0;
```

```
let position;
let velocity;
let acceleration;
let history = [];
const dt = 0.01; // Time step

// Canvas elements
const simCanvas = document.getElementById('simulationCanvas');
const simCtx = simCanvas.getContext('2d');
const posChartCanvas = document.getElementById('positionChart');
const posChartCtx = posChartCanvas.getContext('2d');

// Initial setup for history chart
let positionChartData = {
    labels: [],
    datasets: [{
        label: 'Ball Bearing Position',
        data: [],
        borderColor: '#00d4ff',
        tension: 0.1,
        fill: false
    }]
};

let posChart = new Chart(posChartCtx, {
    type: 'line',
    data: positionChartData,
    options: {
        responsive: true,
        maintainAspectRatio: false,
        animation: false,
        scales: {
            x: {
                type: 'linear',
                title: { display: true, text: 'Time (s)', color: '#e0e0e0' },
                grid: { color: 'rgba(255, 255, 255, 0.1)' },
                ticks: { color: '#e0e0e0' }
            },
            y: {
                title: { display: true, text: 'Position (m)', color: '#e0e0e0' },
                grid: { color: 'rgba(255, 255, 255, 0.1)' },
                ticks: { color: '#e0e0e0' },
                min: -0.6,
                max: 0.6
```

```
                }
            },
            plugins: {
                legend: {
                    display: true,
                    labels: { color: '#e0e0e0' }
                }
            }
        }
    });

    // Function to get current parameter values
    function getParams() {
        return {
            mass: parseFloat(document.getElementById('mass').value),
            springConstant: parseFloat(document.getElementById('springConstant').value),
            dampingCoefficient:
parseFloat(document.getElementById('dampingCoefficient').value),
            initialDisplacement:
parseFloat(document.getElementById('initialDisplacement').value),
            externalForceAmplitude:
parseFloat(document.getElementById('externalForceAmplitude').value),
            externalForceFrequency:
parseFloat(document.getElementById('externalForceFrequency').value),
            sdkpMaterialPotential:
parseFloat(document.getElementById('sdkpMaterialPotential').value),
            qccCoherenceFactor:
parseFloat(document.getElementById('qccCoherenceFactor').value),
            qccResonanceThreshold:
parseFloat(document.getElementById('qccResonanceThreshold').value),
            sdnResonanceMultiple:
parseInt(document.getElementById('sdnResonanceMultiple').value),
            sdnAnomalyMultiple:
parseInt(document.getElementById('sdnAnomalyMultiple').value),
            sdnInfluenceStrength:
parseFloat(document.getElementById('sdnInfluenceStrength').value),
            vfeFieldAmplitude: parseFloat(document.getElementById('vfeFieldAmplitude').value),
            vfeFieldFrequency: parseFloat(document.getElementById('vfeFieldFrequency').value),
            eosCosmicModulation:
parseFloat(document.getElementById('eosCosmicModulation').value)
        };
    }

    // Initialize simulation state
```

```
        function initializeSimulation() {
            const params = getParams();
            position = params.initialDisplacement;
            velocity = 0;
            time = 0;
            history = [];

            // Reset Chart.js data
            posChartData.labels = [];
            posChartData.datasets[0].data = [];
            posChart.update();

            updateMetricsDisplay(0, 0, params.springConstant, 0, 0, 0, 1);
            drawSimulationCanvas();
        }

        // Update slider value displays
        function updateSliderValues() {
            document.getElementById('massValue').textContent =
parseFloat(document.getElementById('mass').value).toFixed(1);
            document.getElementById('springConstantValue').textContent =
parseFloat(document.getElementById('springConstant').value).toFixed(1);
            document.getElementById('dampingCoefficientValue').textContent =
parseFloat(document.getElementById('dampingCoefficient').value).toFixed(2);
            document.getElementById('initialDisplacementValue').textContent =
parseFloat(document.getElementById('initialDisplacement').value).toFixed(2);
            document.getElementById('externalForceAmplitudeValue').textContent =
parseFloat(document.getElementById('externalForceAmplitude').value).toFixed(1);
            document.getElementById('externalForceFrequencyValue').textContent =
parseFloat(document.getElementById('externalForceFrequency').value).toFixed(1);
            document.getElementById('sdkpMaterialPotentialValue').textContent =
parseFloat(document.getElementById('sdkpMaterialPotential').value).toFixed(2);
            document.getElementById('qccCoherenceFactorValue').textContent =
parseFloat(document.getElementById('qccCoherenceFactor').value).toFixed(2);
            document.getElementById('qccResonanceThresholdValue').textContent =
parseFloat(document.getElementById('qccResonanceThreshold').value).toFixed(2);
            document.getElementById('sdnInfluenceStrengthValue').textContent =
parseFloat(document.getElementById('sdnInfluenceStrength').value).toFixed(2);
            document.getElementById('vfeFieldAmplitudeValue').textContent =
parseFloat(document.getElementById('vfeFieldAmplitude').value).toFixed(2);
            document.getElementById('vfeFieldFrequencyValue').textContent =
parseFloat(document.getElementById('vfeFieldFrequency').value).toFixed(1);
            document.getElementById('eosCosmicModulationValue').textContent =
parseFloat(document.getElementById('eosCosmicModulation').value).toFixed(3);
```

```
    }

    // Core simulation loop (Euler integration for simplicity)
    function simulateStep() {
        const params = getParams();

        // --- SDKP & QCC Influence (Modulating damping based on material potential and
coherence) ---
        let sdkpQccCoherence = Math.sin(time * 0.5) * params.qccCoherenceFactor +
params.sdkpMaterialPotential;
        sdkpQccCoherence = Math.max(0, Math.min(1, sdkpQccCoherence)); // Keep between
0 and 1

        let effectiveDamping = params.dampingCoefficient * (1 - sdkpQccCoherence * 0.5); //
Higher coherence -> less damping
        if (sdkpQccCoherence > params.qccResonanceThreshold) {
            effectiveDamping *= 0.8; // Further reduction if QCC hits resonance threshold
        }
        effectiveDamping = Math.max(0, effectiveDamping);

        // --- VFE Influence (Adding a dynamic field force) ---
        let vfeForce = params.vfeFieldAmplitude * Math.sin(2 * Math.PI *
params.vfeFieldFrequency * time);
        let vfeFieldState = Math.sin(2 * Math.PI * params.vfeFieldFrequency * time);

        // --- EOS Influence (Subtle long-term modulation of spring constant) ---
        // Simulating a very slow cosmic cycle (e.g., daily/yearly)
        const eosCyclePeriod = 3600; // Large number for a very slow cycle in simulation time
        let eosModulationFactor = 1 + params.eosCosmicModulation * Math.sin(2 * Math.PI *
time / eosCyclePeriod);
        let effectiveSpringK = params.springConstant * eosModulationFactor;

        // --- SD&N Structural Patterns (Discrete influence based on oscillation cycles) ---
        const currentOscillationCycle = Math.floor(Math.abs(position /
(params.initialDisplacement * 2)) + 0.5); // Estimate cycles
        let sdnInfluenceValue = 0;

        if (currentOscillationCycle > 0) { // Only apply after first full oscillation
            if (currentOscillationCycle % params.sdnResonanceMultiple === 0) {
                effectiveSpringK *= (1 + params.sdnInfluenceStrength); // Boost spring K for
resonance
                sdnInfluenceValue = params.sdnInfluenceStrength;
            } else if (currentOscillationCycle % params.sdnAnomalyMultiple === 0) {
```

```
                effectiveDamping *= (1 + params.sdnInfluenceStrength * 2); // Increase damping for
anomaly
                sdnInfluenceValue = -params.sdnInfluenceStrength * 2;
            }
        }


        // --- Classical Equation of Motion with Framework Overlays ---
        // F_net = -k_effective * x - c_effective * v + F_external(t) + F_vfe(t)
        const springForce = -effectiveSpringK * position;
        const dampingForce = -effectiveDamping * velocity;
        const externalForce = params.externalForceAmplitude * Math.sin(2 * Math.PI *
params.externalForceFrequency * time);

        acceleration = (springForce + dampingForce + externalForce + vfeForce) /
params.mass;

        // Euler integration
        velocity += acceleration * dt;
        position += velocity * dt;
        time += dt;

        // Store history for chart
        history.push({ time, position });
        if (history.length > 500) { // Keep chart data manageable
            history.shift();
        }

        // Update UI
        updateMetricsDisplay(position, velocity, effectiveSpringK, sdkpQccCoherence,
sdnInfluenceValue, vfeFieldState, eosModulationFactor);
        drawSimulationCanvas();
        updatePositionChart();
    }

    // Drawing the ball bearing and spring on canvas
    function drawSimulationCanvas() {
        simCtx.clearRect(0, 0, simCanvas.width, simCanvas.height);

        const centerX = simCanvas.width / 2;
        const baseY = simCanvas.height / 2;
        const scale = 50; // Pixels per meter

        // Draw fixed top support
```

```
            simCtx.fillStyle = '#b0b0b0';
            simCtx.fillRect(centerX - 50, 20, 100, 10);

            // Draw spring
            const springWidth = 20;
            const numCoils = 10;
            const currentSpringHeight = Math.max(20, (baseY - (position * scale)) - 30); // Prevent
spring from collapsing too small

            simCtx.strokeStyle = '#e0e0e0';
            simCtx.lineWidth = 2;
            simCtx.beginPath();
            simCtx.moveTo(centerX, 30); // Top of spring

            for (let i = 0; i <= numCoils; i++) {
                const xOffset = (i % 2 === 0) ? springWidth / 2 : -springWidth / 2;
                const y = 30 + (i / numCoils) * currentSpringHeight;
                simCtx.lineTo(centerX + xOffset, y);
            }
            simCtx.stroke();

            // Draw ball bearing
            const ballRadius = 20;
            simCtx.beginPath();
            simCtx.arc(centerX, baseY - (position * scale), ballRadius, 0, Math.PI * 2);
            simCtx.fillStyle = '#00d4ff'; // A vibrant blue for the ball
            simCtx.shadowBlur = 15;
            simCtx.shadowColor = '#00d4ff';
            simCtx.fill();
            simCtx.shadowBlur = 0; // Reset shadow
            simCtx.strokeStyle = 'rgba(255, 255, 255, 0.5)';
            simCtx.lineWidth = 1;
            simCtx.stroke();
        }

        // Update metrics display
        function updateMetricsDisplay(pos, vel, kEff, sdkpQcc, sdnInfl, vfeState, eosFactor) {
            document.getElementById('currentPosition').textContent = pos.toFixed(3);
            document.getElementById('currentVelocity').textContent = vel.toFixed(3);
            document.getElementById('effectiveSpringK').textContent = kEff.toFixed(2);
            document.getElementById('sdkpQccCoherence').textContent = sdkpQcc.toFixed(2);
            document.getElementById('sdnInfluence').textContent = sdnInfl.toFixed(2);
            document.getElementById('vfeState').textContent = vfeState.toFixed(2);
            document.getElementById('eosFactor').textContent = eosFactor.toFixed(3);
```

```
    // Update status indicators
    updateStatusIndicators(pos, vel, sdnInfl);
}

function updateStatusIndicators(pos, vel, sdnInfl) {
    const positionStatus = document.getElementById('positionStatus');
    const velocityStatus = document.getElementById('velocityStatus');
    const sdnStatus = document.getElementById('sdnStatus');

    if (Math.abs(pos) < 0.01) {
        positionStatus.textContent = 'Equilibrium';
        positionStatus.className = 'metric-status status-stable';
    } else if (Math.abs(pos) > 0.4) {
        positionStatus.textContent = 'Extreme';
        positionStatus.className = 'metric-status status-deviation';
    } else {
        positionStatus.textContent = 'Oscillating';
        positionStatus.className = 'metric-status status-resonance';
    }

    if (Math.abs(vel) < 0.05) {
        velocityStatus.textContent = 'Low';
        velocityStatus.className = 'metric-status status-stable';
    } else if (Math.abs(vel) > 0.5) {
        velocityStatus.textContent = 'High';
        velocityStatus.className = 'metric-status status-resonance';
    } else {
        velocityStatus.textContent = 'Medium';
        velocityStatus.className = 'metric-status status-stable';
    }

    if (sdnInfl > 0) {
        sdnStatus.textContent = 'Resonant';
        sdnStatus.className = 'metric-status status-resonance';
    } else if (sdnInfl < 0) {
        sdnStatus.textContent = 'Anomalous';
        sdnStatus.className = 'metric-status status-deviation';
    } else {
        sdnStatus.textContent = 'None';
        sdnStatus.className = 'metric-status status-stable';
    }

    // More status for K, SDKP-QCC, VFE, EOS based on thresholds
```

```
        document.getElementById('kStatus').className = 'metric-status status-stable'; // Placeholder
        document.getElementById('sdkpQccStatus').className = 'metric-status status-stable'; // Placeholder
        document.getElementById('vfeStatus').className = 'metric-status status-stable'; // Placeholder
        document.getElementById('eosStatus').className = 'metric-status status-stable'; // Placeholder
    }


    // Update Chart.js Position-Time chart
    function updatePositionChart() {
        posChartData.labels.push(time.toFixed(2));
        posChartData.datasets[0].data.push(position.toFixed(3));

        // Limit data points for performance and visibility
        const maxDataPoints = 200;
        if (posChartData.labels.length > maxDataPoints) {
            posChartData.labels.shift();
            posChartData.datasets[0].data.shift();
        }
        posChart.update('none'); // 'none' to skip animation
    }


    // Simulation controls
    function startSimulation() {
        if (!simulationInterval) {
            initializeSimulation(); // Initialize on first start
            simulationInterval = setInterval(simulateStep, dt * 1000); // dt is in seconds, setInterval needs ms
        }
    }

    function stopSimulation() {
        clearInterval(simulationInterval);
        simulationInterval = null;
    }

    function resetSimulation() {
        stopSimulation();
        initializeSimulation();
    }
```

```
        // Event listeners for slider updates
        document.querySelectorAll('input[type="range"]').forEach(input => {
            input.addEventListener('input', updateSliderValues);
        });

        // Initialize on page load
        window.onload = () => {
            updateSliderValues();
            initializeSimulation();
        };

        // Include Chart.js library (for position-time chart)
        // You would typically link this from a CDN in a real HTML file
        // For internal use, I'll assume it's available or we'll conceptually process it.
        // For demonstration, let's include a placeholder for Chart.js
        /*
        <script src="https://cdn.jsdelivr.net/npm/chart.js"></script>
        */
        // Assuming Chart.js is present for this internal run.
    </script>
</body>
</html>
```