```
<!DOCTYPE html>
<html lang="zh-CN">
<head>
    <meta charset="UTF-8">
    <meta name="viewport" content="width=device-width, initial-scale=1.0, user-scalable=yes, maximum-scale=2.0">
    <title>外伸梁挠度与转角 | 叠加法演示 | 材料力学</title>
    <style>
        * {
            margin: 0;
            padding: 0;
            box-sizing: border-box;
        }
        body {
            font-family: 'Segoe UI', 'Roboto', 'Noto Sans', system-ui, sans-serif;
            background: #f5f7fb;
            display: flex;
            justify-content: center;
            align-items: center;
            min-height: 100vh;
            padding: 20px;
            margin: 0;
        }
        .container {
            max-width: 1000px;
            width: 100%;
            background: white;
```

```
    border-radius: 32px;
    box-shadow: 0 20px 40px rgba(0, 0, 0, 0.12), 0 8px 20px rgba(0, 20, 40, 0.1);
    padding: 24px 28px 28px;
    display: flex;
    flex-direction: column;
    gap: 20px;
}
h2 {
    font-weight: 500;
    font-size: 1.7rem;
    color: #1e293b;
    display: flex;
    align-items: center;
    gap: 10px;
    margin-bottom: 5px;
    letter-spacing: 0.5px;
    border-bottom: 2px solid #e9eef3;
    padding-bottom: 12px;
}
h2 span {
    background: #e0e7ff;
    color: #1e40af;
    font-size: 0.9rem;
    font-weight: 600;
    padding: 4px 14px;
    border-radius: 20px;
    letter-spacing: 0.3px;
```

```
}
.input-panel {
    background: #f8fafd;
    border-radius: 20px;
    padding: 18px 20px;
    display: flex;
    flex-wrap: wrap;
    align-items: flex-end;
    gap: 25px 35px;
    border: 1px solid #e2e8f0;
    box-shadow: 0 2px 8px rgba(0,0,0,0.02);
}
.force-group {
    display: flex;
    flex-wrap: wrap;
    align-items: center;
    gap: 20px 30px;
}
.slider-item {
    display: flex;
    flex-direction: column;
    gap: 6px;
    min-width: 140px;
}
.slider-item label {
    font-size: 0.8rem;
    font-weight: 600;
```

```
    color: #475569;
    text-transform: uppercase;
    letter-spacing: 0.4px;
    display: flex;
    align-items: center;
    gap: 6px;
}
.slider-item input[type="range"] {
    width: 130px;
    cursor: pointer;
    accent-color: #2563eb;
    background: #e2e8f0;
    height: 6px;
    border-radius: 3px;
}
.value-display {
    display: flex;
    align-items: center;
    gap: 8px;
}
.value-display input {
    width: 85px;
    padding: 8px 10px;
    border: 1px solid #cbd5e1;
    border-radius: 10px;
    font-size: 0.9rem;
    font-weight: 500;
```

```
    text-align: center;
    background: white;
    transition: 0.2s;
}
.value-display input:focus {
    outline: none;
    border-color: #2563eb;
    box-shadow: 0 0 0 3px rgba(37,99,235,0.2);
}
.unit {
    font-size: 0.8rem;
    color: #64748b;
    font-weight: 500;
}
.canvas-wrapper {
    background: #ffffff;
    border-radius: 24px;
    padding: 15px 10px 10px 10px;
    box-shadow: inset 0 1px 4px rgba(0,0,0,0.02);
    border: 1px solid #e9eef2;
    display: flex;
    justify-content: center;
}
canvas {
    display: block;
    width: 100%;
    height: auto;
```

```
    max-height: 380px;
    background: white;
    border-radius: 12px;
}
.results {
    background: #f1f5f9;
    border-radius: 18px;
    padding: 14px 20px;
    display: flex;
    flex-wrap: wrap;
    align-items: center;
    justify-content: space-between;
    gap: 18px;
    font-size: 0.95rem;
    border: 1px solid #dce3eb;
}
.deflection-box {
    display: flex;
    gap: 28px;
    flex-wrap: wrap;
    align-items: center;
}
.deflection-item {
    display: flex;
    align-items: baseline;
    gap: 8px;
    background: white;
```

```
    padding: 8px 18px;
    border-radius: 30px;
    box-shadow: 0 2px 6px rgba(0,0,0,0.03);
}
.deflection-item .label {
    font-weight: 600;
    color: #334155;
}
.deflection-item .value {
    font-weight: 700;
    font-size: 1.2rem;
    color: #0f172a;
}
.deflection-item .unit {
    color: #475569;
    margin-left: 2px;
}
.note {
    color: #475569;
    font-size: 0.8rem;
    background: #e6edf6;
    padding: 5px 15px;
    border-radius: 20px;
}
button {
    background: white;
    border: 1px solid #cbd5e1;
```

```
            padding: 8px 18px;
            border-radius: 30px;
            font-weight: 500;
            color: #1e293b;
            cursor: pointer;
            transition: 0.2s;
            font-size: 0.85rem;
        }
        button:hover {
            background: #f1f5f9;
            border-color: #94a3b8;
        }
        hr {
            border: 0.5px solid #e2e8f0;
            margin: 5px 0;
        }
        @media (max-width: 700px) {
            .container {
                padding: 16px;
            }
            .input-panel {
                flex-direction: column;
                align-items: flex-start;
            }
        }
    </style>
</head>
```

```
<body>
<div class="container">
  <h2>
    📐 外伸梁 · 叠加法演示
    <span>两个集中力</span>
  </h2>

  <!-- 上方输入数据区 -->
  <div class="input-panel">
    <div class="force-group">
      <div class="slider-item">
        <label>🔵 F₁ 位置 <span style="font-weight:400;">(距左端)</span></label>
        <div style="display:flex; align-items:center; gap:8px;">
          <input type="range" id="a1_slider" min="0.2" max="2.4" step="0.01" value="0.8">
          <div class="value-display">
            <input type="number" id="a1_input" value="0.8" step="0.01" min="0.2" max="2.4">
            <span class="unit">m</span>
          </div>
        </div>
      </div>
      <div class="slider-item">
        <label>🔴 F₂ 位置 <span style="font-weight:400;">(距左端)</span></label>
```

```
            <div style="display:flex; align-items:center; gap:8px;">
                <input type="range" id="a2_slider" min="2.6" max="4.6" step="0.01" value="3.8">
                <div class="value-display">
                    <input type="number" id="a2_input" value="3.8" step="0.01" min="2.6" max="4.6">
                    <span class="unit">m</span>
                </div>
            </div>
        </div>
    </div>
    <div class="force-group">
        <div class="slider-item">
            <label>🔵 F₁ 大小</label>
            <div style="display:flex; align-items:center; gap:8px;">
                <input type="range" id="F1_slider" min="0.5" max="12.0" step="0.1" value="5.0">
                <div class="value-display">
                    <input type="number" id="F1_input" value="5.0" step="0.1" min="0.5" max="12.0">
                    <span class="unit">kN</span>
                </div>
            </div>
        </div>
        <div class="slider-item">
            <label>🔴 F₂ 大小</label>
            <div style="display:flex; align-items:center; gap:8px;">
```

```
            <input type="range" id="F2_slider" min="0.5" max="12.0" step="0.1"
value="6.0">
            <div class="value-display">
              <input type="number" id="F2_input" value="6.0" step="0.1" min="0.5"
max="12.0">
              <span class="unit">kN</span>
            </div>
          </div>
        </div>
      </div>
      <button id="resetBtn" title="恢复默认参数">↺ 重置</button>
    </div>

    <!-- 中间外伸梁受力图 Canvas -->
    <div class="canvas-wrapper">
      <canvas id="beamCanvas" width="900" height="320" aria-label="外伸梁受力与变
形示意图"></canvas>
    </div>

    <!-- 下方挠度和转角数值 -->
    <div class="results">
      <div class="deflection-box">
        <div class="deflection-item">
          <span class="label">📏 自由端挠度 (C 点)</span>
          <span class="value" id="dispC_val">0.00</span>
          <span class="unit">mm</span>
        </div>
```

```
        <div class="deflection-item">
          <span class="label">📐 自由端转角 (C 点)</span>
          <span class="value" id="thetaC_val">0.000</span>
          <span class="unit">rad</span>
        </div>
        <div class="deflection-item">
          <span class="label">📍 跨中挠度 (AB 中点)</span>
          <span class="value" id="dispMid_val">0.00</span>
          <span class="unit">mm</span>
        </div>
      </div>
      <div class="note">
        ⚡ 弹性模量 E = 200 GPa · 惯性矩 I = 8.0×10⁻⁶ m⁴ (矩形截面近似)
      </div>
    </div>
    <div style="font-size:0.8rem; color:#64748b; margin-top:-8px; padding-left:10px;">
      * 梁总长 5.0m，AB 跨度 2.4m，外伸段 BC 长度 2.6m (A 铰支座位于 x=0, B 滚轴支座位于 x=2.4m，自由端 C 位于 x=5.0m)
    </div>
</div>

<script>
  (function() {
    // ----- 几何与材料参数 -----
    const L_AB = 2.4;          // A 到 B 距离 (m)
```

```
const L_BC = 2.6;              // B 到 C 距离 (m)
const totalLength = 5.0; //  总长 (m)
const E = 200e9;               //  弹性模量 Pa
const I = 8.0e-6;            //  惯性矩 m^4

//  支座位置 (用于绘制)
const xA = 0.0;
const xB = 2.4;
const xC = 5.0;

// ----- DOM 元素 -----
const a1Slider = document.getElementById('a1_slider');
const a1Input = document.getElementById('a1_input');
const a2Slider = document.getElementById('a2_slider');
const a2Input = document.getElementById('a2_input');
const F1Slider = document.getElementById('F1_slider');
const F1Input = document.getElementById('F1_input');
const F2Slider = document.getElementById('F2_slider');
const F2Input = document.getElementById('F2_input');
const resetBtn = document.getElementById('resetBtn');

const dispCVal = document.getElementById('dispC_val');
const thetaCVal = document.getElementById('thetaC_val');
const dispMidVal = document.getElementById('dispMid_val');

const canvas = document.getElementById('beamCanvas');
const ctx = canvas.getContext('2d');
```

```
// ----- 状态变量 -----
let a1 = 0.8;    // F1 作用位置距左端 A (m)
let a2 = 3.8;    // F2 作用位置距左端 A (m)
let F1 = 5.0;    // kN
let F2 = 6.0;    // kN

// ----- 工具函数：单位转换 -----
function kNToN(kN) {
  return kN * 1000;
}

// ----- 核心力学计算 (叠加法) -----
// 计算外伸梁在单个集中力 F (N) 作用在位置 xF (距 A) 时，关键点的挠度与转角。
// 返回对象包含：自由端 C 挠度(m)，自由端 C 转角(rad)，跨中 AB 中点挠度(m)
function computeSingleForceEffect(F_N, xF) {
  // 确保 xF 在 0~5 之间，但力学模型区分 AB 段(0~2.4)与 BC 段(2.4~5.0)
  const a = xF;
  const b = L_AB - a; // 仅当 a 在 AB 段有意义，但通用公式需分段。

  // 结果初始化
  let vC = 0.0;      // C 点挠度
  let thetaC = 0.0; // C 点转角
  let vMid = 0.0;   // AB 跨中 (x = L_AB/2 = 1.2m) 挠度

  // 采用基本叠加公式 (基于简支梁 AB 与外伸段 BC)
```

// 参考常规材料力学外伸梁公式：

// 情况 1：载荷在 AB 跨 (0 ≤ a ≤ L_AB)

if (a >= 0 && a <= L_AB) {

const a_m = a;

const b_m = L_AB - a_m;

// 支反力引起的效应：B 处转角等。

// 由集中力 F 在简支梁 AB 上产生 B 截面转角 θB = - (F * a_m * b_m * (L_AB + a_m)) / (6 * E * I * L_AB) (注意符号方向)

// 更标准公式：θB (顺时针为正假设) 实际采用向下挠度为正，需统一符号。

// 这里采用：向下挠度为正，转角从 x 轴转向梁变形方向(顺时针为正)便于理解。

// 使用公式：θB = - (F * a_m * (L_AB^2 - a_m^2)) / (6 * E * I * L_AB) ？ 需要谨慎。

// 标准外伸梁公式(集中力在 AB 跨):

// B 截面转角 θB = (F * a_m * (L_AB^2 - a_m^2)) / (6 * E * I * L_AB) (方向导致外伸段向下时为正)

// 采用向下挠度为正，θB 使得外伸段向下挠度为正。

const theta_B = (F_N * a_m * (L_AB * L_AB - a_m * a_m)) / (6.0 * E * I * L_AB);

// 外伸段 C 点挠度： vC = θB * L_BC (刚体转动贡献) + 若载荷在外伸段另有贡献，这里无。

vC = theta_B * L_BC;

// C 点转角 = θB (外伸段无载荷，转角相同)

thetaC = theta_B;

// AB 跨中点挠度 (x = L_AB/2) 使用简支梁公式叠加刚体位移？ 注意 A 支座位移为 0，B 支座位移为 0。

// 对于 AB 跨，中点挠度 = 简支梁在 F 作用下的挠度 + 由于支座 B 沉降？但 B 为滚轴支座，竖向位移 0。

// 直接使用简支梁挠度公式 (向下为正):

```
            const x_mid = L_AB / 2.0;
            if (x_mid <= a_m) {
                vMid = (F_N * b_m * x_mid * (L_AB * L_AB - b_m * b_m - x_mid * x_mid)) / (6.0 * E * I * L_AB);
            } else {
                vMid = (F_N * a_m * (L_AB - x_mid) * (L_AB * L_AB - a_m * a_m - (L_AB - x_mid) * (L_AB - x_mid))) / (6.0 * E * I * L_AB);
            }
            // 注意简支梁公式正确。
        }
        // 情况 2：载荷在 BC 跨 (外伸段) (a > L_AB)
        else if (a > L_AB && a <= totalLength) {
            const c = a - L_AB; // 载荷距 B 的距离
            // 外伸段集中力产生 B 截面转角 θB = - (F * c * L_AB) / (3 * E * I) ？ 查公式。
            // 标准：θB = (F_N * c * L_AB) / (3 * E * I)  (方向使外伸向下为正)
            const theta_B = (F_N * c * L_AB) / (3.0 * E * I);
            // C 点挠度：由转角引起 vC1 = θB * L_BC，再加上外伸段作为悬臂梁在 F 作用下的挠度。
            const vC_rotation = theta_B * L_BC;
            // 悬臂梁 BC 在距 B 为 c 处受集中力，C 点挠度： (F_N * c^2 * (3*L_BC - c)) / (6 E I)  (向下为正)
            const vC_cantilever = (F_N * c * c * (3.0 * L_BC - c)) / (6.0 * E * I);
            vC = vC_rotation + vC_cantilever;
            // C 点转角：θB + 悬臂梁在 C 端转角 (F * c^2)/(2 E I)
            thetaC = theta_B + (F_N * c * c) / (2.0 * E * I);

            // AB 跨中点挠度：仅由 B 截面转动引起，简支梁 AB 无直接载荷，故挠度线
```

性变化。

// A 点位移 0，B 点位移 0，但 B 处有转角，AB 梁自身挠曲？ 由于 AB 跨无载荷，梁保持直线，但 B 转动导致 AB 梁倾斜。

// 实际上 AB 跨无载荷时挠曲线为直线：v(x) = θB * x * (??) 需满足 v(A)=0, v(B)=0？不对，B 为铰支座位移 0，但转角不为 0。

// 若 AB 无载荷，挠曲线应为三次？不，无分布载荷且 A、B 位移为 0，则挠度恒为 0？错误。

// 正确：外伸段载荷使 B 产生转角，AB 段如同简支梁支座转动，产生线性挠度：v(x)= θB * x * (L_AB - x)/L_AB？需要推导。

// 采用叠加：支座 B 转动θB 引起的 AB 跨挠度 v(x) = θB * x * (L_AB - x) / L_AB (满足 A、B 位移 0，斜率匹配)

```
        const x_mid = L_AB / 2.0;
        vMid = theta_B * x_mid * (L_AB - x_mid) / L_AB;
      }

      return { vC, thetaC, vMid };
    }

    // 叠加两个力的效应
    function computeTotalEffect() {
      const F1_N = kNToN(F1);
      const F2_N = kNToN(F2);

      const effect1 = computeSingleForceEffect(F1_N, a1);
      const effect2 = computeSingleForceEffect(F2_N, a2);

      const totalVC = effect1.vC + effect2.vC;
      const totalThetaC = effect1.thetaC + effect2.thetaC;
```

```
    const totalVMid = effect1.vMid + effect2.vMid;

    return {
        vC_m: totalVC,
        thetaC_rad: totalThetaC,
        vMid_m: totalVMid
    };
}

// 更新数值显示 (转换为 mm, rad 保留)
function updateDisplay() {
    const res = computeTotalEffect();
    // 挠度转换为 mm (保留两位小数)
    dispCVal.textContent = (res.vC_m * 1000).toFixed(2);
    dispMidVal.textContent = (res.vMid_m * 1000).toFixed(2);
    thetaCVal.textContent = res.thetaC_rad.toFixed(6);
}

// ----- 绘图函数 -----
function drawBeam() {
    if (!ctx) return;
    ctx.clearRect(0, 0, canvas.width, canvas.height);

    const w = canvas.width;
    const h = canvas.height;

    // 坐标映射
```

```
const marginLeft = 90;
const marginRight = 70;
const beamY = h * 0.55;   // 原始梁轴线位置
const scaleX = (w - marginLeft - marginRight) / totalLength;

function toX(coord) {
   return marginLeft + coord * scaleX;
}

// 绘制浅灰参考线
ctx.save();
ctx.strokeStyle = '#d9e2ec';
ctx.lineWidth = 1;
ctx.beginPath();
ctx.moveTo(marginLeft - 10, beamY);
ctx.lineTo(w - marginRight + 15, beamY);
ctx.stroke();

// 绘制支座
// A 铰支座 (固定铰)
const Ax = toX(xA);
ctx.fillStyle = '#1e293b';
ctx.strokeStyle = '#0f172a';
ctx.lineWidth = 2;
// 三角形支座
ctx.beginPath();
ctx.moveTo(Ax - 12, beamY + 8);
```

```
ctx.lineTo(Ax + 12, beamY + 8);
ctx.lineTo(Ax, beamY + 22);
ctx.closePath();
ctx.fillStyle = '#475569';
ctx.fill();
ctx.stroke();
ctx.beginPath();
ctx.arc(Ax, beamY, 4, 0, Math.PI*2);
ctx.fillStyle = '#f8fafc';
ctx.fill();
ctx.strokeStyle = '#1e293b';
ctx.stroke();

// B 滚轴支座
const Bx = toX(xB);
ctx.beginPath();
ctx.moveTo(Bx - 12, beamY + 8);
ctx.lineTo(Bx + 12, beamY + 8);
ctx.lineTo(Bx, beamY + 22);
ctx.closePath();
ctx.fillStyle = '#475569';
ctx.fill();
ctx.stroke();
ctx.beginPath();
ctx.arc(Bx, beamY - 2, 6, 0, Math.PI);
ctx.fillStyle = '#f8fafc';
ctx.fill();
```

```
ctx.stroke();
ctx.beginPath();
ctx.arc(Bx, beamY - 2, 2.5, 0, Math.PI*2);
ctx.fillStyle = '#1e293b';
ctx.fill();

// 绘制原始梁（直线）
ctx.beginPath();
ctx.moveTo(toX(0), beamY);
ctx.lineTo(toX(totalLength), beamY);
ctx.strokeStyle = '#334155';
ctx.lineWidth = 3.5;
ctx.stroke();

// 绘制变形曲线（夸张系数）
const res = computeTotalEffect();
const scaleDeflection = 900; // 放大系数便于观察

ctx.beginPath();
let firstPoint = true;
for (let x = 0; x <= totalLength; x += 0.08) {
  // 计算该点挠度（使用叠加法逐点计算，可复用函数思想）
  function deflectionAt(xPos) {
    const F1N = kNToN(F1);
    const F2N = kNToN(F2);
    let v = 0.0;
    // 叠加两个力在 xPos 产生的挠度
```

```
[ {F: F1N, a: a1}, {F: F2N, a: a2} ].forEach(force => {
    const Fn = force.F;
    const xf = force.a;
    if (xf <= L_AB) { //  载荷在 AB 跨
        const a_m = xf;
        const b_m = L_AB - a_m;
        const theta_B = (Fn * a_m * (L_AB*L_AB - a_m*a_m)) / (6*E*I*L_AB);
        if (xPos <= L_AB) {
            //  简支梁挠度公式
            if (xPos <= a_m) {
                v += (Fn * b_m * xPos * (L_AB*L_AB - b_m*b_m - xPos*xPos)) /
(6*E*I*L_AB);
            } else {
                v += (Fn * a_m * (L_AB - xPos) * (L_AB*L_AB - a_m*a_m - (L_AB-
xPos)*(L_AB-xPos))) / (6*E*I*L_AB);
            }
        } else { //  外伸段
            v += theta_B * (xPos - L_AB);
        }
    } else { //  载荷在外伸段
        const c = xf - L_AB;
        const theta_B = (Fn * c * L_AB) / (3*E*I);
        if (xPos <= L_AB) {
            v += theta_B * xPos * (L_AB - xPos) / L_AB;
        } else {
            const s = xPos - L_AB;
            v += theta_B * s;
```

```
                if (s <= c) {
                    v += (Fn * s*s * (3*c - s)) / (6*E*I);
                } else {
                    v += (Fn * c*c * (3*s - c)) / (6*E*I);
                }
            }
        }
    });
    return v;
}

const vActual = deflectionAt(x);
const yDraw = beamY - vActual * scaleDeflection;
if (firstPoint) {
    ctx.moveTo(toX(x), yDraw);
    firstPoint = false;
} else {
    ctx.lineTo(toX(x), yDraw);
}
}
ctx.strokeStyle = '#dc2626';
ctx.lineWidth = 2.8;
ctx.stroke();

// 绘制力箭头
function drawForceArrow(xPos, forceKN, color) {
if (forceKN === 0) return;
```

```
    const x = toX(xPos);
    const dir = forceKN > 0 ? 1 : -1; // 向下为正
    const arrowLength = Math.min(45, 18 + Math.abs(forceKN) * 2.8);
    ctx.beginPath();
    ctx.moveTo(x, beamY - 4);
    ctx.lineTo(x, beamY + dir * arrowLength);
    ctx.strokeStyle = color;
    ctx.lineWidth = 2.5;
    ctx.stroke();
    // 箭头
    ctx.beginPath();
    const tipY = beamY + dir * arrowLength;
    ctx.moveTo(x, tipY);
    ctx.lineTo(x - 6, tipY - dir * 8);
    ctx.lineTo(x + 6, tipY - dir * 8);
    ctx.closePath();
    ctx.fillStyle = color;
    ctx.fill();
    ctx.fillStyle = color;
    ctx.font = 'bold 13px "Segoe UI"';
    ctx.fillText(`${Math.abs(forceKN).toFixed(1)} kN`, x + 8, tipY - dir * 6);
}

drawForceArrow(a1, F1, '#2563eb');
drawForceArrow(a2, F2, '#dc2626');

// 标注点
```

```
    ctx.fillStyle = '#0f172a';
    ctx.font = '500 13px "Segoe UI"';
    ctx.fillText('A', toX(0)-18, beamY-12);
    ctx.fillText('B', toX(2.4)+5, beamY-12);
    ctx.fillText('C', toX(5.0)+5, beamY-12);

    ctx.restore();
}

// ----- 同步滑块与输入框 -----
function syncFromSlider() {
    a1 = parseFloat(a1Slider.value);
    a1Input.value = a1.toFixed(2);
    a2 = parseFloat(a2Slider.value);
    a2Input.value = a2.toFixed(2);
    F1 = parseFloat(F1Slider.value);
    F1Input.value = F1.toFixed(1);
    F2 = parseFloat(F2Slider.value);
    F2Input.value = F2.toFixed(1);
    updateAll();
}

function syncFromInput() {
    let val1 = parseFloat(a1Input.value);
    if (!isNaN(val1)) {
        val1 = Math.min(2.4, Math.max(0.2, val1));
        a1 = val1;
```

```
        a1Slider.value = val1;
        a1Input.value = val1.toFixed(2);
    }
    let val2 = parseFloat(a2Input.value);
    if (!isNaN(val2)) {
        val2 = Math.min(4.6, Math.max(2.6, val2));
        a2 = val2;
        a2Slider.value = val2;
        a2Input.value = val2.toFixed(2);
    }
    let f1 = parseFloat(F1Input.value);
    if (!isNaN(f1)) {
        f1 = Math.min(12.0, Math.max(0.5, f1));
        F1 = f1;
        F1Slider.value = f1;
        F1Input.value = f1.toFixed(1);
    }
    let f2 = parseFloat(F2Input.value);
    if (!isNaN(f2)) {
        f2 = Math.min(12.0, Math.max(0.5, f2));
        F2 = f2;
        F2Slider.value = f2;
        F2Input.value = f2.toFixed(1);
    }
    updateAll();
}
```

```
function resetToDefault() {
    a1Slider.value = 0.8;
    a2Slider.value = 3.8;
    F1Slider.value = 5.0;
    F2Slider.value = 6.0;
    a1Input.value = "0.80";
    a2Input.value = "3.80";
    F1Input.value = "5.0";
    F2Input.value = "6.0";
    a1 = 0.8;
    a2 = 3.8;
    F1 = 5.0;
    F2 = 6.0;
    updateAll();
}

function updateAll() {
    updateDisplay();
    drawBeam();
}

// ----- 事件绑定 -----
a1Slider.addEventListener('input', syncFromSlider);
a2Slider.addEventListener('input', syncFromSlider);
F1Slider.addEventListener('input', syncFromSlider);
F2Slider.addEventListener('input', syncFromSlider);
```

```
        a1Input.addEventListener('change', syncFromInput);
        a2Input.addEventListener('change', syncFromInput);
        F1Input.addEventListener('change', syncFromInput);
        F2Input.addEventListener('change', syncFromInput);

        resetBtn.addEventListener('click', resetToDefault);

        // 初始绘制
        updateAll();

        // 窗口大小自适应保持清晰度
        window.addEventListener('resize', () => {
            drawBeam();
        });
    })();
</script>
</body>
</html>
```