```
<!DOCTYPE html>
<html lang="zh-CN">
<head>
    <meta charset="UTF-8">
    <meta name="viewport" content="width=device-width, initial-scale=1.0">
    <title>材料力学 - 组合变形与叠加原理 交互演示</title>
    <style>
        :root {
            --bg: #1a1d23;
            --panel-bg: #21252b;
            --border: #333842;
            --text: #cdd6e4;
            --text-secondary: #8b95a8;
            --accent: #61afef;
            --danger: #e06c75;
            --warning: #d19a66;
            --success: #98c379;
            --slider-track: #3a3f4b;
            --slider-fill: #61afef;
            --card-bg: #282c34;
            --highlight: #ff6b6b;
        }

        * {
            margin: 0;
            padding: 0;
            box-sizing: border-box;
        }

        body {
            font-family: 'Segoe UI', 'PingFang SC', 'Microsoft YaHei', 'Helvetica Neue', sans-serif;
            background: var(--bg);
            color: var(--text);
            display: flex;
            flex-direction: column;
            align-items: center;
            min-height: 100vh;
            padding: 10px 16px;
            user-select: none;
            -webkit-user-select: none;
        }

        .main-container {
            width: 100%;
```

```
        max-width: 1300px;
        display: flex;
        flex-direction: column;
        gap: 12px;
    }

    /* 顶部标题栏 */
    .header {
        display: flex;
        align-items: center;
        justify-content: space-between;
        flex-wrap: wrap;
        gap: 10px;
        padding: 8px 0;
    }
    .header h1 {
        font-size: 1.55rem;
        font-weight: 700;
        letter-spacing: 0.02em;
        background: linear-gradient(135deg, #61afef, #a78bfa);
        -webkit-background-clip: text;
        -webkit-text-fill-color: transparent;
        background-clip: text;
    }
    .header .badge {
        font-size: 0.8rem;
        padding: 5px 12px;
        border-radius: 20px;
        background: var(--card-bg);
        border: 1px solid var(--border);
        color: var(--text-secondary);
        letter-spacing: 0.03em;
    }
    .mode-switch {
        display: flex;
        gap: 4px;
        background: var(--card-bg);
        border-radius: 10px;
        padding: 3px;
        border: 1px solid var(--border);
    }
    .mode-btn {
        padding: 7px 16px;
        border-radius: 8px;
```

```
        border: none;
        cursor: pointer;
        font-size: 0.85rem;
        font-weight: 600;
        letter-spacing: 0.02em;
        transition: all 0.25s;
        background: transparent;
        color: var(--text-secondary);
        white-space: nowrap;
}
.mode-btn.active {
        background: var(--accent);
        color: #fff;
        box-shadow: 0 2px 10px rgba(97, 175, 239, 0.35);
}
.mode-btn:hover:not(.active) {
        color: var(--text);
        background: rgba(255, 255, 255, 0.04);
}

/* 上方：双图区域 */
.visualization-area {
        display: flex;
        gap: 14px;
        flex-wrap: wrap;
}
.viz-panel {
        flex: 1;
        min-width: 340px;
        background: var(--panel-bg);
        border-radius: 14px;
        border: 1px solid var(--border);
        padding: 14px 16px;
        display: flex;
        flex-direction: column;
        gap: 8px;
        box-shadow: 0 4px 20px rgba(0, 0, 0, 0.3);
}
.viz-panel .panel-title {
        font-size: 0.9rem;
        font-weight: 700;
        letter-spacing: 0.04em;
        color: var(--text);
        text-align: center;
```

```
        text-transform: uppercase;
        opacity: 0.85;
}
.canvas-wrapper {
        position: relative;
        flex: 1;
        display: flex;
        align-items: center;
        justify-content: center;
        min-height: 300px;
}
canvas {
        display: block;
        max-width: 100%;
        max-height: 100%;
}

/* 中间：滑块区域 */
.sliders-area {
        background: var(--panel-bg);
        border-radius: 14px;
        border: 1px solid var(--border);
        padding: 16px 20px;
        display: flex;
        flex-wrap: wrap;
        gap: 16px;
        align-items: flex-end;
        box-shadow: 0 4px 20px rgba(0, 0, 0, 0.3);
        justify-content: center;
}
.slider-group {
        flex: 1;
        min-width: 140px;
        max-width: 280px;
        display: flex;
        flex-direction: column;
        gap: 4px;
}
.slider-group label {
        font-size: 0.8rem;
        font-weight: 600;
        letter-spacing: 0.03em;
        color: var(--text-secondary);
        display: flex;
```

```
        justify-content: space-between;
        align-items: baseline;
    }
    .slider-group label .val {
        font-weight: 700;
        font-size: 0.95rem;
        color: var(--text);
        font-variant-numeric: tabular-nums;
        font-family: 'SF Mono', 'Consolas', 'Monaco', 'JetBrains Mono', monospace;
        min-width: 70px;
        text-align: right;
    }
    .slider-group label .unit {
        font-size: 0.7rem;
        color: var(--text-secondary);
        margin-left: 2px;
        font-weight: 400;
    }
    input[type="range"] {
        -webkit-appearance: none;
        width: 100%;
        height: 8px;
        border-radius: 4px;
        background: var(--slider-track);
        outline: none;
        cursor: pointer;
        margin: 4px 0;
    }
    input[type="range"]::-webkit-slider-thumb {
        -webkit-appearance: none;
        width: 26px;
        height: 26px;
        border-radius: 50%;
        background: #fff;
        border: 2px solid var(--accent);
        cursor: pointer;
        box-shadow: 0 2px 8px rgba(0, 0, 0, 0.35);
        transition: transform 0.15s;
    }
    input[type="range"]::-webkit-slider-thumb:active {
        transform: scale(1.15);
    }
    .slider-color-f {
        accent-color: #e06c75;
```

```
}
.slider-color-m {
    accent-color: #61afef;
}
.slider-color-t {
    accent-color: #d19a66;
}
.slider-color-f::-webkit-slider-thumb {
    border-color: #e06c75;
    background: #ffe0e3;
}
.slider-color-m::-webkit-slider-thumb {
    border-color: #61afef;
    background: #e0f0ff;
}
.slider-color-t::-webkit-slider-thumb {
    border-color: #d19a66;
    background: #fff3e8;
}

/* 下方：公式区域 */
.formula-area {
    background: var(--panel-bg);
    border-radius: 14px;
    border: 1px solid var(--border);
    padding: 16px 20px;
    box-shadow: 0 4px 20px rgba(0, 0, 0, 0.3);
    display: flex;
    flex-wrap: wrap;
    gap: 16px;
}
.formula-block {
    flex: 1;
    min-width: 260px;
    background: var(--card-bg);
    border-radius: 10px;
    padding: 14px 16px;
    border: 1px solid var(--border);
}
.formula-block h3 {
    font-size: 0.85rem;
    font-weight: 700;
    letter-spacing: 0.04em;
    margin-bottom: 8px;
```

```
            color: var(--accent);
            text-transform: uppercase;
        }
        .formula-block .eq {
            font-family: 'Georgia', 'Times New Roman', 'STIX Two Math', 'Latin Modern Math',
serif;
            font-size: 1rem;
            color: var(--text);
            padding: 6px 0;
            letter-spacing: 0.02em;
            line-height: 1.7;
        }
        .formula-block .eq .hl {
            color: #e06c75;
            font-weight: 700;
        }
        .formula-block .eq .hl2 {
            color: #61afef;
            font-weight: 700;
        }
        .formula-block .eq .hl3 {
            color: #d19a66;
            font-weight: 700;
        }
        .formula-block .result-row {
            display: flex;
            justify-content: space-between;
            align-items: center;
            padding: 5px 0;
            border-bottom: 1px solid rgba(255, 255, 255, 0.05);
            font-size: 0.85rem;
        }
        .formula-block .result-row .res-val {
            font-weight: 700;
            font-family: 'SF Mono', 'Consolas', 'Monaco', monospace;
            font-size: 0.9rem;
        }
        .check-pass {
            color: #98c379;
        }
        .check-fail {
            color: #e06c75;
            animation: pulse-warn 1.2s ease-in-out infinite;
        }
```

```
            @keyframes pulse-warn {
                0%,
                100% {
                    opacity: 1;
                }
                50% {
                    opacity: 0.5;
                }
            }
            .divider {
                width: 1px;
                background: var(--border);
                flex-shrink: 0;
                margin: 0 4px;
            }

            /* 响应式 */
            @media (max-width: 900px) {
                .visualization-area {
                    flex-direction: column;
                }
                .viz-panel {
                    min-width: auto;
                }
                .sliders-area {
                    gap: 10px;
                    padding: 12px;
                }
                .slider-group {
                    min-width: 100px;
                    max-width: none;
                }
                .formula-area {
                    flex-direction: column;
                }
                .divider {
                    width: auto;
                    height: 1px;
                    margin: 4px 0;
                }
            }
        </style>
    </head>
    <body>
```

```
    <div class="main-container">
        <!-- 标题 -->
        <div class="header">
            <h1>📐 组合变形与叠加原理 • 交互演示</h1>
            <div style="display:flex;align-items:center;gap:10px;flex-wrap:wrap;">
                <span class="badge">圆形截面 d=50mm</span>
                <span class="badge">许用应力 [σ]=160MPa</span>
                <div class="mode-switch" id="modeSwitch">
                    <button class="mode-btn active" data-mode="all">全部载荷</button>
                    <button class="mode-btn" data-mode="tension-bend">拉弯组合</button>
                    <button class="mode-btn" data-mode="bend-torsion">弯扭组合</button>
                </div>
            </div>
        </div>

        <!-- 上方：双图 -->
        <div class="visualization-area">
            <div class="viz-panel" id="panelLeft">
                <div class="panel-title">🏗️ 构件受力示意图</div>
                <div class="canvas-wrapper">
                    <canvas id="canvasBeam" width="520" height="340"></canvas>
                </div>
            </div>
            <div class="viz-panel" id="panelRight">
                <div class="panel-title">🔍 截面应力分布图</div>
                <div class="canvas-wrapper">
                    <canvas id="canvasSection" width="420" height="340"></canvas>
                </div>
            </div>
        </div>

        <!-- 中间：滑块 -->
        <div class="sliders-area" id="slidersArea">
            <div class="slider-group">
                <label>🔴 轴 力 <span class="val" id="valF">80</span><span class="unit">kN</span></label>
                <input type="range" class="slider-color-f" id="sliderF" min="0" max="200" value="80" step="1">
            </div>
            <div class="slider-group">
                <label>🔵 弯 矩 <span class="val" id="valM">2.00</span><span class="unit">kN • m</span></label>
```

```
                <input type="range" class="slider-color-m" id="sliderM" min="0" max="5" value="2" step="0.01">
            </div>
            <div class="slider-group">
                <label>🟠 扭 矩 <span class="val" id="valT">1.50</span><span class="unit">kN • m</span></label>
                <input type="range" class="slider-color-t" id="sliderT" min="0" max="5" value="1.5" step="0.01">
            </div>
            <div class="slider-group" style="max-width:180px;">
                <label>◯ 许 用 应 力 <span class="val" id="valSigmaAllow">160</span><span class="unit">MPa</span></label>
                <input type="range" id="sliderSigmaAllow" min="60" max="300" value="160" step="5"
                style="accent-color:#98c379;">
            </div>
        </div>

        <!-- 下方：公式 -->
        <div class="formula-area" id="formulaArea">
            <div class="formula-block" id="formulaBlock1">
                <h3>📝 叠加法核心公式</h3>
                <div class="eq">
                    正 应 力 叠 加 ： <span class="hl"> σ (y)</span> = <span class="hl2">F/A</span> + <span class="hl3">M • y/I</span><br>
                    切 应 力 （ 扭 转 ）： <span class="hl"> τ ( ρ )</span> = <span class="hl3">T • ρ /J</span><br>
                    截面参数：A=1963.5mm² , W=12272mm³ , W<sub>t</sub>=24544mm³
                </div>
            </div>
            <div class="divider"></div>
            <div class="formula-block" id="formulaBlock2">
                <h3>⚠ 危险点应力 & 强度校核</h3>
                <div class="eq" id="dangerPointFormula">
                    危险点(截面顶部)：<span class="hl"> σ </span>=<span id="dispSigma">--</span> MPa,
                    <span class="hl"> τ </span>=<span id="dispTau">--</span> MPa
                </div>
                <div class="result-row">
                    <span>第三强度理论 σ<sub>r3</sub>= √ ( σ ² +4 τ ² )</span>
                    <span class="res-val" id="dispSigmaR3">--</span> MPa
                    <span id="checkR3" style="font-weight:700;">--</span>
                </div>
                <div class="result-row">
```

```
                    <span>第四强度理论 σ<sub>r4</sub>= √(σ²+3τ²)</span>
                    <span class="res-val" id="dispSigmaR4">--</span> MPa
                    <span id="checkR4" style="font-weight:700;">--</span>
                </div>
                <div class="result-row" style="border-bottom:none;">
                    <span>最大正应力 σ<sub>max</sub></span>
                    <span class="res-val" id="dispSigmaMax">--</span> MPa
                    <span id="checkSigmaMax" style="font-weight:700;">--</span>
                </div>
            </div>
            <div class="divider"></div>
            <div class="formula-block" id="formulaBlock3">
                <h3>📊 中性轴位置（拉弯组合）</h3>
                <div class="eq">
                    中性轴：σ(y₀)=0 → <span class="hl">y₀</span> = −(F/A)・(I/M)
                </div>
                <div class="result-row">
                    <span>中性轴偏移 y₀ =</span>
                    <span class="res-val" id="dispNeutralAxis">--</span> mm
                </div>
                <div class="result-row">
                    <span>截面半高 d/2 =</span>
                    <span class="res-val">25.0</span> mm
                </div>
                <div style="font-size:0.78rem;color:var(--text-secondary);margin-top:6px;">
                    <span id="neutralNote">--</span>
                </div>
            </div>
        </div>
    </div>

    <script>
        (function() {
            // ============ DOM 元素 ============
            const canvasBeam = document.getElementById('canvasBeam');
            const ctxBeam = canvasBeam.getContext('2d');
            const canvasSection = document.getElementById('canvasSection');
            const ctxSection = canvasSection.getContext('2d');

            const sliderF = document.getElementById('sliderF');
            const sliderM = document.getElementById('sliderM');
            const sliderT = document.getElementById('sliderT');
            const sliderSigmaAllow = document.getElementById('sliderSigmaAllow');
            const valF = document.getElementById('valF');
```

```
const valM = document.getElementById('valM');
const valT = document.getElementById('valT');
const valSigmaAllow = document.getElementById('valSigmaAllow');

const modeBtns = document.querySelectorAll('.mode-btn');

// ============ 截面参数（圆形截面 d=50mm）============
const d = 50; // mm
const r = d / 2; // 25 mm
const A = Math.PI * d * d / 4; // ~1963.5 mm²
const I = Math.PI * Math.pow(d, 4) / 64; // ~306796 mm⁴
const W = Math.PI * Math.pow(d, 3) / 32; // ~12272 mm³
const J = Math.PI * Math.pow(d, 4) / 32; // ~613592 mm⁴
const Wt = Math.PI * Math.pow(d, 3) / 16; // ~24544 mm³

// ============ 状态 ============
let mode = 'all'; // 'all' | 'tension-bend' | 'bend-torsion'
let F_kN = parseFloat(sliderF.value); // 轴力 kN
let M_kNm = parseFloat(sliderM.value); // 弯矩 kN·m
let T_kNm = parseFloat(sliderT.value); // 扭矩 kN·m
let sigmaAllow = parseFloat(sliderSigmaAllow.value); // 许用应力 MPa

// 转换到 N 和 N·mm
function getF_N() { return F_kN * 1000; }
function getM_Nmm() { return M_kNm * 1e6; }
function getT_Nmm() { return T_kNm * 1e6; }

// ============ 计算应力 ============
function calcStresses() {
    const F = getF_N();
    const M = getM_Nmm();
    const T = getT_Nmm();

    // 轴力产生的均匀正应力 (MPa)
    const sigma_F = F / A; // N/mm² = MPa
    // 弯曲正应力在 y=d/2 处 (MPa)
    const sigma_bend_max = M / W; // 顶部拉应力（正弯矩时）
    // 扭转切应力在 ρ=d/2 处 (MPa)
    const tau_torsion_max = T / Wt;

    // 截面顶部 (y=+25mm) 的正应力
    const sigma_top = sigma_F + sigma_bend_max;
    // 截面底部 (y=-25mm) 的正应力
    const sigma_bottom = sigma_F - sigma_bend_max;
```

```
        // 危险点（截面顶部边缘）：同时有最大正应力和最大切应力
        const sigma_danger = sigma_top;
        const tau_danger = tau_torsion_max;

        // 强度理论相当应力
        const sigma_r3 = Math.sqrt(sigma_danger * sigma_danger + 4 * tau_danger *
tau_danger);
        const sigma_r4 = Math.sqrt(sigma_danger * sigma_danger + 3 * tau_danger *
tau_danger);

        // 最大正应力绝对值
        const sigma_max_abs = Math.max(Math.abs(sigma_top),
Math.abs(sigma_bottom));

        // 中性轴位置（拉弯组合，M≠0 时）
        let y0 = null; // 中性轴 y 坐标(mm)，从形心算起
        if (Math.abs(M) > 1e-9) {
            y0 = -(sigma_F * I) / (M); // y0 = -(F/A)*(I/M) = -sigma_F * I / M
        }
        // 如果 M≈0，中性轴在无穷远（纯轴力，或由轴力主导）

        return {
            sigma_F,
            sigma_bend_max,
            tau_torsion_max,
            sigma_top,
            sigma_bottom,
            sigma_danger,
            tau_danger,
            sigma_r3,
            sigma_r4,
            sigma_max_abs,
            y0,
            F,
            M,
            T
        };
    }

    // ============ 绘制构件受力图 ============
    function drawBeamDiagram() {
        const c = ctxBeam;
        const w = canvasBeam.width;
```

```
const h = canvasBeam.height;
c.clearRect(0, 0, w, h);

const stresses = calcStresses();
const F = stresses.F;
const M = stresses.M;
const T = stresses.T;

// 3D 等距参数
const beamLen = 280; // 梁在屏幕上的长度
const beamDia = 55; // 梁在屏幕上的直径
const startX = 70;
const startY = h / 2;
const isoAngle = 0.55; // 等距倾斜角度（弧度，约 31 度）
const cosA = Math.cos(isoAngle);
const sinA = Math.sin(isoAngle);

const endX = startX + beamLen;
const endY = startY;
const wallX = startX - 10;

// 墙面
c.fillStyle = '#3a3f4b';
c.fillRect(wallX - 18, startY - 95, 22, 190);
c.fillStyle = '#4a505c';
c.fillRect(wallX - 14, startY - 85, 14, 170);
// 墙面的斜线纹理
c.strokeStyle = '#555b68';
c.lineWidth = 0.5;
for (let i = -80; i < 80; i += 14) {
    c.beginPath();
    c.moveTo(wallX - 14, startY + i);
    c.lineTo(wallX - 2, startY + i + 6);
    c.stroke();
}

// 梁的 3D 圆柱体
// 顶面椭圆
c.fillStyle = '#4a5568';
c.strokeStyle = '#7a8699';
c.lineWidth = 2;
c.beginPath();
c.ellipse(endX, endY - beamDia / 2, beamDia / 2, beamDia * 0.28, 0, 0, Math.PI
* 2);
```

```
                    c.fill();
                    c.stroke();

                    // 梁主体
                    const gradBeam = c.createLinearGradient(startX, startY - beamDia / 2, startX,
startY + beamDia / 2);
                    gradBeam.addColorStop(0, '#6b7d95');
                    gradBeam.addColorStop(0.35, '#8a9bb5');
                    gradBeam.addColorStop(0.5, '#9aabc5');
                    gradBeam.addColorStop(0.65, '#7d8ea8');
                    gradBeam.addColorStop(1, '#556378');
                    c.fillStyle = gradBeam;
                    c.strokeStyle = '#5a6b80';
                    c.lineWidth = 2;
                    c.beginPath();
                    c.moveTo(startX, startY - beamDia / 2);
                    c.lineTo(endX, endY - beamDia / 2);
                    c.arc(endX, endY, beamDia / 2, -Math.PI / 2, Math.PI / 2, false);
                    c.lineTo(startX, startY + beamDia / 2);
                    c.arc(startX, startY, beamDia / 2, Math.PI / 2, -Math.PI / 2, true);
                    c.closePath();
                    c.fill();
                    c.stroke();

                    // 梁上的高光线
                    c.strokeStyle = 'rgba(255,255,255,0.18)';
                    c.lineWidth = 1.5;
                    c.beginPath();
                    c.moveTo(startX + 8, startY - beamDia / 2 + 8);
                    c.lineTo(endX - 5, endY - beamDia / 2 + 8);
                    c.stroke();

                    // 固定端标记
                    c.fillStyle = '#2d323b';
                    c.strokeStyle = '#5a6375';
                    c.lineWidth = 2;
                    c.beginPath();
                    c.moveTo(startX - 6, startY - beamDia / 2 - 8);
                    c.lineTo(startX + 12, startY - beamDia / 2 - 8);
                    c.lineTo(startX + 12, startY + beamDia / 2 + 8);
                    c.lineTo(startX - 6, startY + beamDia / 2 + 8);
                    c.closePath();
                    c.fill();
                    c.stroke();
```

```
                // 螺栓点
                for (let by = -beamDia / 2 + 10; by <= beamDia / 2 - 8; by += 18) {
                    c.fillStyle = '#1a1d23';
                    c.beginPath();
                    c.arc(startX + 3, startY + by, 4, 0, Math.PI * 2);
                    c.fill();
                    c.strokeStyle = '#444';
                    c.lineWidth = 1;
                    c.stroke();
                }

                // --- 载荷箭头 ---
                const arrowScale = 1.0;
                const freeX = endX;
                const freeY = endY;

                // 轴力箭头（水平，沿梁轴线）
                const fMag = F / 1000; // kN
                if (fMag > 0.05 && (mode === 'all' || mode === 'tension-bend')) {
                    const arrowLen = Math.min(90, 20 + fMag * 0.35);
                    const ax = freeX + 20;
                    const ay = freeY;
                    drawArrow(c, ax, ay, ax + arrowLen, ay, '#e06c75', 3.5, 10, true);
                    // 标签
                    c.fillStyle = '#e06c75';
                    c.font = 'bold 13px "PingFang SC","Microsoft YaHei",sans-serif';
                    c.fillText('F=' + fMag.toFixed(0) + 'kN', ax + arrowLen / 2 - 20, ay - 18);
                }

                // 横向力箭头（产生弯矩），垂直向下
                const mMag = M_kNm;
                if (mMag > 0.005 && (mode === 'all' || mode === 'tension-bend' || mode ===
'bend-torsion')) {
                    const arrowLen = Math.min(80, 15 + mMag * 14);
                    const mx = freeX + 10;
                    const my = freeY + beamDia / 2 + 10;
                    drawArrow(c, mx, my, mx, my + arrowLen, '#61afef', 3.5, 10, true);
                    c.fillStyle = '#61afef';
                    c.font = 'bold 12px "PingFang SC","Microsoft YaHei",sans-serif';
                    c.fillText('M=' + mMag.toFixed(2) + 'kN • m', mx - 40, my + arrowLen / 2 +
5);
                }

                // 扭矩箭头（绕梁轴线旋转）
```

```
const tMag = T_kNm;
if (tMag > 0.005 && (mode === 'all' || mode === 'bend-torsion')) {
    const tx = freeX - 25;
    const ty = freeY - beamDia / 2 - 20;
    // 画旋转箭头（用弧线表示）
    c.strokeStyle = '#d19a66';
    c.lineWidth = 3.5;
    c.setLineDash([8, 3]);
    c.beginPath();
    c.arc(tx, ty, 18, -Math.PI * 0.7, Math.PI * 0.9, false);
    c.stroke();
    c.setLineDash([]);
    // 箭头尖
    const tipAngle = Math.PI * 0.9;
    const tipX = tx + 18 * Math.cos(tipAngle);
    const tipY = ty + 18 * Math.sin(tipAngle);
    drawSmallArrow(c, tipX, tipY, tipAngle + Math.PI / 2, '#d19a66', 8);
    c.fillStyle = '#d19a66';
    c.font = 'bold 12px "PingFang SC","Microsoft YaHei",sans-serif';
    c.fillText('T=' + tMag.toFixed(2) + 'kN・m', tx - 45, ty - 26);
}

// 截面指示线（在梁中间位置标记截面）
const secX = startX + beamLen * 0.45;
c.strokeStyle = 'rgba(255,255,200,0.7)';
c.lineWidth = 1.5;
c.setLineDash([4, 6]);
c.beginPath();
c.moveTo(secX, startY - beamDia / 2 - 6);
c.lineTo(secX, startY + beamDia / 2 + 6);
c.stroke();
c.setLineDash([]);
// 截面标签
c.fillStyle = 'rgba(255,255,200,0.85)';
c.font = '11px "PingFang SC","Microsoft YaHei",sans-serif';
c.fillText('截面 A-A', secX - 22, startY - beamDia / 2 - 12);

// 图例
const legendX = w - 155;
const legendY = 20;
c.fillStyle = 'rgba(0,0,0,0.5)';
c.fillRect(legendX - 8, legendY - 4, 148, 72);
c.strokeStyle = 'rgba(255,255,255,0.2)';
c.lineWidth = 1;
```

```
            c.strokeRect(legendX - 8, legendY - 4, 148, 72);

            c.fillStyle = '#e06c75';
            c.font = '11px "PingFang SC","Microsoft YaHei",sans-serif';
            c.fillText('■ 轴力 F (拉)', legendX, legendY + 16);
            c.fillStyle = '#61afef';
            c.fillText('■ 弯矩 M', legendX, legendY + 36);
            c.fillStyle = '#d19a66';
            c.fillText('■ 扭矩 T', legendX, legendY + 56);
        }

        function drawArrow(ctx, x1, y1, x2, y2, color, lineW, headSize, filled) {
            const dx = x2 - x1;
            const dy = y2 - y1;
            const len = Math.sqrt(dx * dx + dy * dy);
            if (len < 2) return;
            const ux = dx / len;
            const uy = dy / len;
            const nx = -uy;
            const ny = ux;

            ctx.strokeStyle = color;
            ctx.fillStyle = color;
            ctx.lineWidth = lineW;
            ctx.lineCap = 'round';
            ctx.lineJoin = 'round';

            ctx.beginPath();
            ctx.moveTo(x1, y1);
            ctx.lineTo(x2 - ux * headSize * 0.7, y2 - uy * headSize * 0.7);
            ctx.stroke();

            // 箭头头部
            ctx.beginPath();
            ctx.moveTo(x2, y2);
            ctx.lineTo(x2 - ux * headSize + nx * headSize * 0.45, y2 - uy * headSize + ny *
headSize * 0.45);
            ctx.lineTo(x2 - ux * headSize - nx * headSize * 0.45, y2 - uy * headSize - ny *
headSize * 0.45);
            ctx.closePath();
            if (filled) {
                ctx.fill();
            } else {
                ctx.stroke();
```

```
        }
    }

    function drawSmallArrow(ctx, x, y, angle, color, size) {
        ctx.fillStyle = color;
        ctx.beginPath();
        ctx.moveTo(x + size * Math.cos(angle), y + size * Math.sin(angle));
        ctx.lineTo(x + size * 0.4 * Math.cos(angle + 2.5), y + size * 0.4 * Math.sin(angle
+ 2.5));
        ctx.lineTo(x + size * 0.4 * Math.cos(angle - 2.5), y + size * 0.4 * Math.sin(angle
- 2.5));
        ctx.closePath();
        ctx.fill();
    }

    // ============ 绘制截面应力分布图 ============
    function drawSectionDiagram() {
        const c = ctxSection;
        const w = canvasSection.width;
        const h = canvasSection.height;
        c.clearRect(0, 0, w, h);

        const stresses = calcStresses();
        const cx = w / 2 - 10;
        const cy = h / 2;
        const radius = 115; // 截面圆半径（像素）

        // 背景网格
        c.strokeStyle = 'rgba(255,255,255,0.04)';
        c.lineWidth = 0.5;
        for (let gx = cx - radius - 30; gx <= cx + radius + 30; gx += 20) {
            c.beginPath();
            c.moveTo(gx, cy - radius - 20);
            c.lineTo(gx, cy + radius + 20);
            c.stroke();
        }
        for (let gy = cy - radius - 20; gy <= cy + radius + 20; gy += 20) {
            c.beginPath();
            c.moveTo(cx - radius - 30, gy);
            c.lineTo(cx + radius + 30, gy);
            c.stroke();
        }

        // 裁剪圆形区域用于正应力颜色填充
```

```
                c.save();
                c.beginPath();
                c.arc(cx, cy, radius, 0, Math.PI * 2);
                c.clip();

                // 正应力颜色填充
                const sigma_top = stresses.sigma_top;
                const sigma_bottom = stresses.sigma_bottom;
                const absMaxStress = Math.max(Math.abs(sigma_top),
Math.abs(sigma_bottom), 1);

                // 使用垂直渐变（在 canvas 中 y 轴向下，顶部 y 小，底部 y 大）
                const gradY1 = cy - radius;
                const gradY2 = cy + radius;
                const grad = c.createLinearGradient(cx, gradY1, cx, gradY2);

                // 在截面上采样多个点来确定色标
                const numStops = 40;
                for (let i = 0; i <= numStops; i++) {
                    const frac = i / numStops;
                    const yPhys = radius - frac * 2 * radius; // 物理 y 坐标(mm)，从+25 到-
25
                    const sigmaAtY = stresses.sigma_F + (stresses.sigma_bend_max * yPhys /
r);
                    const t = absMaxStress > 0.01 ? sigmaAtY / absMaxStress : 0;
                    const color = stressToColor(sigmaAtY, absMaxStress);
                    grad.addColorStop(frac, color);
                }
                c.fillStyle = grad;
                c.fillRect(cx - radius, cy - radius, radius * 2, radius * 2);

                // 切应力同心环表示（扭转）
                const tauMax = stresses.tau_torsion_max;
                if (tauMax > 0.3 && (mode === 'all' || mode === 'bend-torsion')) {
                    const numRings = 8;
                    for (let i = 1; i <= numRings; i++) {
                        const rhoFrac = i / numRings;
                        const ringR = radius * rhoFrac;
                        const tauAtRho = tauMax * rhoFrac;
                        const alpha = 0.08 + 0.22 * rhoFrac;
                        c.strokeStyle = `rgba(255,180,80,${alpha})`;
                        c.lineWidth = 1.8;
                        c.setLineDash([3, 4]);
                        c.beginPath();
```

```
                    c.arc(cx, cy, ringR, 0, Math.PI * 2);
                    c.stroke();
                    c.setLineDash([]);
                }
                // 切应力方向小箭头（环向）
                for (let a = 0; a < Math.PI * 2; a += Math.PI / 6) {
                    const arx = cx + radius * 0.92 * Math.cos(a);
                    const ary = cy + radius * 0.92 * Math.sin(a);
                    const arrowAngle = a + Math.PI / 2; // 环向（逆时针）
                    const arrowAlpha = 0.5 + 0.4 * (tauMax / 200);
                    c.fillStyle = `rgba(255,160,50,${Math.min(1,arrowAlpha)})`;
                    c.beginPath();
                    const s = 7;
                    c.moveTo(arx + s * Math.cos(arrowAngle), ary + s *
Math.sin(arrowAngle));
                    c.lineTo(arx + s * 0.35 * Math.cos(arrowAngle + 2.3), ary + s * 0.35 *
Math.sin(arrowAngle +
                        2.3));
                    c.lineTo(arx + s * 0.35 * Math.cos(arrowAngle - 2.3), ary + s * 0.35 *
Math.sin(arrowAngle -
                        2.3));
                    c.closePath();
                    c.fill();
                }
            }

            c.restore();

            // 圆形截面轮廓
            c.strokeStyle = '#8899bb';
            c.lineWidth = 3;
            c.beginPath();
            c.arc(cx, cy, radius, 0, Math.PI * 2);
            c.stroke();

            // 外圈装饰
            c.strokeStyle = 'rgba(255,255,255,0.25)';
            c.lineWidth = 1;
            c.beginPath();
            c.arc(cx, cy, radius + 3, 0, Math.PI * 2);
            c.stroke();

            // 形心十字线
            c.strokeStyle = 'rgba(255,255,255,0.3)';
```

```
                c.lineWidth = 0.8;
                c.setLineDash([5, 8]);
                c.beginPath();
                c.moveTo(cx - radius - 10, cy);
                c.lineTo(cx + radius + 10, cy);
                c.moveTo(cx, cy - radius - 10);
                c.lineTo(cx, cy + radius + 10);
                c.stroke();
                c.setLineDash([]);
                c.fillStyle = 'rgba(255,255,255,0.6)';
                c.font = '10px "PingFang SC","Microsoft YaHei",sans-serif';
                c.fillText('形心', cx + 6, cy - 6);

                // 中性轴（拉弯组合，M≠0 时）
                if (stresses.y0 !== null && Math.abs(stresses.M) > 1e-9) {
                    const y0Phys = stresses.y0; // mm
                    const y0Clamped = Math.max(-r, Math.min(r, y0Phys));
                    const y0Px = cy - (y0Clamped / r) * radius;
                    if (Math.abs(y0Phys) <= r + 5) {
                        const y0DrawPx = cy - (Math.max(-r - 5, Math.min(r + 5, y0Phys)) / r)
* radius;
                        c.strokeStyle = '#ffdd57';
                        c.lineWidth = 2;
                        c.setLineDash([7, 3]);
                        c.beginPath();
                        c.moveTo(cx - radius - 10, y0DrawPx);
                        c.lineTo(cx + radius + 10, y0DrawPx);
                        c.stroke();
                        c.setLineDash([]);
                        c.fillStyle = '#ffdd57';
                        c.font = 'bold 11px "PingFang SC","Microsoft YaHei",sans-serif';
                        c.fillText('中性轴 y₀=' + y0Phys.toFixed(1) + 'mm', cx - radius - 8,
y0DrawPx - 7);
                    }
                }

                // 危险点标记（截面顶部边缘）
                const dangerX = cx;
                const dangerY = cy - radius;
                const pulse = 0.5 + 0.5 * Math.sin(Date.now() / 400);

                // 危险点发光效果
                const glowGrad = c.createRadialGradient(dangerX, dangerY, 4, dangerX,
dangerY, 22);
```

```
                glowGrad.addColorStop(0, `rgba(255,80,80,${0.8*pulse+0.4})`);
                glowGrad.addColorStop(1, 'rgba(255,80,80,0)');
                c.fillStyle = glowGrad;
                c.beginPath();
                c.arc(dangerX, dangerY, 22, 0, Math.PI * 2);
                c.fill();

                // 危险点实心圆
                c.fillStyle = '#ff4444';
                c.strokeStyle = '#fff';
                c.lineWidth = 2.5;
                c.beginPath();
                c.arc(dangerX, dangerY, 6, 0, Math.PI * 2);
                c.fill();
                c.stroke();

                // 危险点标签
                c.fillStyle = '#fff';
                c.font = 'bold 12px "PingFang SC","Microsoft YaHei",sans-serif';
                const labelText = '⚠危险点';
                c.fillText(labelText, dangerX + 14, dangerY - 4);

                // 危险点应力状态小单元体
                const unitX = dangerX + 55;
                const unitY = dangerY - 30;
                const unitSize = 22;
                drawStressElement(c, unitX, unitY, unitSize, stresses.sigma_danger,
stresses.tau_danger);

                // 底部危险点（也标记，但通常顶部更危险）
                const dangerX2 = cx;
                const dangerY2 = cy + radius;
                if (Math.abs(stresses.sigma_bottom) > 5 || stresses.tau_danger > 5) {
                    c.fillStyle = 'rgba(255,140,80,0.7)';
                    c.strokeStyle = 'rgba(255,255,255,0.6)';
                    c.lineWidth = 2;
                    c.beginPath();
                    c.arc(dangerX2, dangerY2, 5, 0, Math.PI * 2);
                    c.fill();
                    c.stroke();
                    c.fillStyle = 'rgba(255,255,255,0.8)';
                    c.font = '10px "PingFang SC","Microsoft YaHei",sans-serif';
                    c.fillText('σ=' + stresses.sigma_bottom.toFixed(1) + 'MPa', dangerX2 + 10,
dangerY2 + 4);
```

```
                }

                // 正应力分布指示条（右侧）
                const barX = cx + radius + 42;
                const barW = 18;
                const barH = radius * 2;
                const barY = cy - radius;
                const barGrad = c.createLinearGradient(barX, barY, barX, barY + barH);
                for (let i = 0; i <= numStops; i++) {
                    const frac = i / numStops;
                    const yPhys = r - frac * 2 * r;
                    const sigmaAtY = stresses.sigma_F + (stresses.sigma_bend_max * yPhys /
r);
                    barGrad.addColorStop(frac, stressToColor(sigmaAtY, absMaxStress));
                }
                c.fillStyle = barGrad;
                c.fillRect(barX, barY, barW, barH);
                c.strokeStyle = '#8899bb';
                c.lineWidth = 1.5;
                c.strokeRect(barX, barY, barW, barH);
                // 标注
                c.fillStyle = '#fff';
                c.font = 'bold 10px "PingFang SC","Microsoft YaHei",sans-serif';
                c.fillText('σ', barX + barW / 2 - 5, barY - 8);
                c.fillText('拉', barX + barW + 4, barY + 14);
                c.fillText('压', barX + barW + 4, barY + barH - 4);
                c.fillText('+', barX + barW / 2 - 2, barY + barH / 2 - 2);
                // 刻度
                c.fillStyle = 'rgba(255,255,255,0.6)';
                c.font = '9px "PingFang SC","Microsoft YaHei",sans-serif';
                c.fillText((sigma_top > 0 ? '+' : '') + sigma_top.toFixed(1), barX + barW + 3, barY
+ 10);
                c.fillText((sigma_bottom > 0 ? '+' : '') + sigma_bottom.toFixed(1), barX + barW
+ 3, barY + barH - 3);

                // 标题
                c.fillStyle = 'rgba(255,255,255,0.5)';
                c.font = '10px "PingFang SC","Microsoft YaHei",sans-serif';
                c.fillText('正应力分布', cx - 30, cy - radius - 16);
            }

            function stressToColor(sigma, absMax) {
                // 将应力值映射到颜色：拉应力红色，压应力蓝色，零应力白色
                if (absMax < 0.01) return 'rgba(255,255,255,0.9)';
```

```
            const t = Math.max(-1, Math.min(1, sigma / absMax));
            if (t > 0) {
                // 拉应力：白色→浅红→深红
                const r = 255;
                const g = Math.round(255 - t * 200);
                const b = Math.round(255 - t * 220);
                return `rgb(${r},${g},${b})`;
            } else if (t < 0) {
                // 压应力：白色→浅蓝→深蓝
                const absT = Math.abs(t);
                const r = Math.round(255 - absT * 220);
                const g = Math.round(255 - absT * 180);
                const b = 255;
                return `rgb(${r},${g},${b})`;
            } else {
                return 'rgba(255,255,255,0.9)';
            }
        }

        function drawStressElement(ctx, cx, cy, size, sigma, tau) {
            // 绘制应力单元体
            const s = size;
            const hw = s;
            const hh = s;

            // 单元体背景
            ctx.fillStyle = 'rgba(20,22,28,0.85)';
            ctx.strokeStyle = 'rgba(255,255,255,0.6)';
            ctx.lineWidth = 1.5;
            ctx.fillRect(cx - hw, cy - hh, hw * 2, hh * 2);
            ctx.strokeRect(cx - hw, cy - hh, hw * 2, hh * 2);

            // 正应力箭头（垂直方向）
            if (Math.abs(sigma) > 0.5) {
                const sigmaDir = sigma > 0 ? -1 : 1; // 拉应力向外，压应力向内
                // 上下方向的正应力
                drawStressArrow(ctx, cx, cy - hh, cx, cy - hh - 10, sigma > 0 ? 'up' : 'down',
'#e06c75');
                drawStressArrow(ctx, cx, cy + hh, cx, cy + hh + 10, sigma > 0 ? 'down' : 'up',
'#e06c75');
            }

            // 切应力箭头（水平方向，在上下边缘）
            if (Math.abs(tau) > 0.3) {
```

```
                const tauDir = tau > 0 ? 'right' : 'left';
                drawStressArrow(ctx, cx - hw, cy - hh, cx - hw - 8, cy - hh, tauDir, '#d19a66');
                drawStressArrow(ctx, cx + hw, cy - hh, cx + hw + 8, cy - hh, tauDir ===
'right' ? 'left' : 'right',
                    '#d19a66');
                drawStressArrow(ctx, cx - hw, cy + hh, cx - hw - 8, cy + hh, tauDir === 'right' ?
'left' : 'right',
                    '#d19a66');
                drawStressArrow(ctx, cx + hw, cy + hh, cx + hw + 8, cy + hh, tauDir,
'#d19a66');
            }

            // 标签
            ctx.fillStyle = '#fff';
            ctx.font = 'bold 11px "PingFang SC","Microsoft YaHei",sans-serif';
            ctx.fillText(' σ =' + sigma.toFixed(1), cx - hw - 6, cy - hh - 14);
            ctx.fillText(' τ =' + tau.toFixed(1), cx - hw - 6, cy + hh + 20);
        }

        function drawStressArrow(ctx, x1, y1, x2, y2, direction, color) {
            ctx.strokeStyle = color;
            ctx.fillStyle = color;
            ctx.lineWidth = 2;
            ctx.lineCap = 'round';
            ctx.beginPath();
            ctx.moveTo(x1, y1);
            ctx.lineTo(x2, y2);
            ctx.stroke();
            // 小箭头尖
            const dx = x2 - x1;
            const dy = y2 - y1;
            const len = Math.sqrt(dx * dx + dy * dy);
            if (len < 1) return;
            const ux = dx / len;
            const uy = dy / len;
            const tipSize = 5;
            ctx.beginPath();
            ctx.moveTo(x2, y2);
            ctx.lineTo(x2 - ux * tipSize + uy * tipSize * 0.6, y2 - uy * tipSize - ux * tipSize *
0.6);
            ctx.lineTo(x2 - ux * tipSize - uy * tipSize * 0.6, y2 - uy * tipSize + ux * tipSize *
0.6);
            ctx.closePath();
            ctx.fill();
```

```
        }

        // ============ 更新公式显示 ============
        function updateFormulaDisplay() {
            const stresses = calcStresses();
            document.getElementById('dispSigma').textContent = stresses.sigma_danger.toFixed(2);
            document.getElementById('dispTau').textContent = stresses.tau_danger.toFixed(2);
            document.getElementById('dispSigmaR3').textContent = stresses.sigma_r3.toFixed(2);
            document.getElementById('dispSigmaR4').textContent = stresses.sigma_r4.toFixed(2);
            document.getElementById('dispSigmaMax').textContent = stresses.sigma_max_abs.toFixed(2);

            // 中性轴
            if (stresses.y0 !== null && Math.abs(stresses.M) > 1e-9) {
                document.getElementById('dispNeutralAxis').textContent = stresses.y0.toFixed(2);
                if (Math.abs(stresses.y0) <= r) {
                    document.getElementById('neutralNote').textContent =
                        '✓ 中性轴在截面内，截面部分受拉、部分受压';
                    document.getElementById('neutralNote').style.color = '#ffdd57';
                } else {
                    document.getElementById('neutralNote').textContent =
                        '中性轴在截面外，整个截面同号（全受' + (stresses.y0 > 0 ? '压' : '拉') + '）';
                    document.getElementById('neutralNote').style.color = '#8b95a8';
                }
            } else if (Math.abs(stresses.M) < 1e-9) {
                document.getElementById('dispNeutralAxis').textContent = '∞（无弯曲）';
                document.getElementById('neutralNote').textContent = '纯轴力状态，正应力均匀分布';
                document.getElementById('neutralNote').style.color = '#8b95a8';
            } else {
                document.getElementById('dispNeutralAxis').textContent = stresses.y0 !== null ? stresses.y0.toFixed(
                    2) : '--';
            }

            // 强度校核
            const checkElem = (id, value, allow) => {
```

```
            const el = document.getElementById(id);
            if (value <= allow) {
                el.textContent = '✓ 安全';
                el.className = 'check-pass';
            } else {
                el.textContent = '✗ 超限';
                el.className = 'check-fail';
            }
        };
        checkElem('checkR3', stresses.sigma_r3, sigmaAllow);
        checkElem('checkR4', stresses.sigma_r4, sigmaAllow);
        checkElem('checkSigmaMax', stresses.sigma_max_abs, sigmaAllow);

        // 更新危险点公式中的数值
        document.getElementById('dangerPointFormula').innerHTML =
            '危险点(截面顶部)： <span class="hl">σ</span>=<span id="dispSigma">' + stresses.sigma_danger.toFixed(
                2) + '</span> MPa, ' +
            '<span class="hl">τ</span>=<span id="dispTau">' + stresses.tau_danger.toFixed(2) + '</span> MPa';
    }

    // ============ 主渲染循环 ============
    function render() {
        drawBeamDiagram();
        drawSectionDiagram();
        updateFormulaDisplay();
        requestAnimationFrame(() => {
            // 仅在有动画需求时持续渲染（危险点脉冲）
            if (document.hidden) return;
            // 每 3 帧更新一次截面图（因为有脉冲动画）
        });
    }

    let animFrameId;
    let frameCount = 0;

    function animateLoop() {
        frameCount++;
        if (frameCount % 4 === 0) {
            drawSectionDiagram();
        }
        if (frameCount % 8 === 0) {
            drawBeamDiagram();
```

```
        }
        if (frameCount % 6 === 0) {
            updateFormulaDisplay();
        }
        animFrameId = requestAnimationFrame(animateLoop);
}

function fullRender() {
        drawBeamDiagram();
        drawSectionDiagram();
        updateFormulaDisplay();
}

// ============ 事件处理 ============
function onSliderChange() {
        F_kN = parseFloat(sliderF.value);
        M_kNm = parseFloat(sliderM.value);
        T_kNm = parseFloat(sliderT.value);
        sigmaAllow = parseFloat(sliderSigmaAllow.value);

        valF.textContent = F_kN.toFixed(0);
        valM.textContent = M_kNm.toFixed(2);
        valT.textContent = T_kNm.toFixed(2);
        valSigmaAllow.textContent = sigmaAllow.toFixed(0);

        fullRender();
}

sliderF.addEventListener('input', onSliderChange);
sliderM.addEventListener('input', onSliderChange);
sliderT.addEventListener('input', onSliderChange);
sliderSigmaAllow.addEventListener('input', onSliderChange);

// 模式切换
modeBtns.forEach(btn => {
        btn.addEventListener('click', function() {
            modeBtns.forEach(b => b.classList.remove('active'));
            this.classList.add('active');
            mode = this.dataset.mode;

            // 根据模式调整滑块
            if (mode === 'tension-bend') {
                sliderT.value = 0;
                T_kNm = 0;
```

```
            valT.textContent = '0.00';
            sliderT.disabled = true;
            sliderF.disabled = false;
            sliderM.disabled = false;
            sliderT.style.opacity = '0.35';
            sliderF.style.opacity = '1';
            sliderM.style.opacity = '1';
        } else if (mode === 'bend-torsion') {
            sliderF.value = 0;
            F_kN = 0;
            valF.textContent = '0';
            sliderF.disabled = true;
            sliderT.disabled = false;
            sliderM.disabled = false;
            sliderF.style.opacity = '0.35';
            sliderT.style.opacity = '1';
            sliderM.style.opacity = '1';
        } else {
            sliderF.disabled = false;
            sliderM.disabled = false;
            sliderT.disabled = false;
            sliderF.style.opacity = '1';
            sliderM.style.opacity = '1';
            sliderT.style.opacity = '1';
            // 恢复默认值
            if (parseFloat(sliderF.value) < 1) sliderF.value = 80;
            if (parseFloat(sliderT.value) < 0.01) sliderT.value = 1.5;
            F_kN = parseFloat(sliderF.value);
            T_kNm = parseFloat(sliderT.value);
            valF.textContent = F_kN.toFixed(0);
            valT.textContent = T_kNm.toFixed(2);
        }
        M_kNm = parseFloat(sliderM.value);
        valM.textContent = M_kNm.toFixed(2);
        sigmaAllow = parseFloat(sliderSigmaAllow.value);
        fullRender();
    });
});

// 初始化
function init() {
    valF.textContent = F_kN.toFixed(0);
    valM.textContent = M_kNm.toFixed(2);
    valT.textContent = T_kNm.toFixed(2);
```

```
            valSigmaAllow.textContent = sigmaAllow.toFixed(0);
            fullRender();
            animFrameId = requestAnimationFrame(animateLoop);
        }

        // 页面可见性变化时处理动画
        document.addEventListener('visibilitychange', () => {
            if (document.hidden) {
                if (animFrameId) cancelAnimationFrame(animFrameId);
            } else {
                frameCount = 0;
                animFrameId = requestAnimationFrame(animateLoop);
            }
        });

        // 窗口大小变化时重新渲染
        window.addEventListener('resize', () => {
            fullRender();
        });

        init();

        console.log('✅ 材料力学 - 组合变形与叠加原理交互演示已就绪');
        console.log('   🟡 圆形截面 d=50mm | A=' + A.toFixed(1) + 'mm² | W=' +
W.toFixed(0) + 'mm³ | W_t=' + Wt.toFixed(
            0) + 'mm³ ');
        console.log('   🔴 轴力 F: 0~200 kN (均匀正应力)');
        console.log('   🔵 弯矩 M: 0~5 kN·m (弯曲正应力，线性分布)');
        console.log('   🟠 扭矩 T: 0~5 kN·m (扭转切应力，径向分布)');
        console.log('   📐 危险点：截面顶部边缘（正应力+切应力均最大）');
        console.log('   📏 强度理论：第三强度理论 σ_r3=√(σ²+4τ²) | 第四强度
理论 σ_r4=√(σ²+3τ²)');
        console.log('   🎯 拖动滑块实时观察应力变化 | 支持拉弯/弯扭/全载荷三
种模式');
    })();
</script>
</body>
</html>
```