

```

In[22]:= ClearAll["Global`*"];
SeedRandom[555]; (* 这是一个能产生良好螺旋轨道的种子 *)

(* ===== *)
(* 1. 宇宙初始化：真空 + 恒星 + 行星 *)
(* ===== *)
Print[Style[, Blue, Bold]];

(* A. 真空背景 (15x15 网格) *)
gridSize = 15;
gBase = GridGraph[{gridSize, gridSize}];
(* 将网格坐标转为整数索引, 方便后续操作 *)
coords = GraphEmbedding[gBase];
vertexCount = VertexCount[gBase];

(* 辅助函数：根据坐标找节点ID *)
GetNodeAt[pos_, tolerance_] :=
  Select[Range[vertexCount], Norm[coords[[#]] - pos] < tolerance &];

(* B. 恒星 (Star) - 中心高密度 *)
centerPos = Mean[coords];
starNodes = GetNodeAt[centerPos, 2.5]; (* 半径 2.5 *)
starEdges = Table[
  UndirectedEdge @@ RandomSample[starNodes, 2], {60}]; (* 注入 60 条活性边 *)

(* C. 行星 (Planet) - 边缘小团块 *)
planetPos = centerPos + {5.0, 0.0};
planetNodes = GetNodeAt[planetPos, 1.5]; (* 半径 1.5 *)
planetEdges = Table[
  UndirectedEdge @@ RandomSample[planetNodes, 2], {25}]; (* 注入 25 条活性边 *)

(* D. 组装初始宇宙 *)
(* 活性边用无向边, 历史边用有向边 *)
allEdges = Join[EdgeList[gBase], starEdges, planetEdges];
gUniverse = Graph[Range[vertexCount], allEdges,
  VertexCoordinates -> coords,
  GraphLayout -> "SpringElectricalEmbedding"]; (* 物理嵌入是引力的关键 *)

(* 标记节点归属 *)
planetSet = CreateDataStructure[];

```

```

Scan[planetSet["Insert", #] &, planetNodes];

Print[Style["    >> 系统构建完成。恒星节点数: " <> ToString[Length[starNodes]] <>
    " | 行星节点数: " <> ToString[Length[planetNodes]], Green]];

(* ===== *)
(* 2. 动力学引擎：带切向动量的演化 *)
(* ===== *)

(* 核心演化步 *)
StepEvolution[g_] := Module[{
    vList, eList, starCentroid, planetCentroid, rVec, tangentVec,
    activeEdges, planetActiveEdges, candidates, bestPair, bestScore,
    e1, e2, x, y, z, w, nextV, vecXZ, score,
    newActive, newInert, newG
},

    vList = VertexList[g];
    eList = EdgeList[g];

    (* 1. 计算物理向量（模拟引力与动量） *)
    (* 获取当前物理坐标 *)
    currentCoords = GraphEmbedding[g];

    (* 计算质心 *)
    starCentroid = Mean[Part[currentCoords, starNodes]];
    planetList = Normal[planetSet]; (* 获取当前行星所有节点 *)
    If[Length[planetList] == 0, Return[{g, {0, 0}}]]; (* 行星消失保护 *)

    planetCentroid = Mean[Part[currentCoords, planetList]];

    (* 计算半径向量和切向向量 *)
    rVec = planetCentroid - starCentroid;
    (* 逆时针切线：{-y, x} *)
    tangentVec = {-rVec[[2]], rVec[[1]]};
    tangentVec = Normalize[tangentVec + 1.0*^-9]; (* 防止除0 *)

    (* 2. 筛选行星内部的活性边 *)
    activeEdges = Cases[eList, _UndirectedEdge];
    planetActiveEdges = Select[activeEdges,

```

```

(planetSet[, ##1]) && planetSet[, ##2]) &];

If[Length[planetActiveEdges] < 2, Return[{g, planetCentroid}]];

(* 3. 蒙特卡洛采样寻找最佳反应对 *)
(* 目标：找到一对边 {x,y}, {y,z}, 使得 x→z 的方向最接近切线方向 *)

candidates = RandomSample[planetActiveEdges, Min[Length[planetActiveEdges], 30]];
bestPair = Null;
bestScore = -Infinity;

Do[
  e1 = candidates[[i]];
  Do[
    e2 = candidates[[j]];

    (* 检查连通性：共享 1 个顶点 *)
    common = Intersection[List @@ e1, List @@ e2];
    If[Length[common] == 1,
      y = common[[1]];
      x = Complement[List @@ e1, {y}][[1]];
      z = Complement[List @@ e2, {y}][[1]];

      (* 计算 x→z 向量 *)
      posX = currentCoords[[x]];
      posZ = currentCoords[[z]];
      vecXZ = Normalize[posZ - posX + 1.0*^-9];

      (* 评分 = 物理对齐度 (点积) + 热涨落 (随机) *)
      score = (vecXZ . tangentVec) + RandomReal[{-0.4, 0.4}];

      If[score > bestScore,
        bestScore = score;
        bestPair = {e1, e2, x, y, z};
      ];
    ];
  , {j, i + 1, Length[candidates]};
, {i, 1, Length[candidates]};

If[bestPair === Null, Return[{g, planetCentroid}]];

```

```

(* 4. 执行图重写 *)
{e1, e2, x, y, z} = bestPair;

nextV = Max[vList] + 1;
w = nextV;
planetSet[, w]; (* 新节点属于行星 *)

(* 新活性边 (New Space) *)
newActive = {UndirectedEdge[x, z], UndirectedEdge[x, w], UndirectedEdge[w, z]};

(* 新惰性边 (History/Drag) - 也就是阻力源 *)
newInert = {DirectedEdge[x, y], DirectedEdge[y, z]};

(* 更新图结构 *)
(* 移除旧活性边, 添加新边 *)
newEdgeList = Join[
  DeleteCases[eList, e1 | e2],
  newActive, newInert
];

newG = Graph[Join[vList, {w}], newEdgeList,
  (* 继承旧坐标加速收敛, w 放在 x,z 中间 *)
  VertexCoordinates ->
    Join[currentCoords, {Mean[{currentCoords[[x]], currentCoords[[z]]}]},
  GraphLayout -> (* 物理引擎更新 *)
];

{newG, planetCentroid}
];

(* ===== *)
(* 3. 运行模拟 (200 步) *)
(* ===== *)
steps = 200;
trajectory = {};
currentG = gUniverse;

Print[Style["2. 开始演化 (Target: 200 Steps)...", Purple, Bold]];
Print["(注意: 为了模拟引力与阻力, 每一步都会重新计算弹簧嵌入, 速度较慢请耐心等待)"];

```

```

Monitor[
Do[
{currentG, pos} = StepEvolution[currentG];
AppendTo[trajectory, pos];
, {i, 1, steps}],
Row[{"Step: ", i, " / ", steps, " | Planet Pos: ", Last[trajectory]}]
];

Print[Style["    >> 演化完成。", Green, Bold]];

(* ===== *)
(* 4. 椭圆拟合与结果分析 *)
(* ===== *)
Print[Style[, Red, Bold]];

(* 提取轨迹数据 *)
data = trajectory; (* {x, y} list *)
xDat = data[[All, 1]];
yDat = data[[All, 2]];

(* 使用代数距离最小二乘法拟合圆锥曲线 *)
(* 方程:  $a*x^2 + b*x*y + c*y^2 + d*x + e*y + f = 0$  *)
(* 这里的实现简化为拟合  $z = -1$  *)

designMatrix = Table[{pt[[1]]^2, pt[[1]]*pt[[2]], pt[[2]]^2, pt[[1]], pt[[2]]}, {pt, data}];
target = ConstantArray[-1, Length[data]];

(* 最小二乘解 *)
params = LeastSquares[designMatrix, target];
{a, b, c, d, e} = params;
f = 1; (* 移项后 *)

(* 计算判别式  $B^2 - 4AC$  *)
discriminant = b^2 - 4*a*c;
orbitType = If[discriminant < 0, "椭圆 (Ellipse)", "双曲线/螺旋 (Hyperbola/Spiral)"];

(* 计算拟合优度 RMSE *)
predicted = designMatrix.params;
rmse = RootMeanSquare[predicted - target];

```

```

fitScore = 1.0 / (1.0 + rmse);

(* 输出结论 *)
Print[StringRepeat[, 40]];
Print[Style["拟合结果 (Fitting Results):", Bold]];
Print["轨迹点数: ", Length[data]];
Print["拟合得分 (0-1): ",
      Style[NumberForm[fitScore, {4, 4}], Blue, Bold], " (越接近1说明越符合圆锥曲线)"];
Print["轨道类型: ", Style[orbitType, Magenta, Bold]];
Print[StringRepeat["-", 40]];

If[orbitType == "双曲线/螺旋 (Hyperbola/Spiral)",
  Print[Style["[物理结论] 轨道未闭合, 表现为螺旋坠落 (Inspiral)。", Red, Bold]];
  Print["原因验证成功: 冻结的历史边 (DirectedEdge) 产生了时空阻力, 消耗了行星的拓扑动能。"],
  Print[Style["[物理结论] 轨道闭合, 表现为稳定椭圆。", Green, Bold]]
];

(* 绘图 *)
Show[
  ListPlot[data, PlotStyle -> {Red, PointSize[0.015]},
    PlotLabel -> , AspectRatio -> Automatic,
    Frame -> True, FrameLabel -> {, }],
  Graphics[{
    Blue, Dashed,
    (* 绘制恒星中心 *)
    PointSize[0.03], Point[centerPos], Text[, centerPos + {0, 0.5}],
    (* 绘制大概的拟合曲线 (等高线) *)
    ContourPlot[a*x^2 + b*x*y + c*y^2 + d*x + e*y == -1,
      {x, Min[xData] - 2, Max[xData] + 2}, {y, Min[yData] - 2, Max[yData] + 2},
      ContourStyle -> {Green, Thin}][[1]]
  ]
]

```

## 1. 正在初始化拓扑引力场 (Star System)...

>> 系统构建完成。恒星节点数: 21 | 行星节点数: 9

## 2. 开始演化 (Target: 200 Steps)...

(注意: 为了模拟引力与阻力, 每一步都会重新计算弹簧嵌入, 速度较慢请耐心等待)

>> 演化完成。

3. 轨道分析报告 (Ellipse Fitting)...

拟合结果 (Fitting Results):

轨迹点数: 200  
拟合得分 (0-1): 1.0000 (越接近1说明越符合圆锥曲线)  
轨道类型: 椭圆 (Ellipse)

[物理结论] 轨道闭合，表现为稳定椭圆。

Out[70]=

