```
<!DOCTYPE html>
<html lang="zh-CN">
<head>
    <meta charset="UTF-8">
    <meta name="viewport" content="width=device-width, initial-scale=1.0">
    <title>弯曲变形 • 积分法求位移 • 教学演示</title>
    <style>
        /* ----------  全局重置  ---------- */
        * {
            margin: 0;
            padding: 0;
            box-sizing: border-box;
        }
        body {
            font-family: 'Segoe UI', 'PingFang SC', Roboto, 'Helvetica Neue', sans-serif;
            background: #f5f9fc;
            color: #1e2a3a;
            padding: 20px;
            min-height: 100vh;
            display: flex;
            justify-content: center;
            align-items: center;
        }
        .container {
            max-width: 1300px;
            width: 100%;
            background: #ffffff;
            border-radius: 24px;
            box-shadow: 0 20px 60px rgba(0, 20, 40, 0.12);
            padding: 28px 32px 36px;
            transition: all 0.2s;
        }

        /* ----------  头部  ---------- */
        .header {
            display: flex;
            justify-content: space-between;
            align-items: flex-end;
            flex-wrap: wrap;
            margin-bottom: 22px;
            border-bottom: 2px solid #e8edf3;
            padding-bottom: 16px;
        }
        .header-left h1 {
```

```
        font-size: 26px;
        font-weight: 600;
        letter-spacing: 0.5px;
        color: #0b2b44;
    }
    .header-left h1 small {
        font-size: 16px;
        font-weight: 400;
        color: #4e6b85;
        margin-left: 14px;
    }
    .header-left .sub {
        font-size: 14px;
        color: #5d7a96;
        margin-top: 2px;
    }
    .header-right {
        display: flex;
        gap: 12px;
        align-items: center;
        flex-wrap: wrap;
    }
    .badge {
        background: #e2eaf2;
        border-radius: 40px;
        padding: 4px 16px;
        font-size: 13px;
        color: #1e3b5a;
        font-weight: 500;
    }

    /* ----------  控制面板  ---------- */
    .controls {
        display: flex;
        flex-wrap: wrap;
        gap: 18px 32px;
        background: #f2f6fb;
        border-radius: 18px;
        padding: 14px 24px;
        margin-bottom: 22px;
        align-items: center;
        border: 1px solid #e2eaf2;
    }
    .control-group {
```

```
        display: flex;
        align-items: center;
        gap: 8px 14px;
        flex-wrap: wrap;
    }
    .control-group label {
        font-size: 14px;
        font-weight: 500;
        color: #1e3b5a;
        display: flex;
        align-items: center;
        gap: 4px;
    }
    .control-group label .unit {
        font-weight: 400;
        color: #5d7a96;
        font-size: 12px;
    }
    .control-group input[type="range"] {
        width: 140px;
        height: 4px;
        -webkit-appearance: none;
        background: #c8d6e5;
        border-radius: 4px;
        outline: none;
        transition: background 0.2s;
    }
    .control-group input[type="range"]::-webkit-slider-thumb {
        -webkit-appearance: none;
        width: 16px;
        height: 16px;
        border-radius: 50%;
        background: #1a6c9e;
        cursor: pointer;
        border: 2px solid white;
        box-shadow: 0 2px 6px rgba(0, 20, 40, 0.2);
    }
    .control-group input[type="range"]::-moz-range-thumb {
        width: 16px;
        height: 16px;
        border-radius: 50%;
        background: #1a6c9e;
        cursor: pointer;
        border: 2px solid white;
```

```
        box-shadow: 0 2px 6px rgba(0, 20, 40, 0.2);
}
.control-group .val {
        min-width: 48px;
        font-weight: 600;
        color: #0b2b44;
        font-size: 15px;
        text-align: center;
}
.control-group select {
        padding: 4px 12px;
        border-radius: 8px;
        border: 1px solid #c8d6e5;
        background: white;
        font-size: 14px;
        font-weight: 500;
        color: #1e3b5a;
        cursor: pointer;
        outline: none;
}
.control-group select:focus {
        border-color: #1a6c9e;
}
.btn-reset {
        background: #dce6ef;
        border: none;
        border-radius: 8px;
        padding: 6px 18px;
        font-size: 13px;
        font-weight: 500;
        color: #1e3b5a;
        cursor: pointer;
        transition: background 0.2s;
}
.btn-reset:hover {
        background: #c8d6e5;
}

/* ----------  画布区域  ---------- */
.canvas-main-wrap {
        position: relative;
        background: #fafcfe;
        border-radius: 18px;
        border: 1px solid #e2eaf2;
```

```
        overflow: hidden;
        margin-bottom: 18px;
}
.canvas-main-wrap canvas {
        display: block;
        width: 100%;
        height: auto;
        background: #fafcfe;
}

/* ----------  三图并排  ---------- */
.charts-row {
        display: grid;
        grid-template-columns: 1fr 1fr 1fr;
        gap: 14px;
        margin-bottom: 18px;
}
.chart-card {
        background: #fafcfe;
        border-radius: 16px;
        border: 1px solid #e2eaf2;
        overflow: hidden;
        padding: 6px 6px 2px;
        transition: border-color 0.2s;
}
.chart-card:hover {
        border-color: #b8cddf;
}
.chart-card .chart-title {
        font-size: 13px;
        font-weight: 600;
        color: #1e3b5a;
        text-align: center;
        padding: 4px 0 2px;
        letter-spacing: 0.3px;
}
.chart-card canvas {
        display: block;
        width: 100%;
        height: auto;
        background: #fafcfe;
        border-radius: 8px;
}
```

```
/* ----------  积分过程 & 关键值  ---------- */
.footer-info {
    display: grid;
    grid-template-columns: 2fr 1fr;
    gap: 18px;
    background: #f2f6fb;
    border-radius: 18px;
    padding: 16px 22px;
    border: 1px solid #e2eaf2;
}
.formula-box {
    font-size: 14px;
    line-height: 1.7;
    color: #1a2f44;
}
.formula-box .math {
    font-family: 'Times New Roman', 'CMU Serif', serif;
    font-style: italic;
    color: #0b2b44;
}
.formula-box .highlight {
    background: #dce9f5;
    padding: 1px 8px;
    border-radius: 6px;
    font-weight: 500;
}
.formula-box .step {
    display: inline-block;
    margin-right: 16px;
    font-weight: 500;
    color: #1a6c9e;
}
.values-box {
    display: flex;
    flex-direction: column;
    gap: 4px;
    justify-content: center;
    font-size: 14px;
}
.values-box .item {
    display: flex;
    justify-content: space-between;
    padding: 2px 0;
    border-bottom: 1px dashed #dce6ef;
```

```
}
.values-box .item:last-child {
    border-bottom: none;
}
.values-box .label {
    color: #3e5f7a;
}
.values-box .number {
    font-weight: 600;
    color: #0b2b44;
    font-family: 'Courier New', monospace;
}

/* ----------  响应式  ---------- */
@media (max-width: 880px) {
    .container {
        padding: 16px;
    }
    .controls {
        flex-direction: column;
        align-items: stretch;
        gap: 10px;
    }
    .control-group {
        justify-content: space-between;
    }
    .control-group input[type="range"] {
        flex: 1;
        min-width: 80px;
    }
    .charts-row {
        grid-template-columns: 1fr;
        gap: 12px;
    }
    .footer-info {
        grid-template-columns: 1fr;
        gap: 14px;
    }
    .header {
        flex-direction: column;
        align-items: flex-start;
        gap: 8px;
    }
}
```

```
        @media (max-width: 480px) {
            .control-group input[type="range"] {
                width: 100px;
            }
            .header-left h1 {
                font-size: 20px;
            }
            .header-left h1 small {
                font-size: 13px;
                display: block;
                margin-left: 0;
            }
        }

        /* ----------  工具类  ---------- */
        .text-muted {
            color: #6a869f;
            font-size: 13px;
        }
        .mt-1 {
            margin-top: 6px;
        }
    </style>
</head>
<body>

    <div class="container">

        <!-- ======  头部  ====== -->
        <div class="header">
            <div class="header-left">
                <h1>
                    📐 弯曲变形 • 积分法求位移
                    <small>悬臂梁</small>
                </h1>
                <div class="sub">材料力学 • 交互式教学演示  |  拖动滑块实时观察变形</div>
            </div>
            <div class="header-right">
                <span class="badge">⚡ 实时更新</span>
                <span class="badge">📐 积分法</span>
            </div>
        </div>
```

```
        <!-- ====== 控制面板 ====== -->
        <div class="controls" id="controls">
            <!-- 载荷类型 -->
            <div class="control-group">
                <label>载荷类型</label>
                <select id="loadType">
                    <option value="concentrated">集中力 (自由端)</option>
                    <option value="uniform">均布载荷</option>
                </select>
            </div>

            <!-- 载荷大小 -->
            <div class="control-group">
                <label>载荷 <span class="unit" id="loadUnitLabel">P</span></label>
                <input type="range" id="loadVal" min="1" max="20" step="0.5" value="10">
                <span class="val" id="loadDisplay">10.0</span>
            </div>

            <!-- 梁长 -->
            <div class="control-group">
                <label>梁长 <span class="unit">L</span></label>
                <input type="range" id="lengthVal" min="0.8" max="2.5" step="0.05"
value="1.6">
                <span class="val" id="lengthDisplay">1.60</span>
            </div>

            <!-- 抗弯刚度 EI -->
            <div class="control-group">
                <label>抗弯刚度 <span class="unit">EI</span></label>
                <input type="range" id="eiVal" min="0.5" max="5.0" step="0.1" value="2.0">
                <span class="val" id="eiDisplay">2.00</span>
            </div>

            <button class="btn-reset" id="resetBtn">↺ 重置</button>
        </div>

        <!-- ====== 主画布：梁的变形 ====== -->
        <div class="canvas-main-wrap">
            <canvas id="mainCanvas" width="1000" height="380"></canvas>
        </div>

        <!-- ====== 三图：弯矩 • 转角 • 挠度 ====== -->
        <div class="charts-row">
            <div class="chart-card">
```

```
                <div class="chart-title">📉 弯矩图 M(x)</div>
                <canvas id="momentCanvas" width="400" height="160"></canvas>
            </div>
            <div class="chart-card">
                <div class="chart-title">📈 转角图 θ(x)</div>
                <canvas id="thetaCanvas" width="400" height="160"></canvas>
            </div>
            <div class="chart-card">
                <div class="chart-title">📊 挠度图 v(x)</div>
                <canvas id="deflectCanvas" width="400" height="160"></canvas>
            </div>
        </div>

        <!-- ====== 底部：积分过程 + 关键数值 ====== -->
        <div class="footer-info">
            <div class="formula-box" id="formulaBox">
                <div style="font-weight:600; margin-bottom:4px;">📝 积分法过程</div>
                <div id="formulaContent">
                    <span class="step">①</span> 弯矩方程   <span class="math" id="fM">M(x) = −P(L−x)</span><br>
                    <span class="step">②</span> 转角方程   <span class="math" id="fTheta">θ(x) = ∫ M(x)/EI dx + C₁ </span><br>
                    <span class="step">③</span> 挠度方程   <span class="math" id="fV">v(x) = ∫ θ(x) dx + C₂ </span><br>
                    <span class="step">④ </span> 边界条件   <span class="highlight" id="fBC">x=0: θ =0, v=0</span>   →   <span class="math" id="fConst">C₁ =0, C₂ =0</span>
                </div>
            </div>
            <div class="values-box" id="valuesBox">
                <div class="item"><span class="label">最大挠度 v<sub>max</sub></span><span class="number" id="vmaxDisplay">—</span></div>
                <div class="item"><span class="label">最大转角 θ<sub>max</sub></span><span class="number" id="thetamaxDisplay">—</span></div>
                <div class="item"><span class="label">固定端弯矩 |M<sub>0</sub>|</span><span class="number" id="m0Display">—</span></div>
                <div class="item"><span class="label">自由端挠度 v(L)</span><span class="number" id="vLDisplay">—</span></div>
            </div>
        </div>

    </div>

    <script>
```

```
// ============================================================
//   核心脚本
// ============================================================

// ---------- DOM 引用 ----------
const mainCanvas = document.getElementById('mainCanvas');
const ctxMain = mainCanvas.getContext('2d');
const momentCanvas = document.getElementById('momentCanvas');
const ctxMom = momentCanvas.getContext('2d');
const thetaCanvas = document.getElementById('thetaCanvas');
const ctxTheta = thetaCanvas.getContext('2d');
const deflectCanvas = document.getElementById('deflectCanvas');
const ctxDef = deflectCanvas.getContext('2d');

const loadType = document.getElementById('loadType');
const loadVal = document.getElementById('loadVal');
const lengthVal = document.getElementById('lengthVal');
const eiVal = document.getElementById('eiVal');
const loadDisplay = document.getElementById('loadDisplay');
const lengthDisplay = document.getElementById('lengthDisplay');
const eiDisplay = document.getElementById('eiDisplay');
const loadUnitLabel = document.getElementById('loadUnitLabel');

const fM = document.getElementById('fM');
const fTheta = document.getElementById('fTheta');
const fV = document.getElementById('fV');
const fBC = document.getElementById('fBC');
const fConst = document.getElementById('fConst');

const vmaxDisplay = document.getElementById('vmaxDisplay');
const thetamaxDisplay = document.getElementById('thetamaxDisplay');
const m0Display = document.getElementById('m0Display');
const vLDisplay = document.getElementById('vLDisplay');

const resetBtn = document.getElementById('resetBtn');

// ----------  状态  ----------
let params = {
    loadType: 'concentrated', // 'concentrated' | 'uniform'
    P: 10.0, //  集中力 (N) 或 均布载荷 (N/m)
    L: 1.6, //  梁长 (m)
    EI: 2.0, //  抗弯刚度 (N • m² )
};
```

```
        // ---------- 辅助函数 ----------
        function clamp(v, min, max) { return Math.min(Math.max(v, min), max); }

        // ---------- 更新显示数值 ----------
        function updateDisplay() {
            loadDisplay.textContent = params.P.toFixed(1);
            lengthDisplay.textContent = params.L.toFixed(2);
            eiDisplay.textContent = params.EI.toFixed(2);
            loadUnitLabel.textContent = params.loadType === 'concentrated' ? 'P (N)' : 'q
(N/m)';
        }

        // ---------- 从控件读取参数 ----------
        function readParams() {
            params.loadType = loadType.value;
            params.P = parseFloat(loadVal.value);
            params.L = parseFloat(lengthVal.value);
            params.EI = parseFloat(eiVal.value);
            updateDisplay();
        }

        // ---------- 解析公式 & 关键值 ----------
        function computeFormulas() {
            const { loadType, P, L, EI } = params;
            const isConc = loadType === 'concentrated';

            // 弯矩 M(x)    (向下为正)
            let M_expr, M_func, theta_expr, theta_func, v_expr, v_func;
            let v_max, theta_max, M0, vL;

            if (isConc) {
                // 集中力在自由端
                // M(x) = -P(L-x)    (符号：上侧受拉为负)
                //  θ(x) = P/(2EI)(2Lx - x²)
                // v(x) = P/(6EI)(3L² x - 3Lx²  + x³)
                M_expr = `M(x) = −P • (L−x)`;
                M_func = (x) => -P * (L - x);
                theta_expr = `θ(x) = P/(2EI) • (2Lx − x²)`;
                theta_func = (x) => (P / (2 * EI)) * (2 * L * x - x * x);
                v_expr = `v(x) = P/(6EI) • (3L² x − 3Lx²  + x³)`;
                v_func = (x) => (P / (6 * EI)) * (3 * L * L * x - 3 * L * x * x + x * x * x);

                v_max = v_func(L);
                theta_max = theta_func(L);
```

```
                M0 = Math.abs(M_func(0));
                vL = v_func(L);
            } else {
                // 均布载荷 q
                // M(x) = -q(L-x)² /2
                // θ(x) = q/(6EI)(3L² x - 3Lx² + x³)
                // v(x) = q/(24EI)(6L² x² - 4Lx³ + x⁴)
                M_expr = `M(x) = −q • (L−x)² /2`;
                M_func = (x) => -P * (L - x) * (L - x) / 2;
                theta_expr = `θ(x) = q/(6EI) • (3L² x − 3Lx² + x³)`;
                theta_func = (x) => (P / (6 * EI)) * (3 * L * L * x - 3 * L * x * x + x * x * x);
                v_expr = `v(x) = q/(24EI) • (6L² x² − 4Lx³ + x⁴)`;
                v_func = (x) => (P / (24 * EI)) * (6 * L * L * x * x - 4 * L * x * x * x + x * x * x * x);

                v_max = v_func(L);
                theta_max = theta_func(L);
                M0 = Math.abs(M_func(0));
                vL = v_func(L);
            }

            return { M_expr, M_func, theta_expr, theta_func, v_expr, v_func, v_max, theta_max, M0, vL };
        }

        // ---------- 更新公式显示 ----------
        function updateFormulas() {
            const { loadType, P, L, EI } = params;
            const isConc = loadType === 'concentrated';
            const sym = isConc ? 'P' : 'q';
            const val = P;

            let M_str, theta_str, v_str, bc_str, const_str;

            if (isConc) {
                M_str = `M(x) = −${val.toFixed(1)} • (${L.toFixed(2)}−x)`;
                theta_str = `θ(x) = ${val.toFixed(1)}/(2 •${EI.toFixed(2)}) •(2 •${L.toFixed(2)}x − x²)`;
                v_str = `v(x) = ${val.toFixed(1)}/(6 • ${EI.toFixed(2)}) • (3 • ${L.toFixed(2)}² x − 3 • ${L.toFixed(2)}x² + x³)`;
            } else {
                M_str = `M(x) = −${val.toFixed(1)} • (${L.toFixed(2)}−x)² /2`;
                theta_str = `θ(x) = ${val.toFixed(1)}/(6 • ${EI.toFixed(2)}) • (3 • ${L.toFixed(2)}² x − 3 • ${L.toFixed(2)}x² + x³)`;
```

```
                v_str = `v(x) = ${val.toFixed(1)}/(24 • ${EI.toFixed(2)}) • (6 • ${L.toFixed(2)}² x²
− 4 • ${L.toFixed(2)}x³ + x⁴ )`;
            }

            fM.textContent = M_str;
            fTheta.textContent = theta_str;
            fV.textContent = v_str;
            fBC.textContent = `x=0: θ=0, v=0 (固定端)`;
            fConst.textContent = `C₁ =0, C₂ =0`;

            // 关键值
            const formulas = computeFormulas();
            vmaxDisplay.textContent = formulas.v_max.toFixed(4);
            thetamaxDisplay.textContent = formulas.theta_max.toFixed(4);
            m0Display.textContent = formulas.M0.toFixed(2);
            vLDisplay.textContent = formulas.vL.toFixed(4);
        }

        // ============================================================
        //  绘图函数
        // ============================================================

        // ---------- 主画布：梁的变形 ----------
        function drawMain() {
            const W = mainCanvas.width;
            const H = mainCanvas.height;
            const ctx = ctxMain;
            ctx.clearRect(0, 0, W, H);

            const { loadType, P, L, EI } = params;
            const isConc = loadType === 'concentrated';

            // 布局参数
            const margin = { top: 40, bottom: 50, left: 70, right: 50 };
            const drawW = W - margin.left - margin.right;
            const drawH = H - margin.top - margin.bottom;
            const x0 = margin.left;
            const y0 = margin.top + drawH * 0.65; // 梁的基线位置

            // 缩放
            const scaleX = drawW / L;

            // 计算最大挠度用于缩放变形
            const formulas = computeFormulas();
```

```
let maxDeflect = Math.abs(formulas.v_max);
if (maxDeflect < 1e-12) maxDeflect = 1e-6;
// 变形放大系数：让最大挠度在图上显示约 40~60px
const targetPx = Math.min(drawH * 0.35, 70);
const scaleY = targetPx / maxDeflect;

// 绘制网格 / 参考线
ctx.save();
ctx.strokeStyle = "#dce6ef";
ctx.lineWidth = 0.5;
ctx.setLineDash([4, 6]);
// 水平基线
ctx.beginPath();
ctx.moveTo(x0, y0);
ctx.lineTo(x0 + drawW, y0);
ctx.stroke();
ctx.setLineDash([]);

// 绘制原始梁 (灰色)
ctx.strokeStyle = "#a0b8cc";
ctx.lineWidth = 2.5;
ctx.beginPath();
ctx.moveTo(x0, y0);
ctx.lineTo(x0 + drawW, y0);
ctx.stroke();

// 绘制变形梁 (蓝色)
const steps = 200;
ctx.strokeStyle = "#1a6c9e";
ctx.lineWidth = 3.5;
ctx.beginPath();
let first = true;
for (let i = 0; i <= steps; i++) {
    const t = i / steps;
    const x = t * L;
    const v = (isConc ?
        (P / (6 * EI)) * (3 * L * L * x - 3 * L * x * x + x * x * x) :
        (P / (24 * EI)) * (6 * L * L * x * x - 4 * L * x * x * x + x * x * x * x)
    );
    const px = x0 + t * drawW;
    const py = y0 + v * scaleY;
    if (first) { ctx.moveTo(px, py);
        first = false; } else ctx.lineTo(px, py);
}
```

```
ctx.stroke();

// 绘制固定端 (左侧)
ctx.fillStyle = "#2c3e50";
ctx.strokeStyle = "#2c3e50";
ctx.lineWidth = 2;
// 三角形阴影
const fixW = 12;
const fixH = 28;
ctx.beginPath();
ctx.moveTo(x0 - 2, y0 - fixH);
ctx.lineTo(x0 - 2, y0 + fixH);
ctx.lineTo(x0 + fixW, y0);
ctx.closePath();
ctx.fillStyle = "#d0dce8";
ctx.fill();
ctx.strokeStyle = "#2c3e50";
ctx.lineWidth = 1.8;
ctx.stroke();
// 竖线
ctx.beginPath();
ctx.moveTo(x0 - 2, y0 - fixH);
ctx.lineTo(x0 - 2, y0 + fixH);
ctx.stroke();
// 斜线
ctx.beginPath();
ctx.moveTo(x0 - 2, y0 - fixH);
ctx.lineTo(x0 + fixW, y0);
ctx.moveTo(x0 - 2, y0 + fixH);
ctx.lineTo(x0 + fixW, y0);
ctx.stroke();

// 标注 "固定端"
ctx.fillStyle = "#2c3e50";
ctx.font = "13px 'Segoe UI', sans-serif";
ctx.textAlign = "center";
ctx.fillText("固定端", x0 - 2, y0 + fixH + 22);

// 绘制载荷
const loadX = x0 + drawW; // 自由端
const loadY = y0 + (isConc ? formulas.v_max * scaleY : formulas.v_max * scaleY);
const arrowLen = Math.min(36, 20 + P * 1.2);

if (isConc) {
```

```
        // 集中力箭头向下
        const tipX = loadX;
        const tipY = loadY + arrowLen;
        ctx.fillStyle = "#d45d3a";
        ctx.strokeStyle = "#d45d3a";
        ctx.lineWidth = 2.5;
        ctx.beginPath();
        ctx.moveTo(tipX, loadY);
        ctx.lineTo(tipX, tipY - 10);
        ctx.stroke();
        // 箭头头
        ctx.beginPath();
        ctx.moveTo(tipX, tipY);
        ctx.lineTo(tipX - 8, tipY - 10);
        ctx.lineTo(tipX + 8, tipY - 10);
        ctx.closePath();
        ctx.fill();
        // 标签
        ctx.fillStyle = "#d45d3a";
        ctx.font = "bold 15px 'Segoe UI', sans-serif";
        ctx.textAlign = "center";
        ctx.fillText(`P = ${P.toFixed(1)} N`, tipX, loadY - 10);
    } else {
        // 均布载荷：画一排小箭头
        const numArrows = 10;
        for (let i = 0; i <= numArrows; i++) {
            const t = i / numArrows;
            const px = x0 + t * drawW;
            const x = t * L;
            const v = (P / (24 * EI)) * (6 * L * L * x * x - 4 * L * x * x * x + x * x * x * x);
            const py = y0 + v * scaleY;
            const len = 14 + P * 0.8;
            ctx.fillStyle = "#d45d3a";
            ctx.strokeStyle = "#d45d3a";
            ctx.lineWidth = 1.6;
            ctx.beginPath();
            ctx.moveTo(px, py);
            ctx.lineTo(px, py + len);
            ctx.stroke();
            // 小箭头
            ctx.beginPath();
            ctx.moveTo(px, py + len);
            ctx.lineTo(px - 5, py + len - 7);
            ctx.lineTo(px + 5, py + len - 7);
```

```
                ctx.closePath();
                ctx.fill();
            }
            ctx.fillStyle = "#d45d3a";
            ctx.font = "bold 14px 'Segoe UI', sans-serif";
            ctx.textAlign = "center";
            const midX = x0 + drawW / 2;
            const midV = (P / (24 * EI)) * (6 * L * L * (L / 2) * (L / 2) - 4 * L * (L / 2) * (L / 2)
* (L / 2) + (L / 2) * (
                L / 2) * (L / 2) * (L / 2));
            const midY = y0 + midV * scaleY;
            ctx.fillText(`q = ${P.toFixed(1)} N/m`, midX, midY - 18);
        }

        // 标注 "自由端"
        ctx.fillStyle = "#2c3e50";
        ctx.font = "13px 'Segoe UI', sans-serif";
        ctx.textAlign = "center";
        ctx.fillText("自由端", loadX, y0 + 38);

        // 标注梁长 L
        ctx.fillStyle = "#2c3e50";
        ctx.font = "13px 'Segoe UI', sans-serif";
        ctx.textAlign = "center";
        const labelY = y0 + 52;
        ctx.fillText(`L = ${L.toFixed(2)} m`, x0 + drawW / 2, labelY);

        // 画一个小的变形指示（标注最大挠度位置）
        ctx.fillStyle = "#1a6c9e";
        ctx.font = "12px 'Segoe UI', sans-serif";
        ctx.textAlign = "left";
        const vMaxPos = L;
        const vMaxPx = x0 + drawW;
        const vMaxY = y0 + formulas.v_max * scaleY;
        ctx.fillStyle = "#1a6c9e";
        ctx.font = "12px 'Segoe UI', sans-serif";
        ctx.textAlign = "right";
        ctx.fillText(`v_max = ${formulas.v_max.toFixed(4)} m`, vMaxPx - 6, vMaxY - 6);

        // 小标记点
        ctx.beginPath();
        ctx.arc(vMaxPx, vMaxY, 4, 0, 2 * Math.PI);
        ctx.fillStyle = "#1a6c9e";
        ctx.fill();
```

```
    ctx.restore();

    // 标题
    ctx.fillStyle = "#1e3b5a";
    ctx.font = "bold 15px 'Segoe UI', sans-serif";
    ctx.textAlign = "left";
    ctx.fillText("梁的变形 (挠度放大)", 16, 26);
    ctx.fillStyle = "#5d7a96";
    ctx.font = "12px 'Segoe UI', sans-serif";
    ctx.fillText(`(放大比例 ${scaleY.toFixed(0)}×)`, 180, 26);
}

// ---------- 绘制弯矩图 ----------
function drawMoment() {
    const canvas = momentCanvas;
    const ctx = ctxMom;
    const W = canvas.width,
        H = canvas.height;
    ctx.clearRect(0, 0, W, H);

    const { loadType, P, L, EI } = params;
    const isConc = loadType === 'concentrated';
    const formulas = computeFormulas();
    const M_func = formulas.M_func;

    // 边距
    const pad = { top: 18, bottom: 28, left: 40, right: 20 };
    const drawW = W - pad.left - pad.right;
    const drawH = H - pad.top - pad.bottom;
    const x0 = pad.left;
    const y0 = pad.top + drawH / 2;

    // 计算弯矩范围
    let maxM = 0;
    const steps = 150;
    for (let i = 0; i <= steps; i++) {
        const x = (i / steps) * L;
        const m = Math.abs(M_func(x));
        if (m > maxM) maxM = m;
    }
    if (maxM < 1e-12) maxM = 1e-6;
    const scaleY = (drawH * 0.42) / maxM;
```

```
// 坐标轴
ctx.save();
ctx.strokeStyle = "#8aa3bc";
ctx.lineWidth = 1.2;
ctx.beginPath();
ctx.moveTo(x0, y0);
ctx.lineTo(x0 + drawW, y0);
ctx.stroke();
// 零线标注
ctx.fillStyle = "#6a869f";
ctx.font = "10px sans-serif";
ctx.textAlign = "right";
ctx.fillText("0", x0 - 6, y0 + 4);

// 绘制弯矩曲线 (在零线上方为负，下方为正)
ctx.beginPath();
let first = true;
for (let i = 0; i <= steps; i++) {
    const t = i / steps;
    const x = t * L;
    const m = M_func(x);
    const px = x0 + t * drawW;
    const py = y0 - m * scaleY;
    if (first) { ctx.moveTo(px, py);
        first = false; } else ctx.lineTo(px, py);
}
ctx.strokeStyle = "#c0392b";
ctx.lineWidth = 2.8;
ctx.stroke();

// 填充区域 (零线以下为正弯矩，以上为负)
ctx.beginPath();
ctx.moveTo(x0, y0);
for (let i = 0; i <= steps; i++) {
    const t = i / steps;
    const x = t * L;
    const m = M_func(x);
    const px = x0 + t * drawW;
    const py = y0 - m * scaleY;
    ctx.lineTo(px, py);
}
ctx.lineTo(x0 + drawW, y0);
ctx.closePath();
ctx.fillStyle = "rgba(192, 57, 43, 0.12)";
```

```
        ctx.fill();

        // 标注最大值
        const maxPos = isConc ? 0 : 0;
        const mMax = M_func(maxPos);
        const pMaxX = x0;
        const pMaxY = y0 - mMax * scaleY;
        ctx.fillStyle = "#c0392b";
        ctx.font = "11px 'Segoe UI', sans-serif";
        ctx.textAlign = "left";
        ctx.fillText(`|M|m ax  = ${Math.abs(mMax).toFixed(2)}`, pMaxX + 4, pMaxY - 4);

        // 标注 "固定端" "自由端"
        ctx.fillStyle = "#4a6885";
        ctx.font = "11px 'Segoe UI', sans-serif";
        ctx.textAlign = "center";
        ctx.fillText("固定端", x0, y0 + 22);
        ctx.fillText("自由端", x0 + drawW, y0 + 22);

        ctx.restore();
}

// ---------- 绘制转角图 ----------
function drawTheta() {
        const canvas = thetaCanvas;
        const ctx = ctxTheta;
        const W = canvas.width,
              H = canvas.height;
        ctx.clearRect(0, 0, W, H);

        const { loadType, P, L, EI } = params;
        const isConc = loadType === 'concentrated';
        const formulas = computeFormulas();
        const theta_func = formulas.theta_func;

        const pad = { top: 18, bottom: 28, left: 40, right: 20 };
        const drawW = W - pad.left - pad.right;
        const drawH = H - pad.top - pad.bottom;
        const x0 = pad.left;
        const y0 = pad.top + drawH / 2;

        // 范围
        let maxT = 0;
        const steps = 150;
```

```
for (let i = 0; i <= steps; i++) {
        const x = (i / steps) * L;
        const t = Math.abs(theta_func(x));
        if (t > maxT) maxT = t;
}
if (maxT < 1e-12) maxT = 1e-6;
const scaleY = (drawH * 0.42) / maxT;

// 坐标轴
ctx.save();
ctx.strokeStyle = "#8aa3bc";
ctx.lineWidth = 1.2;
ctx.beginPath();
ctx.moveTo(x0, y0);
ctx.lineTo(x0 + drawW, y0);
ctx.stroke();
ctx.fillStyle = "#6a869f";
ctx.font = "10px sans-serif";
ctx.textAlign = "right";
ctx.fillText("0", x0 - 6, y0 + 4);

// 绘制转角曲线
ctx.beginPath();
let first = true;
for (let i = 0; i <= steps; i++) {
        const t = i / steps;
        const x = t * L;
        const th = theta_func(x);
        const px = x0 + t * drawW;
        const py = y0 - th * scaleY;
        if (first) { ctx.moveTo(px, py);
                first = false; } else ctx.lineTo(px, py);
}
ctx.strokeStyle = "#1e8449";
ctx.lineWidth = 2.8;
ctx.stroke();

// 填充
ctx.beginPath();
ctx.moveTo(x0, y0);
for (let i = 0; i <= steps; i++) {
        const t = i / steps;
        const x = t * L;
        const th = theta_func(x);
```

```
            const px = x0 + t * drawW;
            const py = y0 - th * scaleY;
            ctx.lineTo(px, py);
        }
        ctx.lineTo(x0 + drawW, y0);
        ctx.closePath();
        ctx.fillStyle = "rgba(30, 132, 73, 0.12)";
        ctx.fill();

        // 标注最大值
        const maxPos = L;
        const thMax = theta_func(maxPos);
        const pX = x0 + drawW;
        const pY = y0 - thMax * scaleY;
        ctx.fillStyle = "#1e8449";
        ctx.font = "11px 'Segoe UI', sans-serif";
        ctx.textAlign = "right";
        ctx.fillText(`θmax = ${thMax.toFixed(4)}`, pX - 4, pY - 4);

        // 端点标注
        ctx.fillStyle = "#4a6885";
        ctx.font = "11px 'Segoe UI', sans-serif";
        ctx.textAlign = "center";
        ctx.fillText("固定端", x0, y0 + 22);
        ctx.fillText("自由端", x0 + drawW, y0 + 22);

        ctx.restore();
    }

    // ---------- 绘制挠度图 ----------
    function drawDeflect() {
        const canvas = deflectCanvas;
        const ctx = ctxDef;
        const W = canvas.width,
              H = canvas.height;
        ctx.clearRect(0, 0, W, H);

        const { loadType, P, L, EI } = params;
        const isConc = loadType === 'concentrated';
        const formulas = computeFormulas();
        const v_func = formulas.v_func;

        const pad = { top: 18, bottom: 28, left: 40, right: 20 };
        const drawW = W - pad.left - pad.right;
```

```
const drawH = H - pad.top - pad.bottom;
const x0 = pad.left;
const y0 = pad.top + drawH / 2;

// 范围
let maxV = 0;
const steps = 150;
for (let i = 0; i <= steps; i++) {
	const x = (i / steps) * L;
	const v = Math.abs(v_func(x));
	if (v > maxV) maxV = v;
}
if (maxV < 1e-12) maxV = 1e-6;
const scaleY = (drawH * 0.42) / maxV;

// 坐标轴
ctx.save();
ctx.strokeStyle = "#8aa3bc";
ctx.lineWidth = 1.2;
ctx.beginPath();
ctx.moveTo(x0, y0);
ctx.lineTo(x0 + drawW, y0);
ctx.stroke();
ctx.fillStyle = "#6a869f";
ctx.font = "10px sans-serif";
ctx.textAlign = "right";
ctx.fillText("0", x0 - 6, y0 + 4);

// 绘制挠度曲线
ctx.beginPath();
let first = true;
for (let i = 0; i <= steps; i++) {
	const t = i / steps;
	const x = t * L;
	const v = v_func(x);
	const px = x0 + t * drawW;
	const py = y0 - v * scaleY; // 向下为正
	if (first) { ctx.moveTo(px, py);
		first = false; } else ctx.lineTo(px, py);
}
ctx.strokeStyle = "#1a6c9e";
ctx.lineWidth = 2.8;
ctx.stroke();
```

```
        // 填充
        ctx.beginPath();
        ctx.moveTo(x0, y0);
        for (let i = 0; i <= steps; i++) {
            const t = i / steps;
            const x = t * L;
            const v = v_func(x);
            const px = x0 + t * drawW;
            const py = y0 - v * scaleY;
            ctx.lineTo(px, py);
        }
        ctx.lineTo(x0 + drawW, y0);
        ctx.closePath();
        ctx.fillStyle = "rgba(26, 108, 158, 0.12)";
        ctx.fill();

        // 标注最大值
        const maxPos = L;
        const vMax = v_func(maxPos);
        const pX = x0 + drawW;
        const pY = y0 - vMax * scaleY;
        ctx.fillStyle = "#1a6c9e";
        ctx.font = "11px 'Segoe UI', sans-serif";
        ctx.textAlign = "right";
        ctx.fillText(`v_max = ${vMax.toFixed(4)}`, pX - 4, pY - 4);

        // 端点标注
        ctx.fillStyle = "#4a6885";
        ctx.font = "11px 'Segoe UI', sans-serif";
        ctx.textAlign = "center";
        ctx.fillText("固定端", x0, y0 + 22);
        ctx.fillText("自由端", x0 + drawW, y0 + 22);

        ctx.restore();
    }

    // ---------- 全部重绘 ----------
    function redrawAll() {
        readParams();
        drawMain();
        drawMoment();
        drawTheta();
        drawDeflect();
        updateFormulas();
```

```
}

// ---------- 绑定控件事件 ----------
function bindControls() {
    const inputs = [loadType, loadVal, lengthVal, eiVal];
    inputs.forEach(el => {
        el.addEventListener('input', redrawAll);
        el.addEventListener('change', redrawAll);
    });

    resetBtn.addEventListener('click', () => {
        loadVal.value = '10';
        lengthVal.value = '1.6';
        eiVal.value = '2.0';
        loadType.value = 'concentrated';
        redrawAll();
    });
}

// ---------- 窗口自适应 ----------
function setupCanvasSizes() {
    // 使用固定尺寸，但通过 CSS 保持比例，不需要额外操作
}

// ---------- 初始化 ----------
function init() {
    // 设置初始值
    loadVal.value = '10';
    lengthVal.value = '1.6';
    eiVal.value = '2.0';
    loadType.value = 'concentrated';
    readParams();
    bindControls();
    redrawAll();

    // 窗口变化时重绘（防抖）
    let resizeTimer;
    window.addEventListener('resize', () => {
        clearTimeout(resizeTimer);
        resizeTimer = setTimeout(() => {
            redrawAll();
        }, 100);
    });
}
```

```
        // 页面加载完成后初始化
        window.addEventListener('DOMContentLoaded', init);
    </script>

</body>
</html>
```