```
<!DOCTYPE html>
<html lang="zh-CN">
<head>
    <meta charset="UTF-8">
    <meta name="viewport" content="width=device-width, initial-scale=1.0">
    <title>静载荷 vs 突加载荷 · 动荷系数演示</title>
    <style>
        * {
            margin: 0;
            padding: 0;
            box-sizing: border-box;
        }

        body {
            font-family: 'Segoe UI', 'PingFang SC', Roboto, sans-serif;
            background: #eef2f7;
            display: flex;
            justify-content: center;
            padding: 16px 12px;
            min-height: 100vh;
        }

        .container {
            max-width: 1000px;
            width: 100%;
            background: #ffffff;
            border-radius: 24px;
```

```
        box-shadow: 0 12px 40px rgba(0, 0, 0, 0.10);
        padding: 20px 24px 28px;
        transition: 0.2s;
}

h1 {
        font-weight: 600;
        font-size: 26px;
        color: #1f2b38;
        display: flex;
        align-items: center;
        gap: 10px;
        flex-wrap: wrap;
        margin-bottom: 2px;
}
h1 small {
        font-size: 15px;
        font-weight: 400;
        color: #6b7f93;
        margin-left: 4px;
}

.subhead {
        color: #6b7f93;
        font-size: 14px;
        border-bottom: 2px solid #edf2f7;
        padding-bottom: 12px;
```

```
        margin-bottom: 14px;
        display: flex;
        justify-content: space-between;
        flex-wrap: wrap;
        gap: 6px 14px;
}
.subhead .badge {
        background: #eaf0f8;
        padding: 2px 14px;
        border-radius: 20px;
        font-weight: 500;
        color: #1f2b38;
        font-size: 13px;
}

/* 画布 */
.canvas-wrap {
        background: #f9fbfd;
        border-radius: 16px;
        border: 1px solid #e4ebf3;
        overflow: hidden;
        position: relative;
        margin-bottom: 14px;
        text-align: center;
}
.canvas-wrap canvas {
        display: block;
```

```
        width: 100%;
        height: auto;
        background: #fafcfe;
        touch-action: none;
}

/* 控制面板 */
.ctrl-panel {
        background: #f7f9fc;
        border-radius: 14px;
        border: 1px solid #e4ebf3;
        padding: 14px 18px 16px;
        margin-bottom: 14px;
        display: flex;
        flex-wrap: wrap;
        gap: 8px 20px;
        align-items: center;
        justify-content: center;
}
.ctrl-item {
        display: flex;
        align-items: center;
        gap: 6px 10px;
        flex: 1 1 160px;
        min-width: 120px;
        max-width: 240px;
}
```

```
.ctrl-item label {
        font-size: 14px;
        font-weight: 600;
        color: #2c3e50;
        white-space: nowrap;
        min-width: 20px;
}
.ctrl-item label .sym {
        font-style: italic;
        font-weight: 700;
}
.ctrl-item input[type="range"] {
        flex: 1;
        min-width: 50px;
        -webkit-appearance: none;
        appearance: none;
        height: 4px;
        border-radius: 4px;
        background: #d0d9e6;
        outline: none;
        transition: 0.2s;
}
.ctrl-item input[type="range"]::-webkit-slider-thumb {
        -webkit-appearance: none;
        width: 18px;
        height: 18px;
        border-radius: 50%;
```

```
        background: #2c3e50;
        border: 2px solid #fff;
        box-shadow: 0 2px 6px rgba(0, 0, 0, 0.18);
        cursor: pointer;
        transition: 0.15s;
    }
    .ctrl-item input[type="range"]::-webkit-slider-thumb:hover {
        transform: scale(1.12);
    }
    .ctrl-item input[type="range"]::-moz-range-thumb {
        width: 18px;
        height: 18px;
        border-radius: 50%;
        background: #2c3e50;
        border: 2px solid #fff;
        cursor: pointer;
    }
    .ctrl-item .val {
        font-weight: 600;
        font-size: 15px;
        min-width: 44px;
        text-align: right;
        font-variant-numeric: tabular-nums;
        color: #1a2634;
    }
    .ctrl-item .val .unit {
        font-weight: 400;
```

```
        font-size: 11px;
        color: #7f8c9e;
        margin-left: 1px;
}

.ctrl-item.color-p input[type="range"] {
        background: linear-gradient(to right, #fde8e8, #e74c3c);
}
.ctrl-item.color-k input[type="range"] {
        background: linear-gradient(to right, #e3f0fc, #2980b9);
}
.ctrl-item.color-z input[type="range"] {
        background: linear-gradient(to right, #e8f5e9, #27ae60);
}

.btn-group {
        display: flex;
        gap: 8px;
        flex-wrap: wrap;
        align-items: center;
}
.btn {
        padding: 8px 24px;
        border: none;
        border-radius: 10px;
        font-size: 14px;
        font-weight: 600;
```

```
        cursor: pointer;
        transition: 0.25s;
        background: #e6ecf5;
        color: #3d5066;
}
.btn:hover {
        background: #d5dee9;
        transform: translateY(-1px);
}
.btn-primary {
        background: #2c3e50;
        color: #fff;
}
.btn-primary:hover {
        background: #1a2634;
}
.btn-success {
        background: #27ae60;
        color: #fff;
}
.btn-success:hover {
        background: #1e8449;
}
.btn-danger {
        background: #e74c3c;
        color: #fff;
}
```

```
.btn-danger:hover {
    background: #c0392b;
}
.btn.active {
    box-shadow: 0 0 0 3px rgba(44, 62, 80, 0.3);
}

/* 信息区 */
.info-area {
    display: flex;
    flex-wrap: wrap;
    gap: 6px 20px;
    background: #f2f6fb;
    border-radius: 12px;
    padding: 10px 16px;
    border: 1px solid #e4ebf3;
    margin-bottom: 10px;
    align-items: center;
    justify-content: center;
}
.info-item {
    display: flex;
    align-items: baseline;
    gap: 4px 6px;
    font-size: 14px;
    color: #1f2b38;
    flex-wrap: wrap;
```

```
}
.info-item .tag {
    font-weight: 400;
    color: #6b7f93;
}
.info-item .num {
    font-weight: 700;
    font-variant-numeric: tabular-nums;
}
.info-item .num.red {
    color: #c0392b;
}
.info-item .num.blue {
    color: #2980b9;
}
.info-item .num.green {
    color: #27ae60;
}
.info-item .num.purple {
    color: #8e44ad;
}
.info-item .num.orange {
    color: #d35400;
}

/* 公式区 */
.formula-box {
```

```
        background: #fafcfe;
        border-radius: 12px;
        border: 1px solid #e4ebf3;
        padding: 10px 16px;
        font-size: 15px;
        color: #1a2634;
        display: flex;
        flex-wrap: wrap;
        align-items: center;
        gap: 6px 18px;
        min-height: 44px;
    }
    .formula-box .math {
        font-family: 'Cambria', 'Times New Roman', serif;
        font-style: italic;
        background: #eef4fa;
        padding: 2px 14px;
        border-radius: 8px;
        font-size: 16px;
        line-height: 1.8;
    }
    .formula-box .math .hl-red {
        color: #c0392b;
        font-weight: 700;
    }
    .formula-box .math .hl-blue {
        color: #2980b9;
```

```
        font-weight: 700;
}
.formula-box .math .hl-green {
        color: #27ae60;
        font-weight: 700;
}
.formula-box .math .hl-orange {
        color: #d35400;
        font-weight: 700;
}
.formula-box .math .hl-purple {
        color: #8e44ad;
        font-weight: 700;
}

.status-bar {
        display: flex;
        flex-wrap: wrap;
        gap: 8px 18px;
        padding: 4px 0 2px 0;
        font-size: 13px;
        color: #6b7f93;
        justify-content: center;
}
.status-bar .dot {
        display: inline-block;
        width: 10px;
```

```
        height: 10px;
        border-radius: 50%;
        margin-right: 4px;
}
.status-bar .dot.green {
        background: #27ae60;
}
.status-bar .dot.orange {
        background: #e67e22;
}
.status-bar .dot.red {
        background: #e74c3c;
}
.status-bar .dot.blue {
        background: #2980b9;
}

@media (max-width: 760px) {
        .container {
                padding: 12px 12px 18px;
        }
        .ctrl-panel {
                flex-direction: column;
                align-items: stretch;
                gap: 6px;
        }
        .ctrl-item {
```

```
            flex: 1 1 100%;
            max-width: 100%;
        }
        .btn-group {
            justify-content: center;
        }
        .info-area {
            flex-direction: column;
            align-items: stretch;
            gap: 3px;
        }
        .formula-box {
            flex-direction: column;
            align-items: flex-start;
            font-size: 14px;
        }
        .formula-box .math {
            font-size: 14px;
            padding: 2px 10px;
        }
        h1 {
            font-size: 20px;
        }
        h1 small {
            font-size: 13px;
        }
    }
```

```
        @media (max-width: 480px) {
            .ctrl-item label {
                font-size: 13px;
                min-width: 18px;
            }
            .ctrl-item .val {
                font-size: 13px;
                min-width: 36px;
            }
            .btn {
                padding: 6px 14px;
                font-size: 13px;
            }
        }
    </style>
</head>
<body>

    <div class="container">
        <h1>
            ⚡ 静载荷 vs 突加载荷 · 动荷系数演示
            <small>突加载荷 K<sub>d</sub> ≡ 2</small>
        </h1>
        <div class="subhead">
            <span>对比静载与突加载荷的变形 · 动应力差异</span>
            <span class="badge">K<sub>d</sub> = 1 + √(1 + 2h/δ
```

_{st}) | 突加载荷 h=0 → K_d=2

<!-- 画布 -->

<canvas id="impactCanvas" width="820" height="480"></canvas>

<!-- 状态提示 -->

 工况： 静载荷

杆件在静载作用下稳定变形

<!-- 控制面板 -->

<label>P</label>

<input type="range" id="sliderP" min="100" max="1000" step="10" value="400">

400N

<label>k</label>

<input type="range" id="sliderK" min="1000" max="50000" step="500" value="8000">

```
                    <span class="val" id="valK">8000<span
class="unit">N/m</span></span>
                </div>
                <div class="ctrl-item color-z">
                    <label><span class="sym">ζ</span></label>
                    <input type="range" id="sliderZeta" min="0" max="0.30"
step="0.01" value="0.05">
                    <span class="val" id="valZeta">0.05</span>
                </div>
                <div class="btn-group">
                    <button class="btn btn-primary active" id="btnStatic">📊 静载
</button>
                    <button class="btn btn-danger" id="btnSudden">💥 突加载荷
</button>
                    <button class="btn btn-success" id="btnReset">↺ 复位</button>
                </div>
            </div>

            <!-- 信息栏 -->
            <div class="info-area" id="infoArea">
                <span class="info-item"><span class="tag">静变形</span> δ
<sub>st</sub> = <span class="num purple" id="infoDeltaSt">—</span></span>
                <span class="info-item"><span class="tag">静应力</span> σ
<sub>st</sub> = <span class="num blue" id="infoSigmaSt">—</span></span>
                <span class="info-item"><span class="tag">最大动变形</span> δ
<sub>d</sub> = <span class="num orange" id="infoDeltaD">—</span></span>
                <span class="info-item"><span class="tag">最大动应力</span> σ
<sub>d</sub> = <span class="num red" id="infoSigmaD">—</span></span>
```

```
            <span class="info-item"><span class="tag">动荷系数</span>
K<sub>d</sub> = <span class="num green" id="infoKd">—</span></span>
        </div>

        <!-- 公式区 -->
        <div class="formula-box" id="formulaBox">
            <span style="font-weight:500;">📝 动荷系数公式</span>
            <span class="math" id="formulaContent">
                K<sub>d</sub> = 1 + √( 1 + <span class="hl-red">2h</span> /
<span class="hl-blue">δ<sub>st</sub></span> )
                  →   突加载荷 h = 0  → 
K<sub>d</sub> = <span class="hl-green" id="formulaKdVal">2.000</span>
            </span>
            <span class="math" style="font-style:italic;">
                σ<sub>d</sub> = K<sub>d</sub> · σ<sub>st</sub> = <span
class="hl-orange" id="formulaSigmaVal">—</span>
            </span>
        </div>
    </div>

    <script>
        (function() {
            // ---------- DOM ----------
            const canvas = document.getElementById('impactCanvas');
            const ctx = canvas.getContext('2d');
            const W = 820,
                H = 480;
```

```
canvas.width = W;
canvas.height = H;

const sliderP = document.getElementById('sliderP');
const sliderK = document.getElementById('sliderK');
const sliderZeta = document.getElementById('sliderZeta');
const valP = document.getElementById('valP');
const valK = document.getElementById('valK');
const valZeta = document.getElementById('valZeta');

const infoDeltaSt = document.getElementById('infoDeltaSt');
const infoSigmaSt = document.getElementById('infoSigmaSt');
const infoDeltaD = document.getElementById('infoDeltaD');
const infoSigmaD = document.getElementById('infoSigmaD');
const infoKd = document.getElementById('infoKd');

const formulaKdVal = document.getElementById('formulaKdVal');
const formulaSigmaVal = document.getElementById('formulaSigmaVal');

const modeStatus = document.getElementById('modeStatus');
const statusDetail = document.getElementById('statusDetail');

const btnStatic = document.getElementById('btnStatic');
const btnSudden = document.getElementById('btnSudden');
const btnReset = document.getElementById('btnReset');

// ---------- 物理参数 ----------
```

```
const g = 9.81;
const A = 1e-4; // 截面积 m² (固定)
let P = 400; // N
let k = 8000; // N/m
let zeta = 0.05; // 阻尼比

// 计算物理量
let deltaSt = 0; // 静变形
let sigmaSt = 0; // 静应力
let omega = 0; // 固有频率

// 动画状态
let mode = 'static'; // 'static' | 'sudden'
let time = 0;
let delta = 0; // 当前变形
let deltaMax = 0; // 最大变形 (用于显示)
let animId = null;
let isAnimating = false;
let resetRequested = false;

// 画布几何
const topY = 60;
const bottomY = 390;
const L0 = bottomY - topY; // 杆件原始长度 (px)
const cx = W / 2;
const scaleFactor = 280; // 变形缩放 (px/m)
```

```
// ---------- 物理计算 ----------
function computePhysics() {
    deltaSt = P / k;
    sigmaSt = P / A;
    const m = P / g;
    omega = Math.sqrt(k / m);
}

// ---------- 更新显示数值 ----------
function updateDisplay() {
    computePhysics();

    const deltaD = (mode === 'sudden') ? 2 * deltaSt : deltaSt;
    const sigmaD = (mode === 'sudden') ? 2 * sigmaSt : sigmaSt;
    const Kd = (mode === 'sudden') ? 2.0 : 1.0;

    infoDeltaSt.textContent = (deltaSt * 1000).toFixed(2) + ' mm';
    infoSigmaSt.textContent = (sigmaSt / 1e6).toFixed(2) + ' MPa';
    infoDeltaD.textContent = (deltaD * 1000).toFixed(2) + ' mm';
    infoSigmaD.textContent = (sigmaD / 1e6).toFixed(2) + ' MPa';
    infoKd.textContent = Kd.toFixed(3);

    formulaKdVal.textContent = Kd.toFixed(3);
    formulaSigmaVal.textContent = (sigmaD / 1e6).toFixed(2) + ' MPa';
}

// ---------- 绘制场景 ----------
```

```
function drawScene(currentDelta, maxDelta, phase) {
    ctx.clearRect(0, 0, W, H);

    // ---- 背景网格 ----
    ctx.save();
    ctx.strokeStyle = '#eef2f7';
    ctx.lineWidth = 0.5;
    for (let i = 0; i < 20; i++) {
        const x = 20 + i * 40;
        ctx.beginPath();
        ctx.moveTo(x, 20);
        ctx.lineTo(x, H - 20);
        ctx.stroke();
    }
    for (let i = 0; i < 15; i++) {
        const y = 20 + i * 32;
        ctx.beginPath();
        ctx.moveTo(20, y);
        ctx.lineTo(W - 20, y);
        ctx.stroke();
    }
    ctx.restore();

    // ---- 固定端 ----
    ctx.save();
    ctx.fillStyle = '#34495e';
    ctx.shadowColor = 'rgba(0,0,0,0.08)';
```

```
ctx.shadowBlur = 8;
ctx.fillRect(cx - 55, topY - 18, 110, 18);
ctx.shadowBlur = 0;
ctx.fillStyle = '#5d6d7e';
for (let i = -35; i <= 35; i += 18) {
    ctx.beginPath();
    ctx.arc(cx + i, topY - 9, 4, 0, 2 * Math.PI);
    ctx.fill();
}
ctx.fillStyle = '#2c3e50';
ctx.font = '13px "Segoe UI", sans-serif';
ctx.textAlign = 'center';
ctx.textBaseline = 'bottom';
ctx.fillText('固定端', cx, topY - 22);
ctx.restore();

// ---- 原始位置参考线（虚线）----
ctx.save();
ctx.setLineDash([5, 6]);
ctx.strokeStyle = 'rgba(44,62,80,0.25)';
ctx.lineWidth = 1.2;
ctx.beginPath();
ctx.moveTo(cx - 60, bottomY);
ctx.lineTo(cx + 60, bottomY);
ctx.stroke();
ctx.setLineDash([]);
ctx.fillStyle = 'rgba(44,62,80,0.3)';
```

```
                ctx.font = '11px "Segoe UI", sans-serif';
                ctx.textAlign = 'left';
                ctx.textBaseline = 'top';
                ctx.fillText('原始位置', cx + 64, bottomY - 6);
                ctx.restore();

                // ---- 杆件 ----
                const deformPx = currentDelta * scaleFactor;
                const rodBottom = bottomY + deformPx;
                const rodLength = rodBottom - topY;
                const rodWidth = 18;

                // 杆件颜色：根据应力状态变化
                const stressRatio = Math.min(Math.abs(currentDelta) / (deltaSt * 2 +
0.001), 1.0);
                const r = 40 + 180 * stressRatio;
                const gCol = 180 - 140 * stressRatio;
                const b = 200 - 170 * stressRatio;
                const rodColor = `rgb(${Math.min(r,255)}, ${Math.max(gCol,40)},
${Math.max(b,40)})`;

                ctx.save();
                // 杆件阴影
                ctx.shadowColor = 'rgba(0,0,0,0.06)';
                ctx.shadowBlur = 10;
                ctx.shadowOffsetX = 2;
                ctx.shadowOffsetY = 2;
```

```
                // 杆件主体 (带渐变)
                const grad = ctx.createLinearGradient(cx - rodWidth / 2, topY, cx + rodWidth / 2, topY);
                grad.addColorStop(0, '#5d6d7e');
                grad.addColorStop(0.2, rodColor);
                grad.addColorStop(0.8, rodColor);
                grad.addColorStop(1, '#5d6d7e');
                ctx.fillStyle = grad;

                const rx = cx - rodWidth / 2;
                const ry = topY;
                const rw = rodWidth;
                const rh = rodLength;
                if (rh > 2) {
                    ctx.fillRect(rx, ry, rw, rh);
                }

                ctx.shadowBlur = 0;
                ctx.strokeStyle = '#2c3e50';
                ctx.lineWidth = 1.2;
                if (rh > 2) {
                    ctx.strokeRect(rx, ry, rw, rh);
                }

                // 杆件上的刻度标记
                if (Math.abs(currentDelta) > 0.0005) {
```

```
                ctx.strokeStyle = 'rgba(44,62,80,0.25)';
                ctx.lineWidth = 0.8;
                const numMarks = 6;
                for (let i = 1; i < numMarks; i++) {
                    const t = i / numMarks;
                    const yPos = topY + t * rodLength;
                    const offset = 5 + 8 * (1 - Math.abs(currentDelta) / (deltaSt
* 2 + 0.001));
                    ctx.beginPath();
                    ctx.moveTo(cx - rodWidth / 2 - offset, yPos);
                    ctx.lineTo(cx - rodWidth / 2 - 2, yPos);
                    ctx.stroke();
                    ctx.beginPath();
                    ctx.moveTo(cx + rodWidth / 2 + offset, yPos);
                    ctx.lineTo(cx + rodWidth / 2 + 2, yPos);
                    ctx.stroke();
                }
            }
            ctx.restore();

            // ---- 变形标注 ----
            if (Math.abs(currentDelta) > 0.0001) {
                ctx.save();
                const sign = currentDelta >= 0 ? '+' : '';
                const deltaDisplay = (currentDelta * 1000).toFixed(1);
                ctx.fillStyle = '#c0392b';
                ctx.font = 'bold 14px "Segoe UI", sans-serif';
```

```
ctx.textAlign = 'left';
ctx.textBaseline = 'bottom';
const labelX = cx + rodWidth / 2 + 14;
const labelY = rodBottom - 4;
ctx.fillText(`δ  = ${sign}${deltaDisplay} mm`, labelX, labelY);

// 变形箭头线
ctx.strokeStyle = 'rgba(192,57,43,0.35)';
ctx.lineWidth = 1.4;
ctx.setLineDash([4, 4]);
const arrowX = cx + rodWidth / 2 + 52;
ctx.beginPath();
ctx.moveTo(arrowX, bottomY);
ctx.lineTo(arrowX, rodBottom);
ctx.stroke();
ctx.setLineDash([]);
// 箭头头
if (Math.abs(currentDelta) > 0.0005) {
    ctx.fillStyle = 'rgba(192,57,43,0.4)';
    const dir = currentDelta >= 0 ? 1 : -1;
    const tipY = rodBottom;
    ctx.beginPath();
    ctx.moveTo(arrowX, tipY);
    ctx.lineTo(arrowX - 6, tipY - 8 * dir);
    ctx.lineTo(arrowX + 6, tipY - 8 * dir);
    ctx.closePath();
    ctx.fill();
```

```
            }
            ctx.restore();
        }

        // ---- 载荷箭头 (底部) ----
        const loadY = rodBottom + 14;
        ctx.save();
        const loadColor = (mode === 'sudden' && currentDelta > deltaSt * 1.1) ? '#e74c3c' : '#d35400';
        ctx.fillStyle = loadColor;
        ctx.strokeStyle = loadColor;
        ctx.lineWidth = 2.8;

        // 箭头线
        ctx.beginPath();
        ctx.moveTo(cx, loadY);
        ctx.lineTo(cx, loadY + 50);
        ctx.stroke();

        // 箭头头
        ctx.beginPath();
        ctx.moveTo(cx - 10, loadY + 40);
        ctx.lineTo(cx, loadY + 50);
        ctx.lineTo(cx + 10, loadY + 40);
        ctx.closePath();
        ctx.fill();
```

```
                    // 载荷标签
                    ctx.fillStyle = '#2c3e50';
                    ctx.font = 'bold 15px "Segoe UI", sans-serif';
                    ctx.textAlign = 'center';
                    ctx.textBaseline = 'top';
                    const label = mode === 'static' ? `P = ${P.toFixed(0)} N (静载)` : `P = ${P.toFixed(0)} N (突加)`;
                    ctx.fillText(label, cx, loadY + 56);

                    // 重物方块 (代表质量)
                    ctx.fillStyle = '#e67e22';
                    ctx.shadowColor = 'rgba(0,0,0,0.1)';
                    ctx.shadowBlur = 8;
                    const blockSize = 28;
                    const bx = cx - blockSize / 2;
                    const by = loadY - 8 - blockSize;
                    ctx.fillRect(bx, by, blockSize, blockSize);
                    ctx.shadowBlur = 0;
                    ctx.strokeStyle = '#7f3f00';
                    ctx.lineWidth = 1.2;
                    ctx.strokeRect(bx, by, blockSize, blockSize);
                    ctx.fillStyle = '#fff';
                    ctx.font = 'bold 12px "Segoe UI", sans-serif';
                    ctx.textAlign = 'center';
                    ctx.textBaseline = 'middle';
                    ctx.fillText('m', cx, by + blockSize / 2);
                    ctx.restore();
```

```
// ---- 工况标签 ----
ctx.save();
ctx.fillStyle = '#2c3e50';
ctx.font = 'bold 16px "Segoe UI", sans-serif';
ctx.textAlign = 'left';
ctx.textBaseline = 'top';
const modeLabel = mode === 'static' ? '📊 静载荷工况' : '💥 突加载荷工况';
ctx.fillText(modeLabel, 24, 22);

// 显示动荷系数
const Kd = mode === 'sudden' ? 2.0 : 1.0;
ctx.fillStyle = '#27ae60';
ctx.font = 'bold 15px "Segoe UI", sans-serif';
ctx.textAlign = 'right';
ctx.textBaseline = 'top';
ctx.fillText(`K_d = ${Kd.toFixed(3)}`, W - 24, 22);

// 显示当前变形与最大变形
ctx.fillStyle = '#6b7f93';
ctx.font = '13px "Segoe UI", sans-serif';
ctx.textAlign = 'right';
ctx.textBaseline = 'top';
const deltaCur = (currentDelta * 1000).toFixed(1);
const deltaMaxDisp = (maxDelta * 1000).toFixed(1);
```

```
                ctx.fillText(`当前变形: ${deltaCur} mm  |  最大变形: ${deltaMaxDisp} mm`, W - 24, 44);

                ctx.restore();

                // ---- 图例 ----
                ctx.save();
                const legX = 24,
                    legY = 74;
                ctx.font = '12px "Segoe UI", sans-serif';
                ctx.textAlign = 'left';
                ctx.textBaseline = 'top';
                ctx.fillStyle = '#6b7f93';
                ctx.fillText('🟦 杆件 (刚度 k)', legX, legY);
                ctx.fillStyle = '#d35400';
                ctx.fillText('🟧 载荷 P', legX + 140, legY);
                ctx.fillStyle = '#c0392b';
                ctx.fillText('↕ 变形 δ', legX + 240, legY);
                ctx.fillStyle = '#27ae60';
                ctx.fillText(`⚡ K_d = ${Kd.toFixed(2)}`, legX + 340, legY);
                ctx.restore();

                // ---- 静载参考线 (显示静变形位置) ----
                if (mode === 'sudden' && deltaSt > 0.0001) {
                    ctx.save();
```

```
            const stY = bottomY + deltaSt * scaleFactor;
            ctx.setLineDash([3, 5]);
            ctx.strokeStyle = 'rgba(39,174,96,0.5)';
            ctx.lineWidth = 1.4;
            ctx.beginPath();
            ctx.moveTo(cx - 44, stY);
            ctx.lineTo(cx + 44, stY);
            ctx.stroke();
            ctx.setLineDash([]);
            ctx.fillStyle = 'rgba(39,174,96,0.6)';
            ctx.font = '11px "Segoe UI", sans-serif';
            ctx.textAlign = 'left';
            ctx.textBaseline = 'bottom';
            ctx.fillText('δ_st', cx + 48, stY - 2);
            ctx.restore();
        }
    }

    // ---------- 动画更新 ----------
    function updateAnimation(dt) {
        if (resetRequested) {
            resetRequested = false;
            time = 0;
            delta = 0;
            deltaMax = 0;
            if (mode === 'static') {
                delta = deltaSt;
```

```
            deltaMax = deltaSt;
        }
        return;
    }

    const dtClamped = Math.min(dt, 0.025);

    if (mode === 'static') {
        // 静载：稳定在静变形
        delta = deltaSt;
        deltaMax = deltaSt;
        return;
    }

    // ---- 突加载荷 ----
    const m = P / g;
    const omegaVal = Math.sqrt(k / m);
    const omegaD = omegaVal * Math.sqrt(1 - zeta * zeta);

    // 解析解：δ(t) = δ_st * (1 - e^(-ζωt) * (cos(ωd*t) + ζ/√(1-ζ²) *
sin(ωd*t)))
    // 当 ζ=0 时：δ(t) = δ_st * (1 - cos(ω*t))
    let deltaT;
    if (zeta < 0.001) {
        deltaT = deltaSt * (1 - Math.cos(omegaVal * time));
    } else {
        const expTerm = Math.exp(-zeta * omegaVal * time);
```

```
            const cosTerm = Math.cos(omegaD * time);
            const sinTerm = Math.sin(omegaD * time);
            const factor = zeta / Math.sqrt(1 - zeta * zeta);
            deltaT = deltaSt * (1 - expTerm * (cosTerm + factor * sinTerm));
        }

        // 确保非负
        if (deltaT < 0) deltaT = 0;
        delta = deltaT;

        // 更新最大变形
        if (delta > deltaMax) {
            deltaMax = delta;
        }

        // 如果变形趋近于静变形且振荡幅度很小，认为稳定
        // 但不自动停止，让用户观察

        // 时间推进
        time += dtClamped;
    }

    // ---------- 动画循环 ----------
    let lastTime = 0;
    let frameCount = 0;

    function animate(timestamp) {
```

```
        if (!lastTime) lastTime = timestamp;
        const dt = Math.min((timestamp - lastTime) / 1000, 0.05);
        lastTime = timestamp;

        // 更新物理
        updateAnimation(dt);

        // 绘制
        const currentMax = (mode === 'sudden') ? Math.max(deltaMax, deltaSt) : deltaSt;
        drawScene(delta, currentMax, mode);

        // 更新状态显示
        const Kd = mode === 'sudden' ? 2.0 : 1.0;
        const deltaD = mode === 'sudden' ? 2 * deltaSt : deltaSt;
        const sigmaD = mode === 'sudden' ? 2 * sigmaSt : sigmaSt;
        infoDeltaD.textContent = (deltaD * 1000).toFixed(2) + ' mm';
        infoSigmaD.textContent = (sigmaD / 1e6).toFixed(2) + ' MPa';
        infoKd.textContent = Kd.toFixed(3);
        formulaKdVal.textContent = Kd.toFixed(3);
        formulaSigmaVal.textContent = (sigmaD / 1e6).toFixed(2) + ' MPa';

        animId = requestAnimationFrame(animate);
    }

    // ---------- 切换工况 ----------
    function setMode(newMode) {
```

```
if (newMode === mode && mode !== 'static') {
    // 如果相同模式，复位重新开始
}
mode = newMode;
time = 0;
delta = 0;
deltaMax = 0;

// 更新按钮状态
btnStatic.classList.toggle('active', mode === 'static');
btnSudden.classList.toggle('active', mode === 'sudden');

// 更新模式标签
modeStatus.textContent = mode === 'static' ? '静载荷' : '突加载荷';
statusDetail.textContent = mode === 'static' ?
    '杆件在静载作用下稳定变形' :
    '载荷突加，杆件产生振动，最大变形为静变形的 2 倍';

// 更新物理计算
computePhysics();
updateDisplay();

if (mode === 'static') {
    delta = deltaSt;
    deltaMax = deltaSt;
} else {
    // 突加载荷从 0 开始
```

```
            delta = 0;
            deltaMax = 0;
        }
    }

    // ---------- 复位 ----------
    function resetAnimation() {
        resetRequested = true;
        time = 0;
        delta = 0;
        deltaMax = 0;
        if (mode === 'static') {
            delta = deltaSt;
            deltaMax = deltaSt;
        }
        computePhysics();
        updateDisplay();
        statusDetail.textContent = mode === 'static' ?
            '杆件在静载作用下稳定变形' :
            '载荷突加，杆件产生振动，最大变形为静变形的 2 倍';
    }

    // ---------- 滑块更新 ----------
    function updateParams() {
        P = parseFloat(sliderP.value);
        k = parseFloat(sliderK.value);
        zeta = parseFloat(sliderZeta.value);
```

```
        valP.textContent = P.toFixed(0);
        valK.textContent = k.toFixed(0);
        valZeta.textContent = zeta.toFixed(2);

        computePhysics();
        updateDisplay();

        if (mode === 'static') {
            delta = deltaSt;
            deltaMax = deltaSt;
        } else {
            // 突加载荷：重置动画以应用新参数
            time = 0;
            delta = 0;
            deltaMax = 0;
        }
    }

    // ---------- 事件绑定 ----------
    sliderP.addEventListener('input', updateParams);
    sliderK.addEventListener('input', updateParams);
    sliderZeta.addEventListener('input', updateParams);

    btnStatic.addEventListener('click', function() {
        setMode('static');
    });
```

```
        btnSudden.addEventListener('click', function() {
            setMode('sudden');
        });

        btnReset.addEventListener('click', resetAnimation);

        // ---------- 窗口 resize ----------
        let resizeTimer;
        window.addEventListener('resize', function() {
            clearTimeout(resizeTimer);
            resizeTimer = setTimeout(() => {
                drawScene(delta, (mode === 'sudden') ? Math.max(deltaMax, deltaSt) : deltaSt, mode);
            }, 150);
        });

        // ---------- 初始化 ----------
        computePhysics();
        updateDisplay();
        setMode('static');

        // 启动动画
        animId = requestAnimationFrame(animate);

        console.log('✅ 静载荷 vs 突加载荷演示已启动');
        console.log(`P=${P}N, k=${k}N/m, ζ=${zeta}, δ_st=${(deltaSt*1000).toFixed(2)}mm`);
```

```
            console.log('突加载荷动荷系数  K_d = 2 (恒成立)');

            // 暴露全局变量用于调试
            window.demo = {
                setMode,
                resetAnimation,
                updateParams,
                P: () => P,
                k: () => k,
                deltaSt: () => deltaSt
            };
        })();
    </script>

</body>
</html>
```