基于光线追踪的苹果树树冠结构与受光和阴影情况分析

摘要

本文从对苹果树树冠层受光和阴影面积考量出发，通过构建三维坐标系内光线参数方程与树冠层纺锤状结构，分别建立苹果树树冠模型、行向间苹果树树冠模型、行向间苹果树树冠模型和带坡度的行列间苹果树树冠模型。分别对单个树冠层、单行林带、正方形林区、带斜度正方形林区的受光及遮阴形成阴影随时间、季节变化情况做出分析。最后综合分析结果，对当地林木栽植与管护提出建议。

针对问题 1：首先定义时角和赤纬，结合延安地区经纬度信息，得到当地太阳高度角和太阳方位角计算公式。随后构建苹果树树冠模型，将树冠结构简化为纺锤形分析，通过三层不同疏密参数纺锤结构模拟树冠结构内疏，中密，外次疏的特征。以树冠中心为原点构建三维坐标系，得出树冠三层的表达方程，同时定义面光源，在春分、夏至、秋分、冬至四日以 1 小时为步长在$8\sim17$时时间段内取时间点。针对每个时间点利用太阳高度角与太阳方位角构建光线参数方程，并与树冠三个方程联立。将每个时间点上所有光线与树冠层外层的首个交点标记后拟合成为受光面，计算其面积值再按日期拟合成曲线并分析，得到受光面随时间自东向西，“侧—顶—侧”方向偏移，面积呈 U 形变化；随季节由向顶到向侧，“南—北—南”方向偏移，面积呈“减—增—减”变化。同时，以0.01为步长在光线上取点并判断在椭球哪层内，各层点数量值加权求得加权阴影深度S，设立标准将深度类别分为淡影、暗影和浓影三类。统计每一时间点（冬至8时与17时不作考虑）所有$S>0$的光线与地平面交点并标记，拟合成阴影面，根据标记点数目计算其面积，得到阴影随时间自西向东旋转，面积先减后增的U形变化，随季节由向南到向北，轮廓夏短冬长，深度夏深冬浅，面积呈“减—增—减”变化。

针对问题 2：在苹果树树冠模型基础上扩展到1×3单行林带，针对行列方向和树间距两个变量设计共计 6 组林带，取中间树冠层进行受光面模拟，对林带整体阴影进行阴影模拟。对拟合曲线分析可得：由于相邻树木遮挡，受光面面积总体随时间变更为鞍形变化。东—西走向的受光面面积会季节性在早晚两个时间段随树间距缩短而减小，南—北走向的受光面面积会季节性在正午左右呈同样变化。林带整体阴影仍遵循单个树冠层的变化规律，东西走向条件下树间距在一定限度内对阴影面积呈正向影响，南北走向条件下则不产生影响。

针对问题 3：在行向间苹果树树冠基础上扩展为东南西北方向上3×3正方形林区，进行受光以及阴影模拟，对拟合曲线分析所得结果与问题 2 中类似：受光面面积呈鞍形变化，树间距的增大仅在早晚对面积具有一定限度内的增进作用。林区整体阴影面积变化规律不变，也仅在早晚受树间距正向影响，不同之处在于春、夏、秋三季正午存在相同谷值，冬季该值随树间距增大而增大

针对问题 4：在行列间苹果树树冠模型中添置西北坡向、坡度 10° 的参考因子，将其转化在正东南西北方向上即可进行模拟。分析拟合曲线，由于存在高度差，相比无坡度林区，有坡度林区在春、夏、秋季早晚的受光面积分别较小和较大，冬季由于高度角较小可能照射到被遮挡树冠下半，受光面积全天较大。阴影面积则在春、秋、冬季早间由于坡度方向会有显著提升。

针对问题 5：综合以上模型分析及资料查找，取坡度和树间距作为林区光照情况参考，结合水分、土壤肥力等因素，提出纺锤形、$4\sim5$米树间距、$5\sim10°$ 坡度的植株栽培标准，并进一步提出建议。

关键词：苹果树；树冠结构；光照分布；树木栽种方式

一、 问题重述

1.1 问题背景

习近平总书记就生态文明建设提出了一系列新思想新理念，深刻诠释了“山水林田湖草”是一个生命共同体的发展思路。林草作为地球的“健康色”，林草高质量发展对生态环境保护起着重要意义，在维护国家生态安全，推进生态文明建设中具有非常广泛作用，于社会发展、经济发展、生态发展都有重要的促进作用。因此，扎实推进林草高质量发展与保护，是践行习近平生态文明思想的具体行动，是实现“山水林田湖草”系统治理的必然选择。

黄土高原日照时间长，昼夜温差大，工业污染较少，是联合国粮农组织认定的苹果最佳优生区，也是红枣、核桃、花椒、葡萄等经济林果品的优生区域，具有发展经济林果产业优势。黄土高原地区社会经济的可持续发展是生态环境改善的根本保障，黄土高原一些县区的实践证明，经济林果产业的发展能够实现生态环境改善和社会经济发展的双赢。经济林是兼具生态、经济和社会效益的林种，国家新一轮退耕还林取消了营造生态林与经济林的比例限制，并明确提出大力发展特色经济林等林业产业。因此，需要对光照这一经济作物质量的重要影响因素进行分析，突出当地资源优势，因地制宜、合理布局、规模发展，加大对现有经济林的精细化科学管理，提高经济林果品的产量和质量，提升黄土高原经济林果品基地建设，以经济效益保障生态效益，巩固生态环境治理成效。[1][2]

1.2 问题重述

问题 1：试分析单棵植株树冠的受光及遮阴形成阴影与时间，季节变化的相关性；
问题 2：试分析不同林带行向间植株树冠的受光及遮阴而形成阴影与时间、季节变化的相关性；
问题 3：试分析林区中不同行列间树木树冠的受光及遮阴形成阴影与时间，季节变化的相关性；
问题 4：以西北坡向，坡度为$10°$的陕北黄土高原地形重新分析问题三；
问题 5：根据以上分析，对于林木栽植与林木结构管护提出合理建议。

二、 问题分析

考虑到真实树冠内部枝叶交错，结构复杂，将树冠层结构简化为三层纺锤结构，以树冠中心为坐标轴原点建立三维坐标系，并列出每层相应的椭球方程。

问题 1 的分析：

该题目可分为受光与遮阴两个部分：

1. 针对单棵苹果树树冠的受光，根据设定时间的太阳高度角与太阳方位角，以太阳高度平面作为面光源，在以树冠中心为坐标轴原点的三维坐标系中建立光线参数方程，与树冠层外层的椭球方程联立后求解出首个交点并进行标记，所有标记点拟合后形成曲面即为设定时间下单株苹果树树冠的受光面；
2. 针对单株苹果树树冠遮阴形成阴影，以一定步长在光线上取点，判断点是否在树冠层内，在其中哪一层，通过对树冠层三层分别设立权重，得到每条光线所包含点的加权总数，设立标准判断阴影深度。

问题 2 的分析：

在问题一模型的基础上将受光目标拓展为1×3单行林带，根据重要特征，按

行列方向与树间距设计共计 6 组林带进行相关情况模拟。针对受光仅需考虑中间树冠层的情况，向整行推广即可；由于有邻木遮挡，针对阴影则须考虑整体阴影情况。

问题 3 的分析：

在问题 2 模型的基础上将1×3单行林带拓展为东南西北方向的3×3正方形林区，依旧在春分、夏至、秋分、冬至四日内均匀取点进行相关情况模拟。针对受光仅需考虑中间树冠层的情况，针对阴影则须考虑整体阴影情况。对面积计算结果拟合呈曲线后进行图表分析，即得到变化情况。

问题 4 的分析：

在问题 3 模型的基础上将3×3正方形林区添加西北坡向，坡度为 10° 的影响参数，并转化为东—西方向和南—北方向上的坡度进行分析，步骤同上。

问题 5 的分析：

以林区受光情况作为第一参考指标，根据已建立模型的分析结果添入指标因子，再以水分、土壤肥力等因素作为辅助参考指标修改，综合得出对林木栽植与结构管护的合理建议。

三、 模型假设

1. 本题考虑的所有苹果树树冠均可被简化为纺锤形结构；
2. 假设苹果树的树冠结构不随时间、季节变化；
3. 太阳光可被近似认为是平行光。

四、 符号说明

符号	意义	单位
Λ	本地经度	°
Φ	本地纬度	°
δ	太阳赤纬	°
h	时角	°
θ_s	太阳高度角	°
ϕ_s	太阳方位角	°
f_{ij}	苹果树树冠结构方程	/
S	加权阴影深度值	/
θ	坡度	°
d	树冠间距	m

表 1 符号说明

五、 模型建立与求解

5.1 问题一模型建立与求解

5.1.1 太阳高度角及方位角计算

对于一个特定区域范围内的经济林带而言，不同时间下其受光照情况各不相同，其主要原因是当地太阳高度角及太阳方位角在随着时间季节变化而变化。在此引入这两个概念，并通过题目给出的延安经纬度，对太阳高度角及太阳方位角进行了计算。

首先对另外两个会在计算过程中出现的值——时角与赤纬进行解释

时角是天文学专有名词，对于天球上一个点而言，其时角是两个平面之间的角度：一个平面包含地轴和天顶（子午面），另一个平面是穿过该点与极点（地轴）的球面大圆切圆所形成的平面（赤经圈），如图 1 所示。时角由子午线确定，遵循的规则是在子午线的东边则为负时角，在子午线的西边则为正时角。时角与赤纬结合使用，即可精确确定天球上一点在赤道坐标系下的坐标。其计算过程如下：

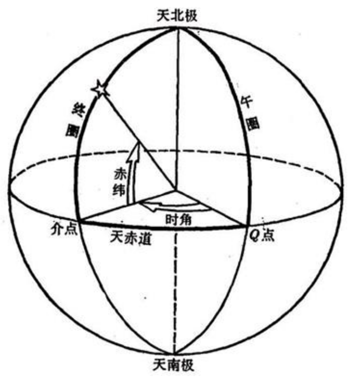


图 1 时角与赤纬

$$h = 15 \times (t + (\Lambda - 120°)/15° - 12) \tag{1}$$

其中t为北京时间，Λ是本地经度。

赤纬与地球上的纬度相似，是纬度在天球上的投影。赤纬的单位是度，更小的单位是“角分”和“角秒”。太阳赤纬等于太阳入射光与地球赤道之间的角度，由于地球自转轴与公转平面之间的角度基本不变，因此太阳的赤纬随季节周期性变化，变化的周期等于地球的公转周期，即一年。由于地球公转轨道的偏心率非常低，可以看作是一个圆圈，太阳赤纬可用下面两个公式来计算：

$$\delta = -23.44° \cdot \cos\left(\frac{360}{365} \cdot (N + 10)\right) \tag{2}$$

余弦中的角度的单位是角度，N 是一年中的日数，例如1月1日的日数计为1

通过时角与赤纬，可计算任意时刻的太阳高度角和太阳方位角。

太阳高度角，也称太阳高度，是指某地的太阳光线与当地地平面的所交的最小线面角，即为以太阳视平面的几何中心和理想地平线所夹的角度。其计算可以使用以下算式，得到良好的近似值：

$$\sin\theta_s = \cos h \cos\delta \cos\Phi + \sin\delta \sin\Phi \tag{3}$$

此处 θ_s是太阳高度角。h是地方恒星时系统下的时角。δ是当前太阳的赤纬，Φ是当地的纬度。

太阳方位角则是太阳在方位上的角度，它通常被定义为从正北方向沿着地平残顺时针量度的角。其可以利用下面的公式，经由计算得到良好的近似值。注意到此处公式使用的是余弦函数，所以方位角永远是正值，因此，角度永远被解释为小于180度，而必须依据时角来修正。

$$\cos\phi_s = \frac{\sin\delta - \sin\theta_s \sin\Phi}{\cos\theta_s \cos\Phi} \tag{4}$$

其中 ϕ_s是太阳方位角，θ_s是太阳高度角，h是时角，δ是太阳赤纬，Φ是当地纬度。

延安当地的经度为$\Lambda = 109.5°$，纬度为$\phi = 36.5°$，故由式(3)(4)可得一

年中的春分、夏至、秋分、冬至四天的每个时段的太阳高度角及方位角，如表2、3 所示。

时间	8 时	11 时	14 时	17 时
春分（3 月 21 日）	15.253°	46.078°	48.806°	19.927°
夏至（6 月 22 日）	28.868°	64.460°	68.721°	33.650°
秋分（9 月 23 日）	15.003°	45.728°	48.437°	19.670°
冬至（12 月 22 日）	0.551°	25.411°	27.300°	4.643°

表 2 太阳高度角随季节、时间的变化

时间	8 时	11 时	14 时	17 时
春分（3 月 21 日）	102.306°	141.640°	210.452°	253.746°
夏至（6 月 22 日）	80.961°	113.631°	237.557°	275.847°
秋分（9 月 23 日）	102.634°	141.929°	210.204°	253.415°
冬至（12 月 22 日）	120.127°	154.068°	200.161°	236.186°

表 3 太阳方位角随季节、时间的变化

5.1.2 苹果树树冠模型

考虑到苹果树树冠具体形态较难模拟，模型建立会非常复杂，经过资料查找与不同苹果树植株形态比对，现将苹果树树冠简化为纺锤形，以方便模型建立与分析。在文献查阅和实际观察中易得，树冠不同部分的枝叶疏密程度不同，整体呈现为内疏，中密，外次疏的特征。考虑到纺锤模型本身为均匀实心结构，需要对模型做出一定的优化调整。首先定义疏密参数，越接近1的其枝叶越稠密，越接近0的其枝叶越稀疏。此处使用叠套的方式来模拟苹果树的疏密特征，由内到外依次将疏密参数为0.2、0.8、0.5共三层纺锤进行嵌套。具体结构和三维示意图如图 3、4 所示：

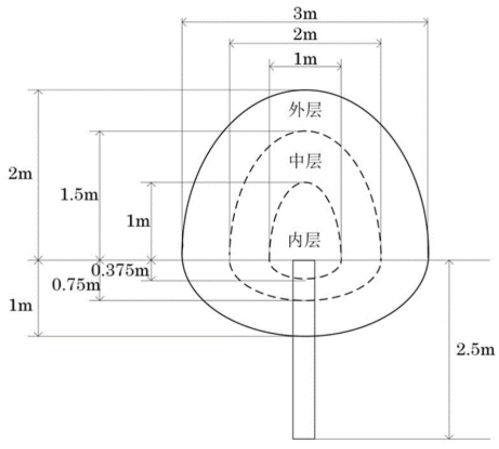


图 3 苹果树树冠结构示意图

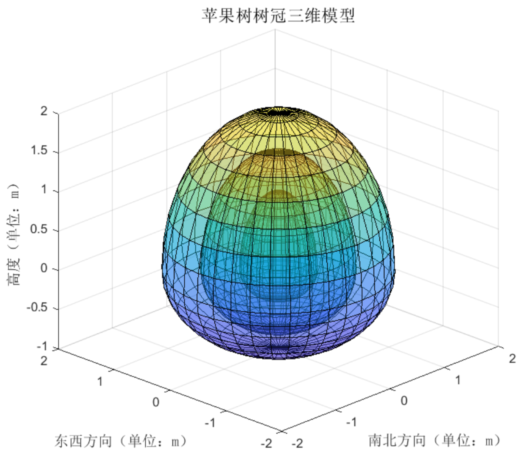


图 4 苹果树树冠三维模型

以树冠中心为原点，正南方向为x轴正半轴，正东方向为y轴正半轴，树木生长方向为z轴正半轴建立三维坐标系。使用椭球方程对树冠进行表示。

$$(\frac{x}{a})^2+(\frac{y}{b})^2+(\frac{z}{c})^2=1$$

其中a、b、c分别等于该层x轴、y轴、z轴半径，由结构图易得：

内层：

$$f_{11} = 4x^2 + 4y^2 + z^2 - 1 \qquad (z \geq 0)$$
$$f_{12} = 4x^2 + 4y^2 + \frac{64}{9}z^2 - 1 \quad (z \leq 0)$$

中层：

$$f_{21} = x^2 + y^2 + \frac{4}{9}z^2 - 1 \qquad (z \geq 0)$$
$$f_{22} = x^2 + y^2 + \frac{16}{9}z^2 - 1 \quad (z \leq 0)$$

外层：

$$f_{31} = \frac{4}{9}x^2 + \frac{4}{9}y^2 + \frac{z^2}{4} - 1 \qquad (z \geq 0)$$
$$f_{32} = \frac{4}{9}x^2 + \frac{4}{9}y^2 + z^2 - 1 \quad (z \leq 0)$$

5.1.3 苹果树树冠受光情况分析

在此情况下只需考虑树冠外层表面受光情况。具体分析分布如下：

Step1: 在其时太阳高度平面（$0, 0, z_0$）上任意取点（x_0, y_0, z_0），在上述建立的三维坐标系中建立某一时间光线的空间直线参数方程：

$$\frac{x - x_0}{m} = \frac{y - y_0}{n} = \frac{z - z_0}{p} = t$$

$$即\begin{cases} x = x_0 + mt \\ y = y_0 + nt \\ z = z_0 + pt \end{cases}$$

其中$\frac{n}{m} = \tan(90^\circ - \phi_s)$，$p = tan\,\theta_s$

Step2：将经过该点的光线方程和树冠外层方程联立，判断其光线方程是否与树冠外层方程存在交点。如若存在两个交点，则只取第一个交点在树冠外层标记；如若只有一个交点，则直接标记该交点即可；如若不存在交点，则无需标记。

Step3：将所有标记点拟合成面并显示，即为该时间段树冠受光面。

Step4：对受光面积进行计算，设空间曲面 S 的方程为 $z = z(x, y)$，$(x, y) \in \Omega$，其中 Ω 为曲面 S 在 O_{xy} 平面上的投影域，函数 $f(x, y, z)$ 在曲面S上连续，如果 $z(x, y)$ 在 Ω 上有连续的一阶偏导数，则有

$$\begin{aligned}\Delta s_i \quad &= \iint_{\Delta\sigma_i} \sqrt{1 + \left(\frac{\partial z}{\partial x}\right)^2 + \left(\frac{\partial z}{\partial y}\right)^2}\, d\sigma \\ &= \sqrt{1 + \left(\frac{\partial z}{\partial x}\right)_i^2 + \left(\frac{\partial z}{\partial y}\right)_i^2}\, \Delta\sigma_i.\end{aligned}$$

其中 $\Delta\sigma_i$ 是 ΔS_i 在 O_{xy} 上的投影域，$\left(\frac{\partial z}{\partial x}\right)_i$ 和 $\left(\frac{\partial z}{\partial y}\right)_i$ 表示在 $\Delta\sigma_i$ 内某点 (x_i, y_i) 处的两个偏导数。由第一型曲面积分的定义，于是将第一型曲面积分化为二重积分的计算

$$\iint_S f(x, y, z) dS = \iint_{\Omega_{xy}} f(x, y, z(x, y)) \sqrt{1 + \left(\frac{\partial z}{\partial x}\right)^2 + \left(\frac{\partial z}{\partial y}\right)^2}\, d\sigma$$

在此以春分、夏至、秋分、冬至四个具有季节代表性的日期为例，分取当天 8 时、

11 时、14 时，17 时四个时间段进行光照情况模拟，具体情况如下述俯视图所示（均遵循“上北下南，左西右东”的方向原则）。

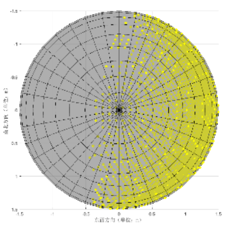
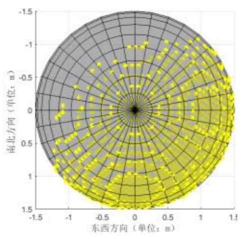
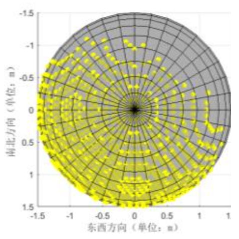
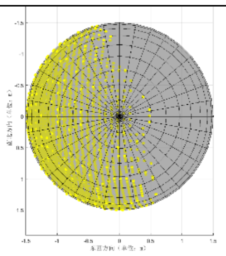
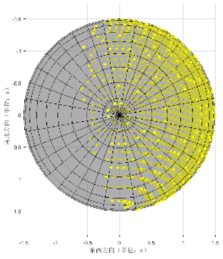
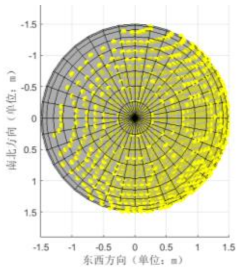
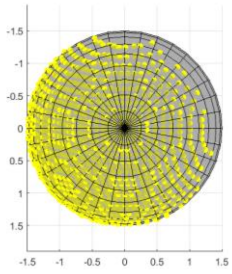
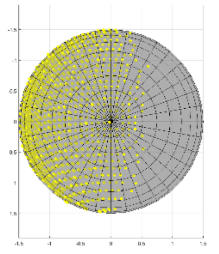
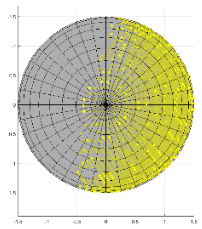
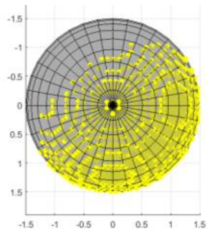
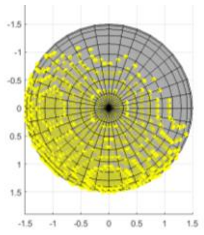
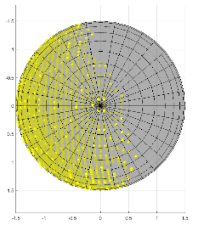
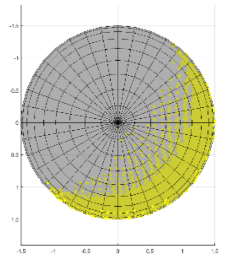
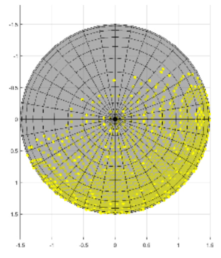
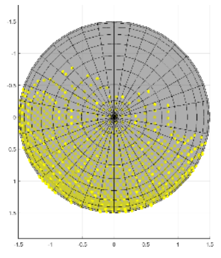
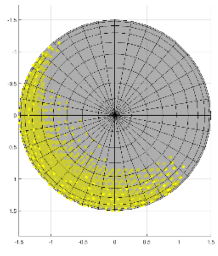
时间	8 时	11 时	14 时	17 时
春分				
受光面积	$18.98m^2$	$7.79m^2$	$7.58m^2$	$14.72m^2$
夏至				
受光面积	$10.45m^2$	$6.21m^2$	$6.06m^2$	$9.18m^2$
秋分				
受光面积	$19.27m^2$	$7.84m^2$	$7.65m^2$	$14.87m^2$
冬至				
受光面积	$20.67m^2$	$13.17m^2$	$12.50m^2$	$19.44m^2$

表 4 单株苹果树树冠受光随季节、时间的变化情况

通过对表格横向与纵向的分析可得：

1、随时间变化：树冠层受光面由早到晚呈自东向西的旋转趋势，早晚受光更集中在树冠层侧面上（图中显示为圆的边缘），靠近正午时受光面开始向顶层偏移（图中显示为圆的中心）。此时受光也更密集充分。

2、随季节变化：夏季受光在树冠层顶层更为集中，受光面整体偏向北方一些，

冬季受光在树冠层侧面更为集中，受光面整体偏向南方一些。总的趋势呈现为受光由侧到顶，再回到侧循环变化，受光面偏移自北向南，再回到南循环变化。

将模拟范围扩展至 8 时至 17 时，步长取为一个小时，对每一时间点的树冠层受光照面积进行统计，数据拟合曲线如下图所示。

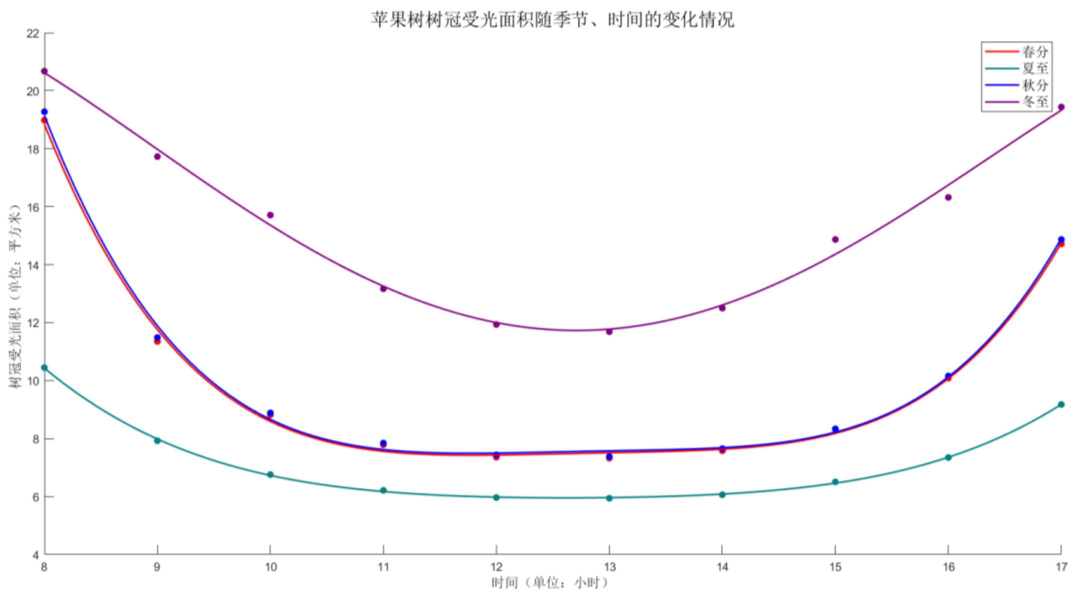


图 5 苹果树树冠受光面积随季节、时间的变化情况

分析上图中曲线变化趋势可得：

1、受光面积随时间变化情况：8 时光照面积即处于峰值，随后受光面积随时间缩减，约在正午时分达到谷值，而后又随时间扩增，整体变化曲线为U形，即呈现先减后增的变化趋势。

2、受光面积随季节变化情况：在一天之中，夏季树冠受光面积在谷值稳定持续时间较长，整体变化趋于平缓，冬季树冠受光面积处于时刻变化状态，峰值与谷值存在一定差距，而春秋两季属于树冠层受光面积早晚变化速度较快、变化幅度较大，中间时间段却较为平稳地维持在谷值。此外，夏季整体受光面积最小，冬日整体受光面积最大，春秋季整体受光面积相近，随四季更迭整体呈现“减—增—减”的循环变化趋势。

5.1.4 苹果树树冠阴影情况分析

通过观察比对可以发现，阳光下树木产生阴影在不同部分阴影深度不一，在此特引入加权阴影深度 S 对树冠遮挡产生阴影进行评估。

Step1：建立光线方程，具体步骤同上。

Step2：记内层内点个数集为N_1，中层内点个数集为N_2，外层内点个数集为N_3，初始值均设为0

Step3：取步长为0.01，分别判断每一个点是否在树冠层的纺锤形状内，如不在则跳过该点，如在，则判断该点在树冠层内层、中层还是外层，并在相应的个数集内+1。

Step4：对N_1、N_2、N_3分别设立权重，通过加权相加的办法，计算得出该条线的

加权阴影深度值S：

$$S = N_1 \times 0.2 + N_2 \times 0.8 + N_3 \times 0.5$$

并根据S值的大小对加权阴影深度进行分类：

- 如$0 < S \leq 6$,则产生的阴影影深为淡影
- 如$6 < S \leq 12$,则产生的阴影影深为暗影
- 如$S > 12$，则产生的阴影影深为浓影

Step5：重复上述过程，对所有S值大于 0 的光线与地平面的交点进行标记，将所有标记点拟合成面，所得即当前时段树冠层遮挡阳光产生的阴影。

Step6：根据组成阴影的标记点个数计算该阴影面积。

在此以春分、夏至、秋分、冬至四日为例，分取当天 8 时、11 时、14 时，17 时四个时间段进行阴影情况模拟，其中冬至日由于 8 时和 17 时太阳高度角过低，造成阴影过长，数据量较大，在此更改为 9 时与 16 时。具体情况如下述俯视图所示（均遵循“上北下南，左西右东”的方向原则）

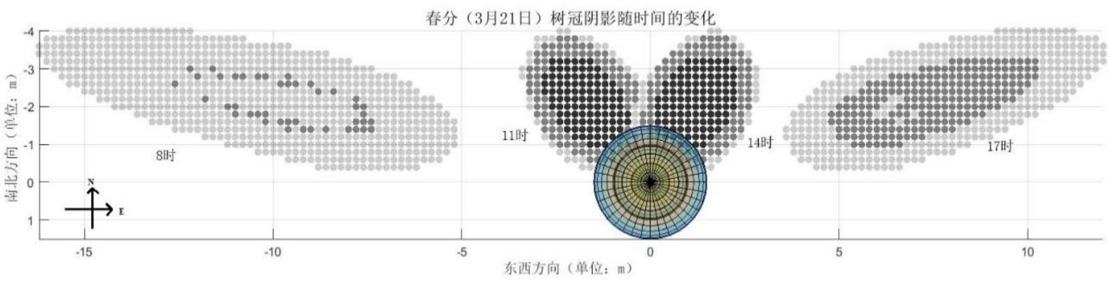


图 6 苹果树树冠阴影在春分（3 月 21 日）不同时间的情况

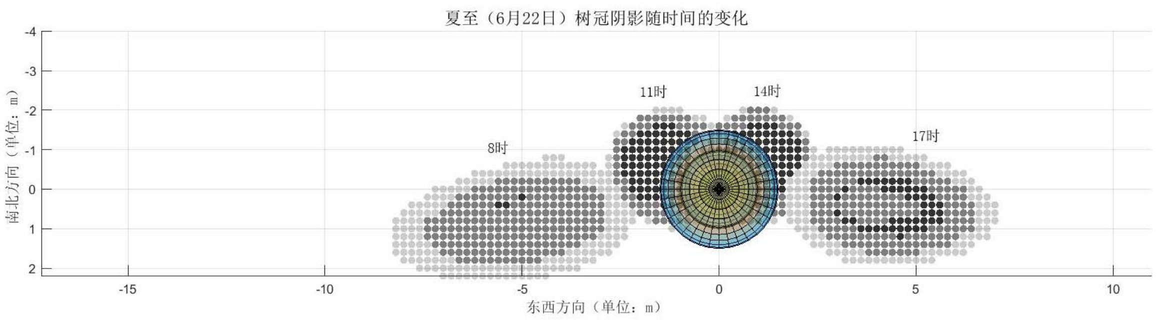


图 7 苹果树树冠阴影在夏至（6 月 22 日）不同时间的情况

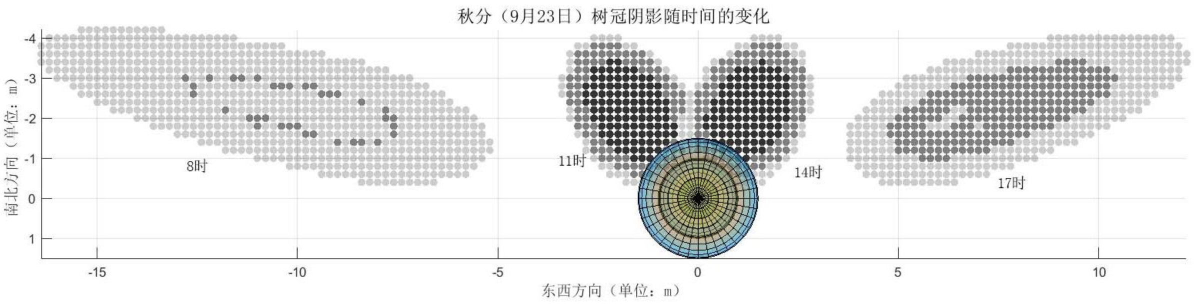


图 8 苹果树树冠阴影在秋分（9 月 23 日）不同时间的情况

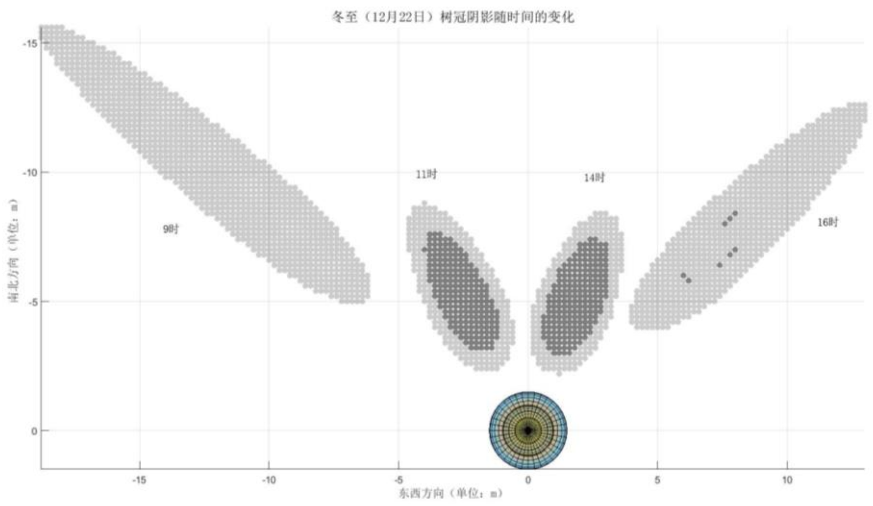


图 9 苹果树树冠阴影在冬至（12 月 22 日）不同时间的情况

综合上面四图分析可得：阴影总体移动方向为自西向东顺时针旋转，在夏季整体方向偏南方一些，在冬季整体方向偏北方一些。无论对于哪一个时间点而言，树冠层阴影主要深度都集中在阴影中间，越靠近边缘影深越浅。时间越靠近早晚，日期越接近冬季，阴影越长，整体面积越大，影深越浅（以淡影和暗影为主）；时间越靠近正午，日期越接近夏季，阴影越短，整体面积越小，影深越深（以暗影与浓影为主）

将模拟范围扩展至 8 时至 17 时，步长取为一个小时，对每一时间点的阴影面积进行统计，数据拟合曲线如下图所示。

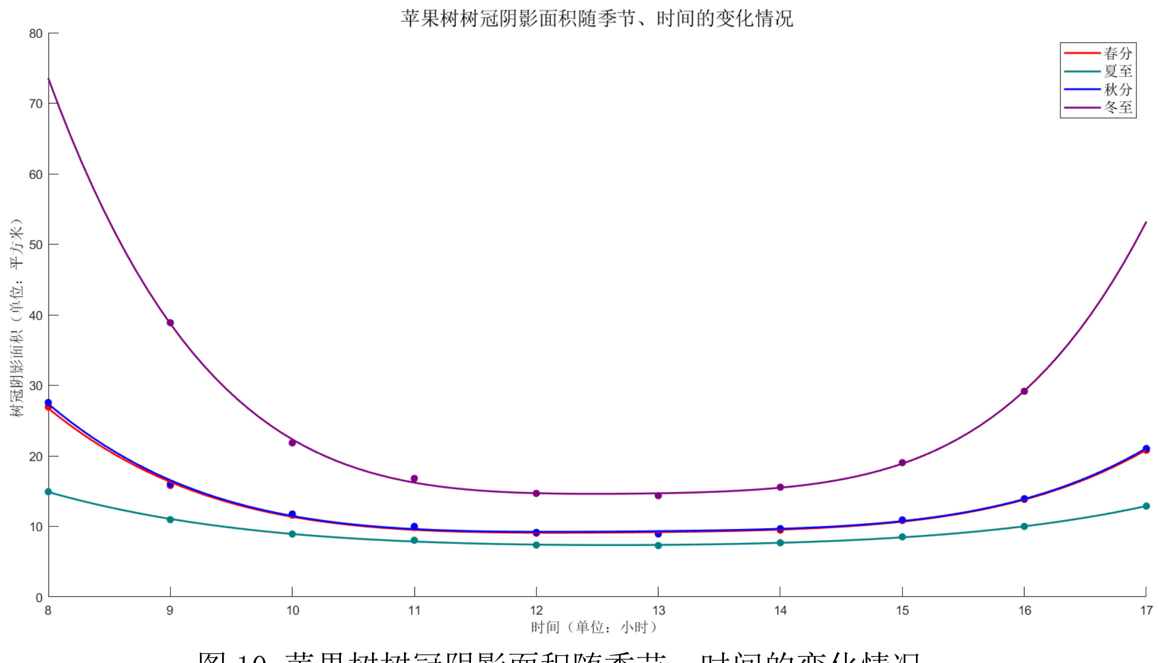


图 10 苹果树树冠阴影面积随季节、时间的变化情况

通过对上图的观察分析不难得出：

1. 阴影面积随时间变化情况：8 时（初始时间）阴影面积即处于峰值，从 8 时至 11 时面积随时间缩减，11 时至 14 时基本维持在谷值，处于平稳状态，14 时至 17 时面积随时间扩增，整体变化曲线为U形，即呈现先减后增的变化趋势。
2. 阴影面积随季节变化情况：春、秋、夏季阴影面积一天中变化不大，冬季早

晚与正午的阴影面积存在较大差值。此外夏季阴影面积最小，冬日阴影面积最大，春秋季阴影面积相近，随四季更迭整体呈现“减—增—减”的循环变化趋势。

5.2 问题二模型建立与求解

5.2.1 行向间苹果树树冠模型

在不同林带行向间，由于相邻树木之间的相互遮挡，树木树冠的受光及遮阴而形成阴影情况也会发生相应变化，经过相关查询、研究与评估，在此只对较为重要的两个影响因素——行列方向以及树与树之间的距离进行影响程度分析。

关于行列方向，此处对东—西走向与南—北走向两种不同方向的单行林带做出区分，分别进行相关分析后再比对。

关于树与树间距，则只需在问题一模型的基础上进行扩展，将受光实体改成以相同的数棵苹果树沿直线等间隔排列构建而成的单行林带。此处取树木数量均为3，树与树间距分别为$3.5m$，$5m$，$6.5m$的三种单行林带进行相关数据模拟。模型的具体结构如下图所示：

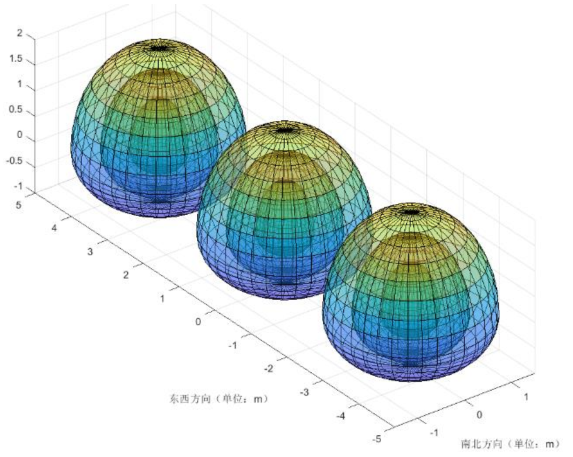
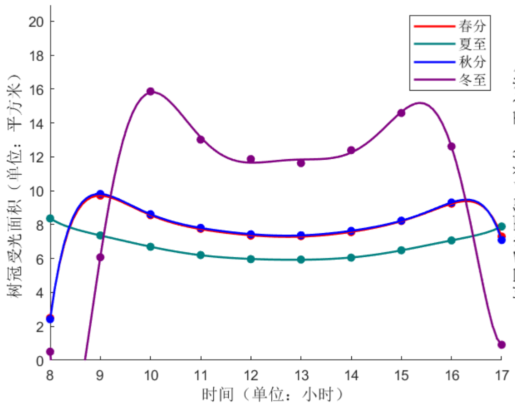


图 11 行向间苹果树树冠三维结构示意图

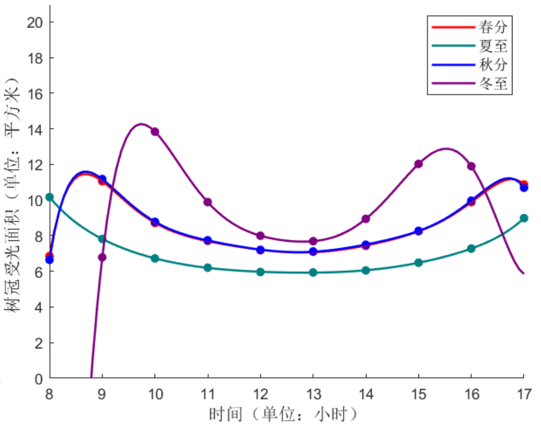
5.2.2 行向间苹果树树冠受光情况分析

由于排列的规律性以及太阳光的平行光性质，此处只需考虑中间树冠层的受光情况，最后再进行推广即可。

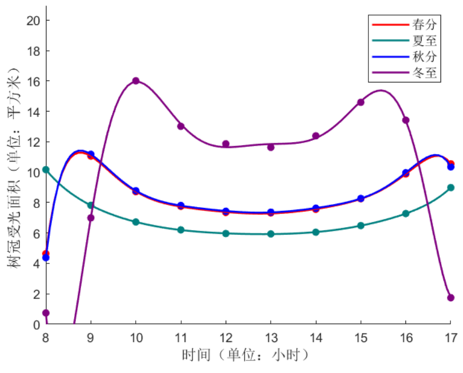
分东—西走向与南—北走向两组，每组以春分、夏至、秋分、冬至四日为例，以1小时为步长，在8时—17时的时间段内取点，使用5.1.3中所使用建立受光面的方式模拟该时间点该组单行林带受光情况，并计算受光面积。



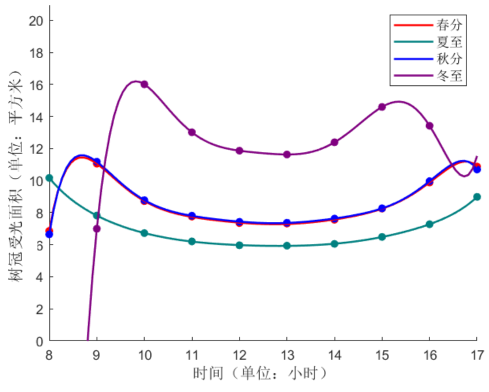
东西方向 3.5m 间距



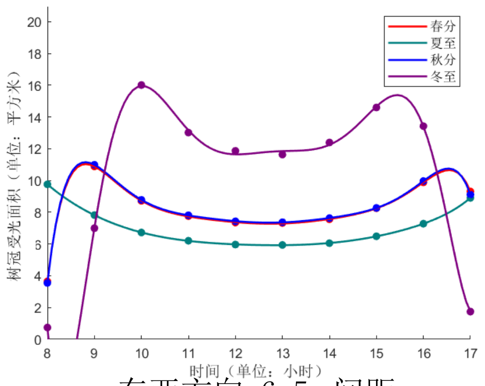
南北方向 3.5m 间距



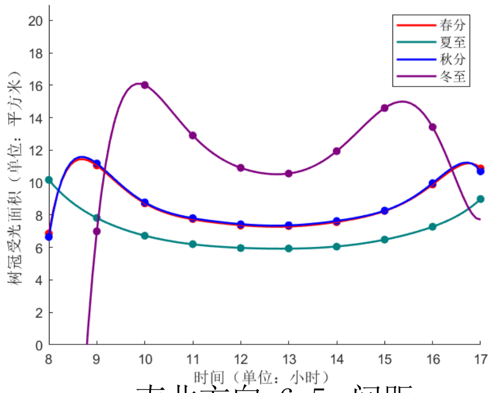
东西方向 5m 间距



南北方向 5m 间距



东西方向 6.5m 间距



南北方向 6.5m 间距

图 12 不同方向、间距的行向苹果树树冠受光面积随时间、季节变化的情况

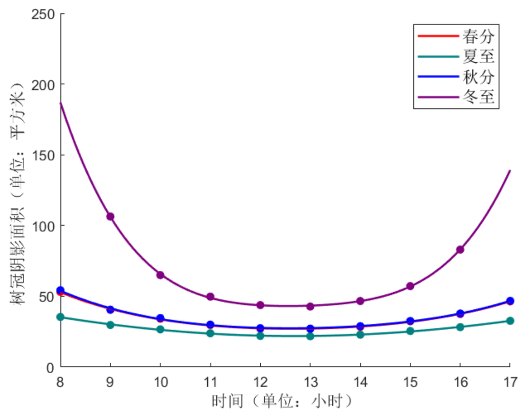
由对上六图的比对分析可得：

1. 不同于单个树冠层，在春、夏、秋三季的林带中，中间树冠层的受光面积在早晨和傍晚（8 时—9 时与 16 时—17 时两个时间段）存在极小值，随后会在短时间内急速升高抵达较大值，中间时间段与同时间段单个树冠层的受光情况规律近似。
2. 对于东—西走向的单行林带，仅在树与树间距离较近时，会对春、夏、秋三季早晚的受光面积产生一定影响，具体规律为三季早晚中间树冠层的受光面积随树间距缩减而减小。冬季不受影响
3. 对于南北走向的单行林带，冬季正午左右中间树冠层的受光面积会随距离缩短而减小。春、夏、秋三季不受影响

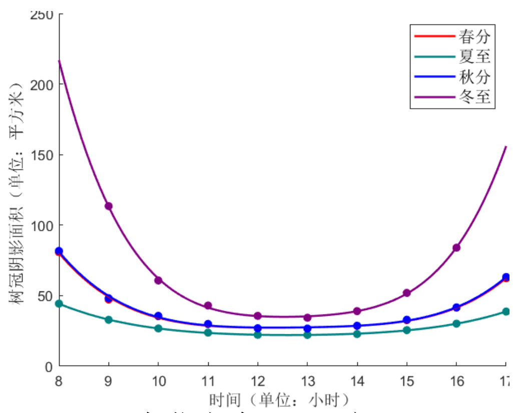
5.2.3 行向间苹果树树冠阴影情况分析

考虑到三个树冠层的阴影可能存在相互重叠的影响，此处选择对单行林带整体阴影进行分析。

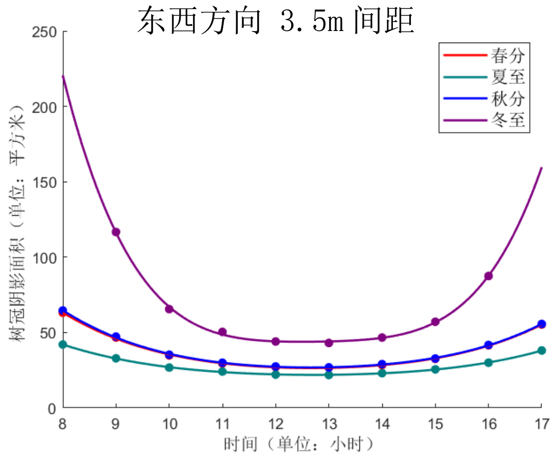
分出东—西走向与南—北走向两组，每组仍以春分、夏至、秋分、冬至四日为例，以 1 小时为步长，在 8 时—17 时的时间段内取点，使用 5.1.4 中所使用建立阴影方式模拟该时间点该组单行林带阴影并计算面积。其中冬至日由于 8 时和 17 时太阳高度角过低，在此不做模拟分析。按组将阴影面积值点拟合成曲线后，生成结果如下所示：



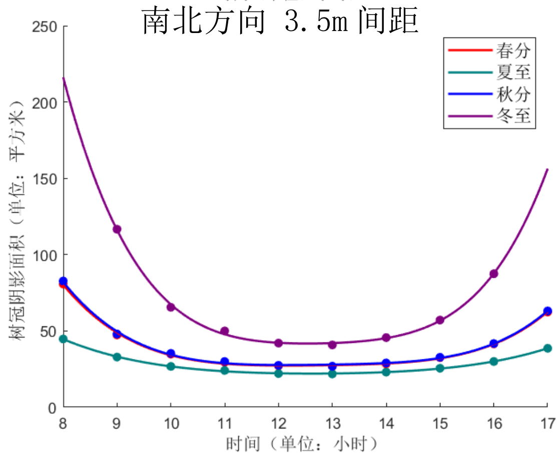
东西方向 3.5m 间距



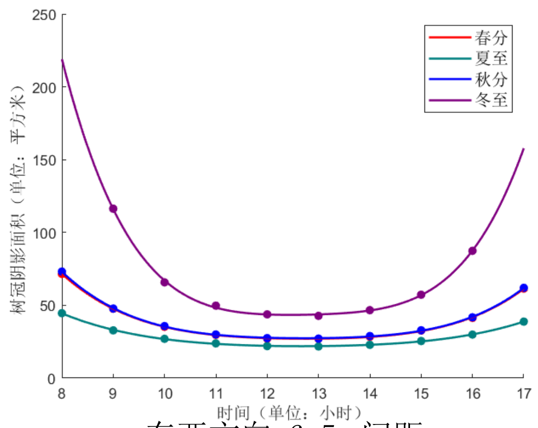
南北方向 3.5m 间距



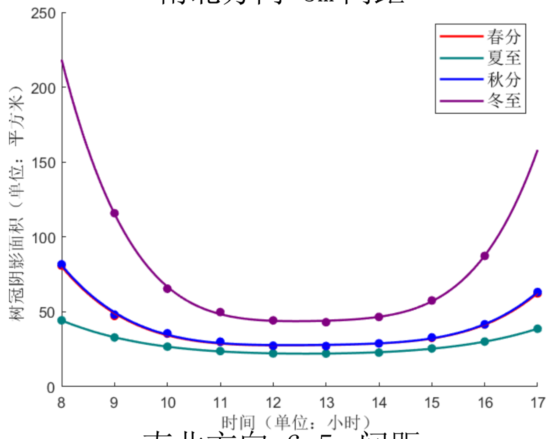
东西方向 5m 间距



南北方向 5m 间距



东西方向 6.5m 间距



南北方向 6.5m 间距

图 13 不同行间、间距的行向苹果树树冠阴影面积随时间、季节变化的情况

由对上六图的比对分析可得：

1. 无论间距大小与行列方向如何改变，单行林带整体阴影的变化趋势仍符合 5.1.4 中对于单个树冠层产生阴影的分析。详见上，在此不多赘述；
2. 对于东—西走向的单行林带而言，树与树间距离仅对早晨与傍晚（即春夏秋三季 8 时与 17 时两个时间点，冬季 9 时与 16 时两个时间点）时的阴影面积产生一定影响。在一定限度内，树间距越大，单行林带所产生的整体阴影面积越大；
3. 对于南—北走向的单行林带而言，树与树间距离对其产生的整体阴影面积几乎完全没有影响；

4. 单行林带的行列方向对于夏、冬两季其整体阴影面积影响不大，春、秋两季单行林带的整体阴影面积则呈现为南—北走向的大于东—西走向的。

5.3 问题三模型建立与求解

5.3.1 行列间苹果树树冠模型

相较起行向间树与树之间的遮挡关系而言，行列间的遮挡关系无疑更加复杂。在行向间苹果树树冠模型的基础上，将等距3样本单行林带拓展为东南西北正方向上的3×3的正方形林区，建立行列间苹果树树冠模型，考虑不同间距的树冠受光及阴影随时间、季节变化的情况。具体结构如下图所示：

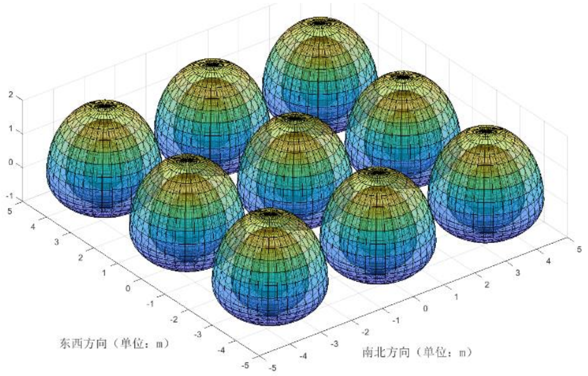
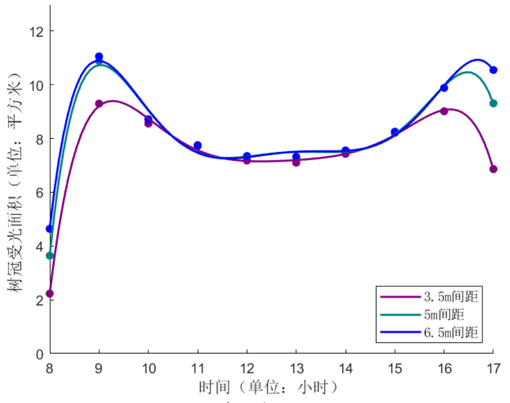


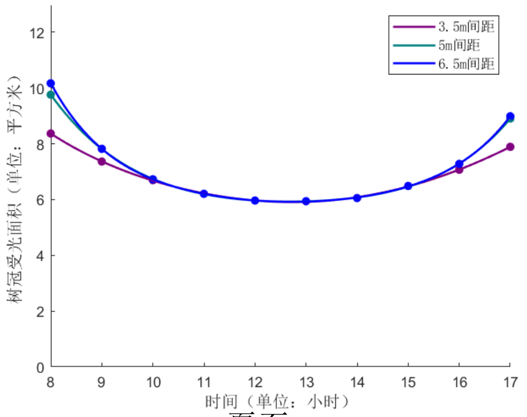
图 14 行列间苹果树树冠三维模型

5.3.2 行列间苹果树树冠受光情况分析

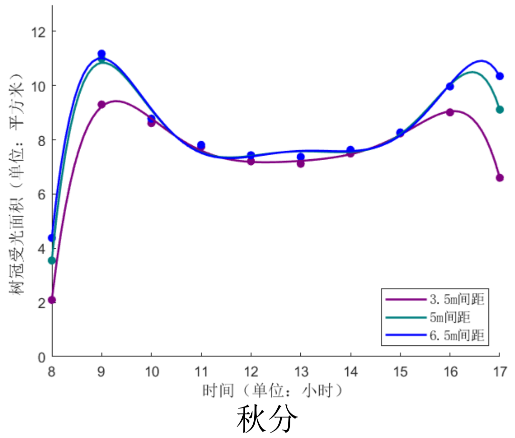
与5.2.2类似，仅考虑中间树冠层的受光情况，在春分、夏至、秋分、冬至四日以 1 小时为步长在 8 时—17 时时间段内取点，使用 5.1.3 中的建立方式模拟特定时间点树冠层受光并计算面积，数值拟合成曲线，生成结果如下所示：



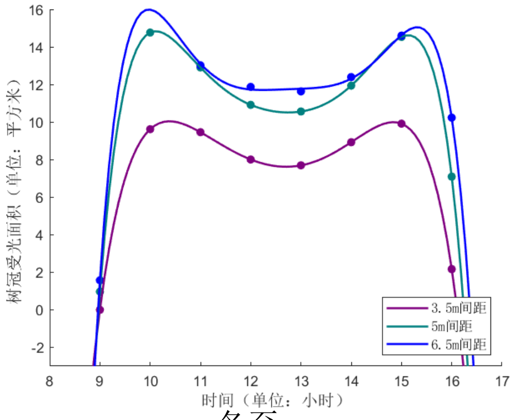
春分



夏至



秋分



冬至

图 15 不同间距的行列间苹果树树冠受光面积随时间、季节变化的情况

对上四图分析可得，相较于夏季的*U*形变化曲线，由于早晚太阳始末高度角均较低，彼时相邻树冠层间会存在大面积遮挡，春、秋、冬季的变化曲线更偏向鞍形，在早晚多出了一个短时大幅变化过程。树间距的扩大也只在四季的此时段内对受光面积的极值存在一定限度内的增大作用，其余时间段并无影响。

5.3.3 行列间苹果树树冠阴影情况分析

与5.2.3类似，考虑区块内整体阴影面积。依旧对春分、夏至、秋分、冬至四日以 1 小时为步长在 8 时—17 时时间段内取点，使用5.1.4中所使用建立阴影方式模拟特定时间点林区总阴影并计算面积。冬至日 8 时和 17 时在此不作模拟。将阴影面积值点拟合成曲线后，生成结果按季节分类，如下所示：

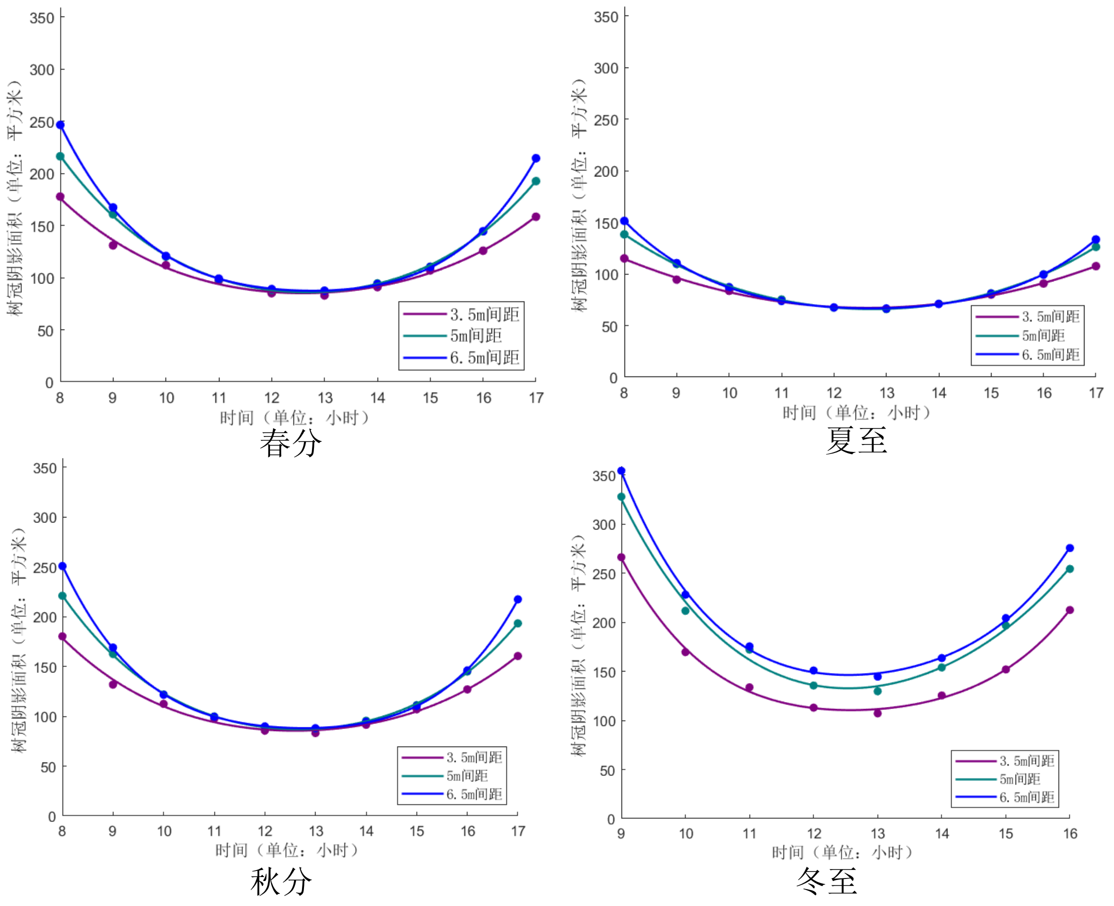


图 16 不同间距的行列间苹果树树冠阴影面积随时间、季节变化的情况

对上四图曲线进行比对分析可得：

1. 与单棵苹果树阴影情况类似，不同季节与不同间距均呈现阴影面积先减后增的变化趋势。春秋两季变化曲线近似甚至重合，相较于春、秋、冬三季，夏季变化较为平稳。

2. 不同树间距仅对早晚阴影面积产生影响，树间距越小阴影面积越少，正午时分不同树间距产生的阴影的面积均会处于一个相同的谷值。此值以夏季最小，春秋两季相同。而对于冬季而言，不同树间距对应谷值均不同，树间距越大，谷值越大。

5.4 问题四模型建立与求解

5.4.1 带坡度的行列间苹果树树冠模型

此问要求将问题三所建立模型运用在西北坡向，坡度为10°的陕北黄土高原地形之上。由于地面此时不再为水平面，需要对先前建立的行列间苹果树树冠模型进行一定改良。

仍考虑3×3的正方形林区，以中间树冠层的中心坐标轴原点，将坡面西北方向抬高，东南方向压低，使得整体坡面与水平面呈10°夹角。则东—西方向上与南—北方向上坡度（比）为：

$$i=\frac{\sqrt{2}}{2}\tan 10^\circ$$

根据设定的树间距d可计算正东（西）、正北（南）方向上相邻树冠层的高度差，具体方程为：

$$\Delta h=\frac{\sqrt{2}}{2}\tan 10^\circ\times d$$

将正方形林区内 9 个树冠层的z轴坐标用特征向量表示，结果为：

$$\begin{pmatrix} -\sqrt{2}\tan 10^\circ\times d & -\frac{\sqrt{2}}{2}\tan 10^\circ\times d & 0 \\ -\frac{\sqrt{2}}{2}\tan 10^\circ\times d & 0 & \frac{\sqrt{2}}{2}\tan 10^\circ\times d \\ 0 & \frac{\sqrt{2}}{2}\tan 10^\circ\times d & \sqrt{2}\tan 10^\circ\times d \end{pmatrix}$$

以此为基准，对行列间苹果树树冠模型中的各个样本进行位置上的改变，具体结构如下图所示：

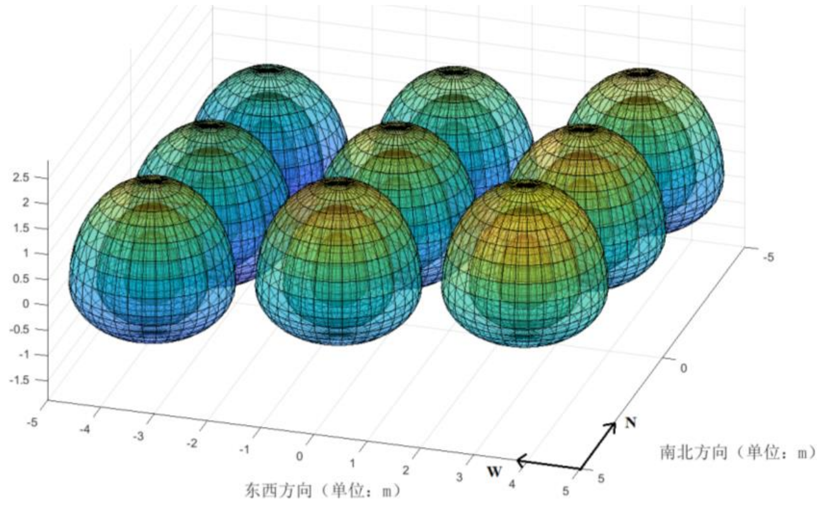


图 17 带坡度的行列间苹果树树冠三维模型图

仿照问题 3 中的方式，对此新模型重新进行模拟分析。在研究阴影面积时，需考虑到坡面与水平面之间的投影关系进行计算。

5.4.2 带坡度的行列间苹果树树冠受光情况分析

依旧以春分、夏至、秋分、冬至四日作为划分，考察中间的苹果树树冠受光面积，拟合完成后，将有坡度与5.3.2得出无坡度的结果置于同一张图中，以方便比对分析。结果如下所示：

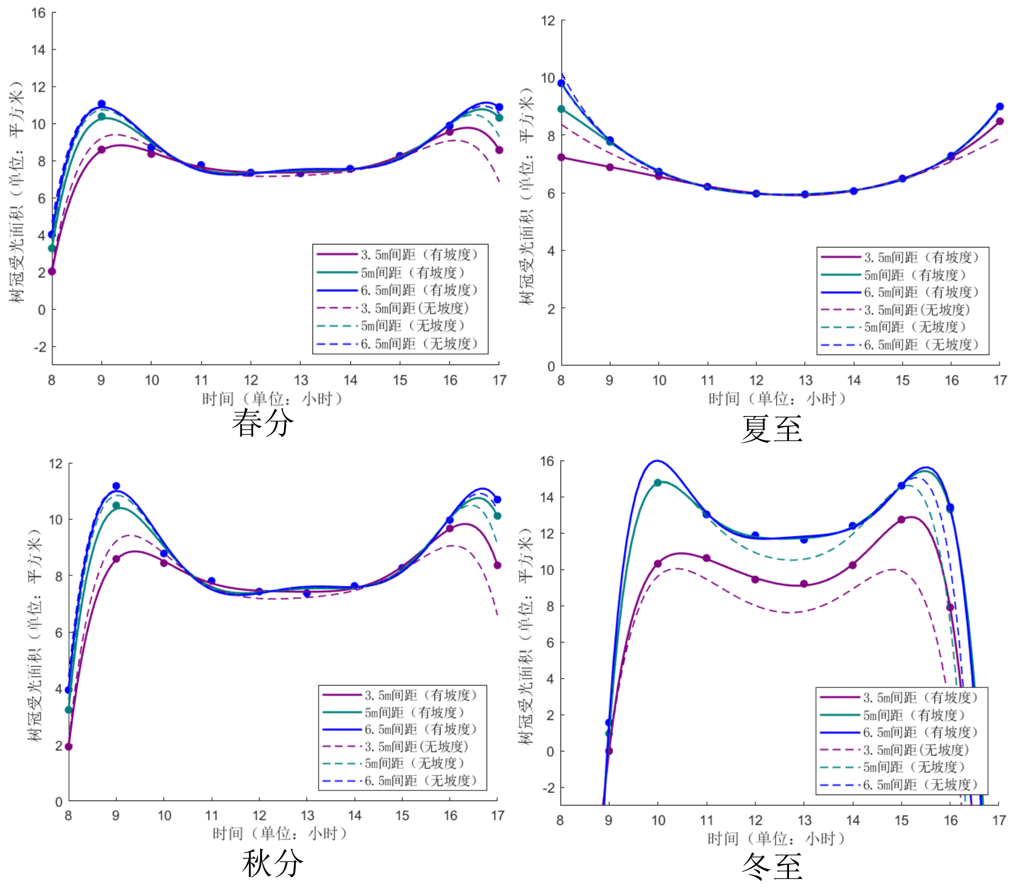
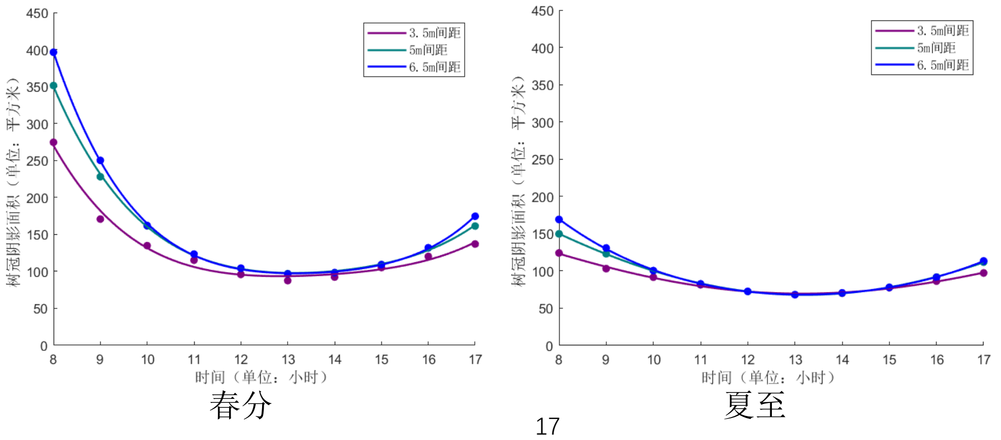


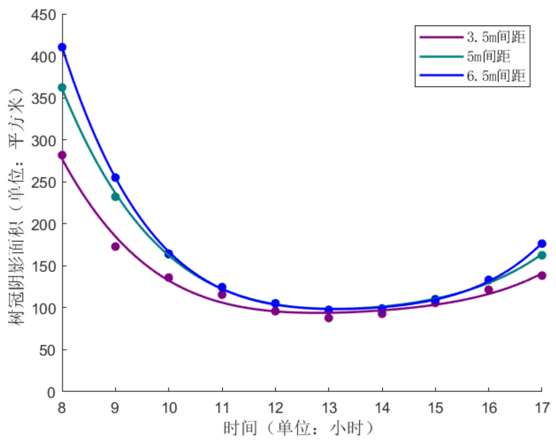
图 18 带坡度的不同间距行列间苹果树树冠受光面积随季节、时间变化的情况

图中可清晰看到，除5.3.2的结论在此处仍然适用外，坡度对树冠层受光情况确实存在一定影响。在春、夏、秋三季的早上（8 时—10 时），有坡度比无坡度的树冠层受光面积略小，考虑到可能是存在坡度使得近日方向树冠层偏高，对其背后的树冠层进行了更完全的遮挡。而三季傍晚（15 时—17 时）太阳方位偏转，近日方向树冠层处于较低位，使其身后的树冠层更充分暴露在阳光下。冬季处于比较特殊的位置，由于太阳高度角较低，阳光得以照到因高度差而露出的被遮挡树冠层的下半部分。在入射光线与角度较小时，纺锤形结构的下半部扁平特征决定了受光部分的表面积必大于上半部被遮挡部分的表面积，因此冬季各时间点均呈现为有坡度受光面积大于无坡度受光面积。

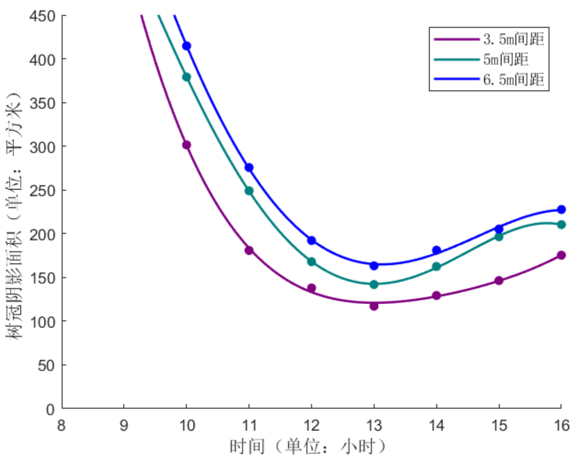
5.4.3 带坡度的行列间苹果树树冠阴影情况分析

依旧以春分、夏至、秋分、冬至四日作为划分，拟合结果如下所示





秋分



冬至

图 19 带坡度的不同间距行列间苹果树树冠阴影面积随季节、时间变化的情况

可以看到，坡度的抬升并不改变原行列间整体阴影面积的变化趋势，在春、秋、冬三季其只对初始阶段（8 时，冬至日为 9 时），也是阴影面积变化曲线抵达的极值做出提升，剩余时段不作影响。由于太阳高度角较大，树冠层投影集中在树周围小范围内，阴影面积也偏小，坡度的抬升在夏季更是对全过程都不产生影响。因为这只相当于将太阳光以一种比实际略低的高度角对林区进行照射，产生影响并不算大。

5.5 问题五的建立与求解

苹果为喜光性树种。光照充足，有利于正常生长和结果，提高果实的品质。如未能达到所需要的光照要求，会使苹果树出现枝叶软弱，果树的虫害增多，果树开花率低，果树本身的坐果率下降，果树的根须也会受到一定的影响，导致营养输送低，果实颜色不好，缺少光泽度。因此，林区的受光照情况，包括光照吸收效果、光能利用率等，是苹果林木栽植与管护中的重要考量因素。土壤肥力、水分等其他重要指标也同样需要考虑到。

此处将林区的整体受光面积与阴影面积作为评判其受光照情况的参考标准。受光面积越大，表明林区内果树光照吸收效果越好，阴影面积越大，说明林区的光能利用率越高。

根据以上模型建立与分析，将林区内**树间距**与**坡度**作为影响因素，代入问题中进行考量：

树间距：根据以上分析结果，理论上树间距越大，苹果树树冠层的受光面积越大，产生阴影面积也越大。但考虑到苹果林的经济效益属性，过大的树间距会造成林地资源浪费，反而会得不偿失。而过小的林间距会导致植株之间的资源竞争激烈，不利于树木生长，因此建议林间距应保持在适度且充足的范围内。此处建议以$4-5$米为佳，经查找资料，这也符合纺锤状树形经济林的标准株距。

坡度：根据以上分析结果，小范围内坡度的提升对春、秋、冬三季全天受光面积总额并无多大影响，在冬季会使全天受光面积总额有较大提升。对于阴影面积而言，坡度的提升会对四季每日初始阴影面积有所提升。除此之外，坡度带来的高度差使树木根系均匀分布在土壤不同深度处，能使土壤肥力与水分得到更充分的利用。经过资料查询，25°以上坡度的土

壤开始产生严重的水土流失，考虑到黄土高原土质本身较为松软，此处建议在5°~10°为佳。

除此之外，合适的树形与科学的修剪方案也是高质量经济林建设中不可或缺的一环，经相关资料[4]查找，本文中所使用的纺锤形在所有现有苹果树改造树形中透光率最差，即光线吸收效果最好，光能利用率最高，此处建议设为树形修剪目标。在此基础之上，再结合树木具体生长情况，合理修剪虫枝、病枝、过密枝，保证植株健康以及树体结构合理性，强化浇水、施肥管理工作，提升苹果经济林产业效益以及林木质量，推动经济林可持续发展。

六、 模型评价

6.1 模型的优点

本文建立的模型有以下优点：

1. 在考虑太阳光线和苹果树树冠之间关系时，利用光线追踪的方式，使得模型更利于理解，同时模型具有普适性；
2. 在考虑苹果树树冠受光面积时，利用曲面积分来进行计算，结果准确性较高；
3. 在考虑苹果树树冠遮阴而形成的阴影在坡面上的面积时，利用投影面来进行计算，准确性较高。

6.2 模型的缺点

本文建立的模型也有以下缺点：

1. 在利用光线追踪的方式来考虑苹果树树冠的受光和阴影情况时，代码运行时间较长，需要优化；
2. 本文在问题一中建立了加权阴影深度值S来考虑阴影的深度，但由于时间紧迫和篇幅所限，在后续的问题中没有充分考虑此因素；
3. 苹果树的树冠结构可能会随着季节的变化发生改变，本文假设了其树冠结构不会发生改变；
4. 本文仅考虑了苹果树树冠结构为纺锤形时的受光和阴影情况。

6.3 模型的推广

本文仅考虑了纺锤形的树冠结构，若需要进一步改进模型，可考虑更多类型的树冠结构和树冠大小。此外，可进一步考虑加权阴影深度值的影响，以及太阳光照强度因素的影响。这样，能更加全面的考察经济林树木栽种方式与最优光照分布，以促进地生态环境与社会经济长远稳定的持续发展。

参考文献

[1] 习近平新时代中国特色社会主义思想学习纲要/中共中央宣传部编.--北京：学习出版社，2019.6 ISBN 978-7-5147-0921-6

[2] 黄土高原植被建设应从扩大面积向提升质量转变.科技导报. https://www.sohu.com/a/247052344_650021

[3] 许华森,孙志梅.乔化稀植苹果树冠投影的理论模型研究[J].河北农业大学学报,2021,44(3):41-46,95. DOI:10.13320/j.cnki.jauh.2021.0045.

[4] 白岗栓,杜社妮,王建平. 陕北山地苹果树形改造研究. 中国农业大学学报 202110.11841/j.issn.1007-4333.2021.12.06

附录

附录 A 支撑材料列表

附件 1 apple_tree_model.m（苹果树树冠三维模型，MATLAB 代码）
附件 2 single_apple_tree_model_light.m（单株苹果树树冠受光模型，MATLAB 代码）
附件 3 single_apple_tree_model_light_area.m（单株苹果树树冠受光面积曲线拟合模型，MATLAB 代码）
附件 4 single_apple_tree_model_dark.m（单株苹果树树冠阴影模型，MATLAB 代码）
附件 5 single_apple_tree_model_dark_area.m（单株苹果树树冠阴影面积拟合模型，MATLAB 代码）
附件 6 triple_tree_model.m（行向间苹果树树冠三维模型，MATLAB 代码）

```
clc,clear
```

附件 7 triple_tree_model_light.m（行向间苹果树树冠受光模型，MATLAB 代码）
附件 8 triple_tree_model_light_area.m（行向间苹果树树冠受光面积拟合模型，MATLAB 代码）
附件 9 triple_tree_model_dark.m（行向间苹果树树冠阴影模型，MATLAB 代码）
附件 10 triple_tree_model_dark_area.m（行向间苹果树树冠阴影面积拟合模型，MATLAB 代码）
附件 11 trees_model.m（行列间苹果树树冠三维模型，MATLAB 代码）
附件 12 trees_model_light.m（行列间苹果树树冠受光模型，MATLAB 代码）

```
clc,clear
```

附件 13 trees_model_light_area.m（行列间苹果树树冠受光面积拟合模型，MATLAB 代码）
附件 14 trees_model_dark.m（行列间苹果树树冠阴影模型，MATLAB 代码）

```
clc,clear
```

附件 15 trees_model_dark_area.m（行列间苹果树树冠阴影面积拟合模型，MATLAB 代码）
附件 16 trees_model_10.m（带坡度的行列间苹果树树冠三维模型，MATLAB 程序）
附件 17 trees_model_10_light.m（带坡度的行列间苹果树树冠受光模型，MATLAB 代码）
附件 18 trees_model_10_light_area.m（带坡度的行列间苹果树树冠受光面积拟合模型，MATLAB 代码）
附件 19 trees_model_10_dark.m（带坡度的行列间苹果树树冠阴影模型，MATLAB 程序）
附件 20 trees_model_10_dark_area.m（带坡度的行列间苹果树树冠阴影面积拟合模型，MATLAB 程序）

附录 B 支撑材料

附件 1 apple_tree_model.m（苹果树树冠三维模型，MATLAB 代码）

```
clc,clear
%树冠可视化模型
[theta,phi] = meshgrid(0:10:360,0:10:90);
X = 1.5*cos(phi*pi/180).*cos(theta*pi/180);
Y = 1.5*cos(phi*pi/180).*sin(theta*pi/180);
Z = 2*sin(phi*pi/180);
ss1=surf(X, Y, Z);
ss1.FaceAlpha=0.4;
hold on;
[theta,phi2] = meshgrid(0:10:360,-90:10:0);
X2 = 1.5*cos(phi2*pi/180).*cos(theta.*pi/180);
Y2 = 1.5*cos(phi2*pi/180).*sin(theta*pi/180);
Z2 = 1*sin(phi2*pi/180);
ss2=surf(X2, Y2, Z2);
ss2.FaceAlpha=0.4;
hold on;
[theta,phi] = meshgrid(0:10:360,0:10:90);
X3 = 1*cos(phi*pi/180).*cos(theta*pi/180);
Y3 = 1*cos(phi*pi/180).*sin(theta*pi/180);
Z3 = 1.5*sin(phi*pi/180);
ss3=surf(X3, Y3, Z3);
ss3.FaceAlpha=0.4;
hold on;
X4 = 1*cos(phi2*pi/180).*cos(theta.*pi/180);
Y4 = 1*cos(phi2*pi/180).*sin(theta*pi/180);
Z4 = 0.75*sin(phi2*pi/180);
ss4=surf(X4, Y4, Z4);
ss4.FaceAlpha=0.4;
hold on;
X5 = 0.5*cos(phi*pi/180).*cos(theta*pi/180);
Y5 = 0.5*cos(phi*pi/180).*sin(theta*pi/180);
Z5 = 1*sin(phi*pi/180);
ss5=surf(X5, Y5, Z5);
ss5.FaceAlpha=0.4;
hold on;
X6 = 0.5*cos(phi2*pi/180).*cos(theta.*pi/180);
Y6 = 0.5*cos(phi2*pi/180).*sin(theta*pi/180);
Z6 = 0.375*sin(phi2*pi/180);
ss6=surf(X6, Y6, Z6);
ss6.FaceAlpha=0.4;

axis equal
```

附件 2 single_apple_tree_model_light.m（单株苹果树树冠受光模型，MATLAB 代码）

```
clc,clear
%树冠可视化模型
[theta,phi] = meshgrid(0:10:360,0:10:90);
X11 = 1.5*cos(phi*pi/180).*cos(theta*pi/180);
Y11 = 1.5*cos(phi*pi/180).*sin(theta*pi/180);
Z11 = 2*sin(phi*pi/180);
ss11=surf(X11, Y11, Z11);
ss11.FaceAlpha=0.4;
ss11.FaceColor = [0.5 0.5 0.5];
hold on;
[theta,phi2] = meshgrid(0:10:360,-90:10:0);
X12 = 1.5*cos(phi2*pi/180).*cos(theta.*pi/180);
Y12 = 1.5*cos(phi2*pi/180).*sin(theta*pi/180);
Z12 = 1*sin(phi2*pi/180);
ss12=surf(X12, Y12, Z12);
ss12.FaceAlpha=0.4;
ss12.FaceColor = [0.5 0.5 0.5];
hold on;

%树冠模型参数
a11=1.5;b11=1.5;c11=2;
a12=1.5;b12=1.5;c12=1;
center1=[0,0,0];
%两个角度
elevation_angle=[15.253    46.078  48.806  19.927  28.868  64.460  68.721
    33.650  15.003  45.728  48.437  19.670   10.438 25.411  27.300  14.029];
azimuth_angle=[102.306 141.640 210.452 253.746 80.961  113.631 237.557
    275.847 102.634 141.929 210.204 253.415 129.748 154.068 200.161 225.982];
for s=1:1:16
    azimuth_angle(s)=180-azimuth_angle(s);
end
    % 将角度转换为弧度
    elevation_rad = deg2rad(elevation_angle(1));
    azimuth_rad = deg2rad(azimuth_angle(1));

    % 光线方向向量
    direction_vector = [cos(azimuth_rad) * cos(elevation_rad);
                   sin(azimuth_rad) * cos(elevation_rad);
                   sin(elevation_rad)];
    %光线参数
    point1=[0,0,-2.5];
    point22=point1+direction_vector';
    p=1;
    while point22(3)<5
```

```
        point22=point22+direction_vector';
        p=p+1;
    end

    direction=point22-point1;
    num=0;
%步长计算
S=0;
numIntersectionPoints1=0;
for x =-10:0.1:10
    for y =-40:0.1:20
        point1=[0,0,-2.5];
        point1=point1+[x,0,0]+[0,y,0];
        point2=point22+[x,0,0]+[0,y,0];
        %plot3([point1(1),point2(1)],[point1(2),point2(2)],[point1(2),point2
(2)])
        Bool = 0;
        for t = 0:0.01:1  % 步长为0.01
            point = point2 - t * direction;
            % 判断点是否在半椭球内
            if point(3)>=0
                if (((point(1)-center1(1))/a11)^2 + ((point(2)-
center1(2))/b11)^2 + ((point(3)-center1(3))/c11)^2 <= 1) & Bool == 0
                    numIntersectionPoints1 = numIntersectionPoints1 + 1;
                    Bool = 1;
                    scatter3([point(1)], [point(2)],[point(3)],'filled',
'MarkerFaceColor','y','MarkerFaceAlpha', 1);
                    A = sqrt((point(1)/a11^2)^2 + (point(2)/b11^2)^2 +
(point(3)/c11^2)^2);
                    sintheta=(point(2)/b11^2)/A;
                    sintheta=sqrt(1-sintheta^2);
                    S=S+0.01*abs(sintheta);
                end
            elseif point(3)<0
                if (((point(1)-center1(1))/a12)^2 + ((point(2)-
center1(2))/b12)^2 + ((point(3)-center1(3))/c12)^2 <= 1) & Bool == 0
                    numIntersectionPoints1 = numIntersectionPoints1 + 1;
                    Bool = 1;
                    scatter3([point(1)], [point(2)],[point(3)],'filled',
'MarkerFaceColor','y','MarkerFaceAlpha', 1);
                    A = sqrt((point(1)/a12^2)^2 + (point(2)/b12^2)^2 +
(point(3)/c12^2)^2);
                    sintheta=(point(2)/b12^2)/A;
                    sintheta=sqrt(1-sintheta^2);
                    S=S+0.01*abs(sintheta);
                end
            end
        end

        hold on;

    end
end
disp(numIntersectionPoints1)
disp(S)
xlabel('南北方向（单位：m）','FontSize',12,'FontName','宋体');
ylabel('东西方向（单位：m）','FontSize',12,'FontName','宋体');
view(90,90)
axis equal;
```

附件 3 single_apple_tree_model_light_area.m（单株苹果树树冠受光面积曲线拟合模型，MATLAB 代码）

```
y1 = [18.9801
11.3502
8.8233
7.7895
7.3577
7.3253
7.5792
8.3211
10.0851
14.7158];
y2 = [10.4491
7.9258
6.7594
6.2143
5.9664
5.9419
6.0624
6.5068
7.3459
9.1783];
y3 = [19.2739
11.4861
8.8863
7.8453
7.4391
7.3775
7.653
8.3368
10.1594
14.8667
];
y4 = [20.6738
    17.7271
15.7093
13.1685
11.9353
11.6851
12.4989
14.8676
16.3198
19.4376];
x = [8 9 10 11 12 13 14 15 16 17];
x2=[8 9 10 11 12 13 14 15 16 17];

y1=y1';
y2=y2';
y3=y3';
y4=y4';

% 设置多项式阶数
n = 4;

% 进行四条曲线的曲线拟合
p1 = polyfit(x, y1, n);
p2 = polyfit(x, y2, n);
p3 = polyfit(x, y3, n);
p4 = polyfit(x2, y4, n);

% 生成一系列 x 值
x_fit = linspace(min(x), max(x), 100);

% 计算四条拟合曲线对应的 y 值
y_fit1 = polyval(p1, x_fit);
y_fit2 = polyval(p2, x_fit);
y_fit3 = polyval(p3, x_fit);
y_fit4 = polyval(p4, x_fit);

% 绘制原始数据和拟合曲线
figure;
hold on;
```

```
scatter(x, y1, 'o','filled','MarkerFaceColor','r');
scatter(x, y2, 'o', 'filled','MarkerFaceColor',[0,0.5,0.5]);
scatter(x, y3, 'o', 'filled','MarkerFaceColor','b');
scatter(x2, y4, 'o', 'filled','MarkerFaceColor',[0.5,0,0.5]);
plot(x_fit, y_fit1,'color','r','LineWidth',1.5);
plot(x_fit, y_fit2,'color',[0,0.5,0.5],'LineWidth',1.5);
plot(x_fit, y_fit3,'color','b','LineWidth',1.5);
plot(x_fit, y_fit4,'color',[0.5,0,0.5],'LineWidth',1.5);
hold off;
xlabel('时间（单位：小时）','FontSize',12,'FontName','宋体');
ylabel('树冠受光面积（单位：平方米）','FontSize',12,'FontName','宋体');
title('苹果树树冠受光面积随季节、时间的变化情况','FontSize',16,'FontName','宋体
');
legend('','','','', '春分', '夏至', '秋分', '冬至
','FontSize',12,'FontName','宋体');
```

附件 4 single_apple_tree_model_dark.m（单株苹果树树冠阴影模型，MATLAB 代码）

```
clc,clear
%树冠可视化模型
[theta,phi] = meshgrid(0:10:360,0:10:90);
X = 1.5*cos(phi*pi/180).*cos(theta*pi/180);
Y = 1.5*cos(phi*pi/180).*sin(theta*pi/180);
Z = 2*sin(phi*pi/180);
ss1=surf(X, Y, Z);
ss1.FaceAlpha=0.4;
hold on;
[theta,phi2] = meshgrid(0:10:360,-90:10:0);
X2 = 1.5*cos(phi2*pi/180).*cos(theta.*pi/180);
Y2 = 1.5*cos(phi2*pi/180).*sin(theta*pi/180);
Z2 = 1*sin(phi2*pi/180);
ss2=surf(X2, Y2, Z2);
ss2.FaceAlpha=0.4;
hold on;
[theta,phi] = meshgrid(0:10:360,0:10:90);
X3 = 1*cos(phi*pi/180).*cos(theta*pi/180);
Y3 = 1*cos(phi*pi/180).*sin(theta*pi/180);
Z3 = 1.5*sin(phi*pi/180);
ss3=surf(X3, Y3, Z3);
ss3.FaceAlpha=0.4;
hold on;
X4 = 1*cos(phi2*pi/180).*cos(theta.*pi/180);
Y4 = 1*cos(phi2*pi/180).*sin(theta*pi/180);
Z4 = 0.75*sin(phi2*pi/180);
```

```
ss4=surf(X4, Y4, Z4);
ss4.FaceAlpha=0.4;
hold on;
X5 = 0.5*cos(phi*pi/180).*cos(theta*pi/180);
Y5 = 0.5*cos(phi*pi/180).*sin(theta*pi/180);
Z5 = 1*sin(phi*pi/180);
ss5=surf(X5, Y5, Z5);
ss5.FaceAlpha=0.4;
hold on;
X6 = 0.5*cos(phi2*pi/180).*cos(theta.*pi/180);
Y6 = 0.5*cos(phi2*pi/180).*sin(theta*pi/180);
Z6 = 0.375*sin(phi2*pi/180);
ss6=surf(X6, Y6, Z6);
ss6.FaceAlpha=0.4;

%树冠模型参数
a11=1.5;b11=1.5;c11=2;a21=1;b21=1;c21=1.5;a31=0.5;b31=0.5;c31=1;
a12=1.5;b12=1.5;c12=1;a22=1;b22=1;c22=0.75;a32=0.5;b32=0.5;c32=0.375;
center=[0,0,0];

elevation_angle=[10.43771148
10.43771148
18.92008084
25.41121023
29.24919056
29.9130712
27.30034527
21.79243076
14.02900813
14.02900813
];
azimuth_angle=[129.7484118
129.7484118
140.9571165
154.0680759
168.9518354
184.7639009
200.1609288
214.0333748
225.9820887
225.9820887
];
elevation_angle=elevation_angle';
azimuth_angle=azimuth_angle';
```

```
for s=1:1:10
    azimuth_angle(s)=180-azimuth_angle(s);
end

for n=1:1:10
    disp(['阴影点个数为：',num2str(n)])
    % 将角度转换为弧度
    elevation_rad = deg2rad(elevation_angle(n));
    azimuth_rad = deg2rad(azimuth_angle(n));

    % 光线方向向量
    direction_vector = [cos(azimuth_rad) * cos(elevation_rad);
                   sin(azimuth_rad) * cos(elevation_rad);
                   sin(elevation_rad)];
    %光线参数
    point1=[0,0,-2.5];
    point22=point1+direction_vector';
    p=1;
    while point22(3)<5
        point22=point22+direction_vector';
        p=p+1;
    end

    direction=point22-point1;
    num=0;
    %步长计算
    for x =-50:0.2:50
        for y =-200:0.2:20
            point1=[0,0,-2.5];
            point1=point1+[x,0,0]+[0,y,0];
            point2=point22+[x,0,0]+[0,y,0];
            numIntersectionPoints1=0;
            numIntersectionPoints2=0;
            numIntersectionPoints3=0;
            direction=point2-point1;
            for t = 0:0.01:1  % 步长为 0.01
                point = point1 + t * direction;
                % 判断点是否在半椭球内
                if point(3)>=0
                    if ((point(1)-center(1))/a31)^2 + ((point(2)-
center(2))/b31)^2 + ((point(3)-center(3))/c31)^2 <= 1
                        numIntersectionPoints3 = numIntersectionPoints3 + 1;
                    elseif ((point(1)-center(1))/a21)^2 + ((point(2)-
center(2))/b21)^2 + ((point(3)-center(3))/c21)^2 <= 1
```

```
                    numIntersectionPoints2 = numIntersectionPoints2 + 1;
                elseif ((point(1)-center(1))/a11)^2 + ((point(2)-
center(2))/b11)^2 + ((point(3)-center(3))/c11)^2 <= 1
                    numIntersectionPoints1 = numIntersectionPoints1 + 1;
                end
            elseif point(3)<0
                if ((point(1)-center(1))/a32)^2 + ((point(2)-
center(2))/b32)^2 + ((point(3)-center(3))/c32)^2 <= 1
                    numIntersectionPoints3 = numIntersectionPoints3 + 1;
                elseif ((point(1)-center(1))/a22)^2 + ((point(2)-
center(2))/b22)^2 + ((point(3)-center(3))/c22)^2 <= 1
                    numIntersectionPoints2 = numIntersectionPoints2 + 1;
                elseif ((point(1)-center(1))/a12)^2 + ((point(2)-
center(2))/b12)^2 + ((point(3)-center(3))/c12)^2 <= 1
                    numIntersectionPoints1 = numIntersectionPoints1 + 1;
                end
            end
        end

sum=numIntersectionPoints3*0.2+numIntersectionPoints2*0.8+numIntersectionPo
ints1*0.5;

        if sum>0
            % 平面方程
            % 假设平面方程为 Ax + By + Cz + D = 0
            A = 0;
            B = 0;
            C = 1;
            D = 2.5;

            syms m
            point = point1 + m * direction;
            eqn = A * point(1) + B * point(2) + C * point(3) + D == 0;
            t_sol = solve(eqn, m);

        % 计算交点坐标

            intersectionPoints = point1 + double(t_sol) * direction;
            if sum>12
                scatter3([intersectionPoints(1)],
[intersectionPoints(2)],[intersectionPoints(3)],'filled',
'MarkerFaceColor','k','MarkerFaceAlpha', 0.8);
                num=num+1;
            elseif sum>6
```

```
                    scatter3([intersectionPoints(1)],
[intersectionPoints(2)],[intersectionPoints(3)],'filled',
'MarkerFaceColor','k','MarkerFaceAlpha', 0.5);
                    num=num+1;
                elseif sum>0
                    scatter3([intersectionPoints(1)],
[intersectionPoints(2)],[intersectionPoints(3)],'filled',
'MarkerFaceColor','k','MarkerFaceAlpha', 0.2);
                    num=num+1;
                end
                %disp(['阴影程度为: ',num2str(sum)])

                hold on;
            end

        end
    end
    disp(['阴影点个数为: ',num2str(num)])
end

axis equal;
```

附件 5 single_apple_tree_model_dark_area.m（单株苹果树树冠阴影面积拟合模型，MATLAB 代码）

```
y1 = [1346 790 580 492 452 446 474 542 692 1040] * 0.02;
y2 = [746 548 446 402 368 364 384 426 500 644] * 0.02;
y3 = [1378 800 588 500 458 448 484 546 696 1052] * 0.02;
y4 = [1944 1092 840 734 718 778 952 1458] * 0.02;
x = [8 9 10 11 12 13 14 15 16 17];
x2=[9 10 11 12 13 14 15 16];

% 设置多项式阶数
n = 4;

% 进行四条曲线的曲线拟合
p1 = polyfit(x, y1, n);
p2 = polyfit(x, y2, n);
p3 = polyfit(x, y3, n);
p4 = polyfit(x2, y4, n);

% 生成一系列 x 值
x_fit = linspace(min(x), max(x), 100);

% 计算四条拟合曲线对应的 y 值
```

```
y_fit1 = polyval(p1, x_fit);
y_fit2 = polyval(p2, x_fit);
y_fit3 = polyval(p3, x_fit);
y_fit4 = polyval(p4, x_fit);

% 绘制原始数据和拟合曲线
figure;
hold on;
scatter(x, y1, 'o','filled','MarkerFaceColor','r');
scatter(x, y2, 'o', 'filled','MarkerFaceColor',[0,0.5,0.5]);
scatter(x, y3, 'o', 'filled','MarkerFaceColor','b');
scatter(x2, y4, 'o', 'filled','MarkerFaceColor',[0.5,0,0.5]);
plot(x_fit, y_fit1,'color','r','LineWidth',1.5);
plot(x_fit, y_fit2,'color',[0,0.5,0.5],'LineWidth',1.5);
plot(x_fit, y_fit3,'color','b','LineWidth',1.5);
plot(x_fit, y_fit4,'color',[0.5,0,0.5],'LineWidth',1.5);
hold off;
xlabel('时间（单位：小时）','FontSize',12,'FontName','宋体');
ylabel('树冠阴影面积（单位：平方米）','FontSize',12,'FontName','宋体');
title('苹果树树冠阴影面积随季节、时间的变化情况','FontSize',16,'FontName','宋体
');
legend('','','','', '春分', '夏至', '秋分', '冬至
','FontSize',12,'FontName','宋体');
```

附件 6 triple_tree_model.m（行向间苹果树树冠三维模型，MATLAB 代码）

```
clc,clear
%树冠 1 可视化模型
for u=-1:1:1
    [theta,phi] = meshgrid(0:10:360,0:10:90);
    X = 1.5*cos(phi*pi/180).*cos(theta*pi/180);
    Y = 1.5*cos(phi*pi/180).*sin(theta*pi/180)+u*3.5;
    Z = 2*sin(phi*pi/180);
    ss1=surf(X, Y, Z);
    ss1.FaceAlpha=0.4;
    hold on;
    [theta,phi2] = meshgrid(0:10:360,-90:10:0);
    X2 = 1.5*cos(phi2*pi/180).*cos(theta.*pi/180);
    Y2 = 1.5*cos(phi2*pi/180).*sin(theta*pi/180)+u*3.5;
    Z2 = 1*sin(phi2*pi/180);
    ss2=surf(X2, Y2, Z2);
    ss2.FaceAlpha=0.4;
    hold on;
    [theta,phi] = meshgrid(0:10:360,0:10:90);
    X3 = 1*cos(phi*pi/180).*cos(theta*pi/180);
    Y3 = 1*cos(phi*pi/180).*sin(theta*pi/180)+u*3.5;
    Z3 = 1.5*sin(phi*pi/180);
    ss3=surf(X3, Y3, Z3);
    ss3.FaceAlpha=0.4;
    hold on;
    X4 = 1*cos(phi2*pi/180).*cos(theta.*pi/180);
    Y4 = 1*cos(phi2*pi/180).*sin(theta*pi/180)+u*3.5;
    Z4 = 0.75*sin(phi2*pi/180);
    ss4=surf(X4, Y4, Z4);
    ss4.FaceAlpha=0.4;
    hold on;
    X5 = 0.5*cos(phi*pi/180).*cos(theta*pi/180);
    Y5 = 0.5*cos(phi*pi/180).*sin(theta*pi/180)+u*3.5;
    Z5 = 1*sin(phi*pi/180);
    ss5=surf(X5, Y5, Z5);
    ss5.FaceAlpha=0.4;
    hold on;
    X6 = 0.5*cos(phi2*pi/180).*cos(theta.*pi/180);
    Y6 = 0.5*cos(phi2*pi/180).*sin(theta*pi/180)+u*3.5;
    Z6 = 0.375*sin(phi2*pi/180);
    ss6=surf(X6, Y6, Z6);
    ss6.FaceAlpha=0.4;
    hold on;
end

xlabel('南北方向（单位：m）')
ylabel('东西方向（单位：m）')
axis equal
```

附件 7 triple_tree_model_light.m（行向间苹果树树冠受光模型，MATLAB 代码）

```
clc,clear
for d = 3.5:1.5:6.5

%树冠模型参数
a11=1.5;b11=1.5;c11=2;
a12=1.5;b12=1.5;c12=1;
center1=[0,0,0];

% 树冠模型参数
a31 = 1.5; b31 = 1.5; c31 = 2;
a32 = 1.5; b32 = 1.5; c32 = 1;
center3 = [0, -d, 0];  % 将中心点在 x 轴上移动 5 个单位
```

```
% 树冠模型参数
a21 = 1.5; b21 = 1.5; c21 = 2;
a22 = 1.5; b22 = 1.5; c22 = 1;
center2 = [0, d, 0];  % 将中心点在 x 轴上移动 5 个单位

% 树冠模型参数
a41 = 1.5; b41 = 1.5; c41 = 2;
a42 = 1.5; b42 = 1.5; c42 = 1;
center4 = [d, 0, 0];  % 将中心点在 x 轴上移动 5 个单位

% 树冠模型参数
a51 = 1.5; b51 = 1.5; c51 = 2;
a52 = 1.5; b52 = 1.5; c52 = 1;
center5 = [-d, 0, 0];  % 将中心点在 x 轴上移动 5 个单位

% 树冠模型参数
a61 = 1.5; b61 = 1.5; c61 = 2;
a62 = 1.5; b62 = 1.5; c62 = 1;
center6 = [-d, -d, 0];  % 将中心点在 x 轴上移动 5 个单位

% 树冠模型参数
a71 = 1.5; b71 = 1.5; c71 = 2;
a72 = 1.5; b72 = 1.5; c72 = 1;
center7 = [d, -d, 0];  % 将中心点在 x 轴上移动 5 个单位

% 树冠模型参数
a91 = 1.5; b91 = 1.5; c91 = 2;
a92 = 1.5; b92 = 1.5; c92 = 1;
center9 = [d, d, 0];  % 将中心点在 x 轴上移动 5 个单位



%两个角度
elevation_angle=[15.25329928
26.74728536
37.30069887
46.07805656
51.73268607
52.76041559
48.80598458
41.08804954
31.11833248
19.92684826
28.8681958
```

```
40.86994525
52.88480367
64.46002994
74.10822164
76.37468044
68.72145662
57.6097592
45.69154488
33.64999697
15.00342485
26.47644953
36.99593591
45.72815553
51.34071796
52.35927901
48.43739643
40.7661971
30.83552072
19.67006902
10.43771148
18.92008084
25.41121023
29.24919056
29.9130712
27.30034527
21.79243076
14.02900813
]';
azimuth_angle=[102.3059678
112.6554838
125.2734277
141.6402462
162.8883742
187.4492199
210.4519166
228.7179887
242.6460181
253.7456587
80.96109056
89.19354329
99.08022618
113.6313597
142.3663618
197.7927875
```

```
237.5568977
255.9546618
267.1545624
275.8472996
102.6339155
112.9897418
125.6048415
141.9284839
163.0410832
187.380392
210.2042401
228.3963008
242.3106454
253.415112
129.7484118
140.9571165
154.0680759
168.9518354
184.7639009
200.1609288
214.0333748
225.9820887
]';

    % 将角度转换为弧度

for n=1:1:38
    elevation_rad = deg2rad(elevation_angle(n));
    azimuth_rad = deg2rad(azimuth_angle(n));

    % 光线方向向量
    direction_vector = [cos(azimuth_rad) * cos(elevation_rad);
                    sin(azimuth_rad) * cos(elevation_rad);
                    sin(elevation_rad)];
    %光线参数
    point1=[0,0,-2.5];
    point22=point1+direction_vector';
    p=1;
    while point22(3)<5
        point22=point22+direction_vector';
        p=p+1;
    end

    direction=point22-point1;
    num=0;
%步长计算
numIntersectionPoints1=0;
S=0;
for x =-10:0.1:10
    for y =-10:0.1:10
        point1=[0,0,-2.5];
        point1=point1+[x,0,0]+[0,y,0];
        point2=point22+[x,0,0]+[0,y,0];
        %plot3([point1(1),point2(1)],[point1(2),point2(2)],[point1(2),point2
(2)])
        bool = 0;
        for t = 0:0.01:1  % 步长为0.01
            point = point2 - t * direction;
            % 判断点是否在半椭球内
            if point(3)>=0
                    bool1 = (((point(1)-center3(1))/a31)^2 + ((point(2)-
center3(2))/b31)^2 + ((point(3)-center3(3))/c31)^2 <= 1) & bool == 0;
                    bool2 = (((point(1)-center2(1))/a21)^2 + ((point(2)-
center2(2))/b21)^2 + ((point(3)-center2(3))/c21)^2 <= 1) & bool == 0;
                    bool3 = (((point(1)-center1(1))/a11)^2 + ((point(2)-
center1(2))/b11)^2 + ((point(3)-center1(3))/c11)^2 <= 1) & bool == 0;
                    bool4 = (((point(1)-center4(1))/a41)^2 + ((point(2)-
center4(2))/b41)^2 + ((point(3)-center4(3))/c41)^2 <= 1) & bool == 0;
                    bool5 = (((point(1)-center5(1))/a51)^2 + ((point(2)-
center5(2))/b51)^2 + ((point(3)-center5(3))/c51)^2 <= 1) & bool == 0;
                    bool6 = (((point(1)-center6(1))/a61)^2 + ((point(2)-
center6(2))/b61)^2 + ((point(3)-center6(3))/c61)^2 <= 1) & bool == 0;
                    bool7 = (((point(1)-center7(1))/a71)^2 + ((point(2)-
center7(2))/b71)^2 + ((point(3)-center7(3))/c71)^2 <= 1) & bool == 0;
                    bool8 = (((point(1)-center8(1))/a81)^2 + ((point(2)-
center8(2))/b81)^2 + ((point(3)-center8(3))/c81)^2 <= 1) & bool == 0;
                    bool9 = (((point(1)-center9(1))/a91)^2 + ((point(2)-
center9(2))/b91)^2 + ((point(3)-center9(3))/c91)^2 <= 1) & bool == 0;
                    if bool1 || bool2 || bool3 ||bool4 ||bool5||bool6 ||
bool7 ||bool8 ||bool9
                        numIntersectionPoints1 = numIntersectionPoints1 +
bool3;
                        if bool3 ==1
                            A = sqrt((point(1)/a11^2)^2 + (point(2)/b11^2)^2 +
(point(3)/c11^2)^2);

                            sintheta=(point(2)/b11^2)/A;
                            sintheta=sqrt(1-sintheta^2);
                            S=S+0.01*abs(sintheta);
                        end
                        bool = 1;
                        break
                    end

            elseif point(3)<0
                    bool1 = (((point(1)-center3(1))/a32)^2 + ((point(2)-
center3(2))/b32)^2 + ((point(3)-center3(3))/c32)^2 <= 1) & bool == 0;
                    bool2 = (((point(1)-center2(1))/a22)^2 + ((point(2)-
center2(2))/b22)^2 + ((point(3)-center2(3))/c22)^2 <= 1) & bool == 0;
                    bool3 = (((point(1)-center1(1))/a12)^2 + ((point(2)-
center1(2))/b12)^2 + ((point(3)-center1(3))/c12)^2 <= 1) & bool == 0;
                    bool4 = (((point(1)-center4(1))/a42)^2 + ((point(2)-
center4(2))/b42)^2 + ((point(3)-center4(3))/c42)^2 <= 1) & bool == 0;
                    bool5 = (((point(1)-center5(1))/a52)^2 + ((point(2)-
center5(2))/b52)^2 + ((point(3)-center5(3))/c52)^2 <= 1) & bool == 0;
                    bool6 = (((point(1)-center6(1))/a62)^2 + ((point(2)-
center6(2))/b62)^2 + ((point(3)-center6(3))/c62)^2 <= 1) & bool == 0;
                    bool7 = (((point(1)-center7(1))/a72)^2 + ((point(2)-
center7(2))/b72)^2 + ((point(3)-center7(3))/c72)^2 <= 1) & bool == 0;
                    bool8 = (((point(1)-center8(1))/a82)^2 + ((point(2)-
center8(2))/b82)^2 + ((point(3)-center8(3))/c82)^2 <= 1) & bool == 0;
                    bool9 = (((point(1)-center9(1))/a92)^2 + ((point(2)-
center9(2))/b92)^2 + ((point(3)-center9(3))/c92)^2 <= 1) & bool == 0;

                    if bool1 || bool2 || bool3 ||bool4 ||bool5||bool6 ||
bool7 ||bool8 ||bool9
                        numIntersectionPoints1 = numIntersectionPoints1 +
bool3;
                        if bool3 ==1
                            A = sqrt((point(1)/a11^2)^2 + (point(2)/b11^2)^2 +
(point(3)/c11^2)^2);
                            sintheta=(point(2)/b11^2)/A;
                            sintheta=sqrt(1-sintheta^2);
                            S=S+0.01*abs(sintheta);
                        end
                        bool = 1;
                        %scatter3(point(1), point(2), point(3), 'filled',
'MarkerFaceColor','y', 'MarkerFaceAlpha', 1);
                        break
                    end

            hold on;
            end
```

```
        end
    end
end
%disp(numIntersectionPoints1)
disp([num2str(n),'受光面积为：',num2str(S)])
end
end
axis equal;
```

附件 8 triple_tree_model_light_area.m（行向间苹果树树冠受光面积拟合模型，MATLAB 代码）

```
y1 = [6.863
11.0573
8.7201
7.7555
7.3502
7.3204
7.5572
8.2543
9.8841
10.8768
];
y2 = [10.169
7.8205
6.7271
6.2065
5.9662
5.9419
6.0549
6.4876
7.2802
8.9881
];
y3 = [6.649
11.1795
8.7818
7.8119
7.4322
7.3725
7.6315
8.2729
9.9695
10.6945
];
```

```
y4 = [
6.9969
16.0132
13.0191
11.8685
11.6315
12.3926
14.5989
13.4217
];
x = [8 9 10 11 12 13 14 15 16 17];
x2=[9 10 11 12 13 14 15 16];

y1=y1';y2=y2';y3=y3';y4=y4';

% 设置多项式阶数
n = 8;

% 进行四条曲线的曲线拟合
p1 = polyfit(x, y1, n);
p2 = polyfit(x, y2, n);
p3 = polyfit(x, y3, n);
p4 = polyfit(x2, y4, n);

% 生成一系列 x 值
x_fit = linspace(min(x), max(x), 100);

% 计算四条拟合曲线对应的 y 值
y_fit1 = polyval(p1, x_fit);
y_fit2 = polyval(p2, x_fit);
y_fit3 = polyval(p3, x_fit);
y_fit4 = polyval(p4, x_fit);

% 绘制原始数据和拟合曲线
figure;
hold on;
scatter(x, y1, 'o','filled','MarkerFaceColor','r');
scatter(x, y2, 'o', 'filled','MarkerFaceColor',[0,0.5,0.5]);
scatter(x, y3, 'o', 'filled','MarkerFaceColor','b');
scatter(x2, y4, 'o', 'filled','MarkerFaceColor',[0.5,0,0.5]);
plot(x_fit, y_fit1,'color','r','LineWidth',1.5);
plot(x_fit, y_fit2,'color',[0,0.5,0.5],'LineWidth',1.5);
plot(x_fit, y_fit3,'color','b','LineWidth',1.5);
plot(x_fit, y_fit4,'color',[0.5,0,0.5],'LineWidth',1.5);
```

```
hold off;
xlabel('时间（单位：小时）','FontSize',12,'FontName','宋体');
ylim([0 21]);
ylabel('树冠受光面积（单位：平方米）','FontSize',12,'FontName','宋体');

legend('','','','', '春分', '夏至', '秋分', '冬至
','FontSize',10,'FontName','宋体','Location','northeast');
```

附件 9 triple_tree_model_dark.m（行向间苹果树树冠阴影模型，MATLAB 代码）

```
%树冠模型参数
a11=1.5;b11=1.5;c11=2;a21=1;b21=1;c21=1.5;a31=0.5;b31=0.5;c31=1;
a12=1.5;b12=1.5;c12=1;a22=1;b22=1;c22=0.75;a32=0.5;b32=0.5;c32=0.375;
center1=[0,0,0];
center2=[-6.5,0,0];
center3=[6.5,0,0];

elevation_angle=[10.43771148
18.92008084
25.41121023
29.24919056
29.9130712
27.30034527
21.79243076
14.02900813
];
azimuth_angle=[129.7484118
140.9571165
154.0680759
168.9518354
184.7639009
200.1609288
214.0333748
225.9820887
];
elevation_angle=elevation_angle';
azimuth_angle=azimuth_angle';
for s=1:1:8
    azimuth_angle(s)=180-azimuth_angle(s);
end

for n=1:1:10
    %dark=0;
    %middle=0;
    %small=0;
    %disp(['阴影点个数为：',num2str(n)])
    % 将角度转换为弧度
    elevation_rad = deg2rad(elevation_angle(n));
    azimuth_rad = deg2rad(azimuth_angle(n));

    % 光线方向向量
    direction_vector = [cos(azimuth_rad) * cos(elevation_rad);
                        sin(azimuth_rad) * cos(elevation_rad);
                        sin(elevation_rad)];
    %光线参数
    point1=[0,0,-2.5];
    point22=point1+direction_vector';
    p=1;
    while point22(3)<5
        point22=point22+direction_vector';
        p=p+1;
    end

    direction=point22-point1;
    num=0;
    %步长计算
    for x =-40:0.2:40
        for y =-40:0.2:40
            point1=[0,0,-2.5];
            point1=point1+[x,0,0]+[0,y,0];
            point2=point22+[x,0,0]+[0,y,0];
            numIntersectionPoints1=0;
            numIntersectionPoints2=0;
            numIntersectionPoints3=0;
            direction=point2-point1;
            for t = 0:0.01:1  % 步长为0.01
                point = point1 + t * direction;
                % 判断点是否在半椭球内
                if point(3)>=0
                    if ((point(1)-center1(1))/a31)^2 + ((point(2)-
center1(2))/b31)^2 + ((point(3)-center1(3))/c31)^2 <= 1
                        numIntersectionPoints3 = numIntersectionPoints3 + 1;
                    elseif ((point(1)-center1(1))/a21)^2 + ((point(2)-
center1(2))/b21)^2 + ((point(3)-center1(3))/c21)^2 <= 1
                        numIntersectionPoints2 = numIntersectionPoints2 + 1;
                    elseif ((point(1)-center1(1))/a11)^2 + ((point(2)-
center1(2))/b11)^2 + ((point(3)-center1(3))/c11)^2 <= 1
                        numIntersectionPoints1 = numIntersectionPoints1 + 1;
                    end
                    if ((point(1)-center2(1))/a31)^2 + ((point(2)-
center2(2))/b31)^2 + ((point(3)-center2(3))/c31)^2 <= 1
                        numIntersectionPoints3 = numIntersectionPoints3 + 1;
                    elseif ((point(1)-center2(1))/a21)^2 + ((point(2)-
center2(2))/b21)^2 + ((point(3)-center2(3))/c21)^2 <= 1
                        numIntersectionPoints2 = numIntersectionPoints2 + 1;
                    elseif ((point(1)-center2(1))/a11)^2 + ((point(2)-
center2(2))/b11)^2 + ((point(3)-center2(3))/c11)^2 <= 1
                        numIntersectionPoints1 = numIntersectionPoints1 + 1;
                    end
                    if ((point(1)-center3(1))/a31)^2 + ((point(2)-
center3(2))/b31)^2 + ((point(3)-center3(3))/c31)^2 <= 1
                        numIntersectionPoints3 = numIntersectionPoints3 + 1;
                    elseif ((point(1)-center3(1))/a21)^2 + ((point(2)-
center3(2))/b21)^2 + ((point(3)-center3(3))/c21)^2 <= 1
                        numIntersectionPoints2 = numIntersectionPoints2 + 1;
                    elseif ((point(1)-center3(1))/a11)^2 + ((point(2)-
center3(2))/b11)^2 + ((point(3)-center3(3))/c11)^2 <= 1
                        numIntersectionPoints1 = numIntersectionPoints1 + 1;
                    end
                elseif point(3)<0
                    if ((point(1)-center1(1))/a32)^2 + ((point(2)-
center1(2))/b32)^2 + ((point(3)-center1(3))/c32)^2 <= 1
                        numIntersectionPoints3 = numIntersectionPoints3 + 1;
                    elseif ((point(1)-center1(1))/a22)^2 + ((point(2)-
center1(2))/b22)^2 + ((point(3)-center1(3))/c22)^2 <= 1
                        numIntersectionPoints2 = numIntersectionPoints2 + 1;
                    elseif ((point(1)-center1(1))/a12)^2 + ((point(2)-
center1(2))/b12)^2 + ((point(3)-center1(3))/c12)^2 <= 1
                        numIntersectionPoints1 = numIntersectionPoints1 + 1;
                    end
                    if ((point(1)-center2(1))/a32)^2 + ((point(2)-
center2(2))/b32)^2 + ((point(3)-center2(3))/c32)^2 <= 1
                        numIntersectionPoints3 = numIntersectionPoints3 + 1;
                    elseif ((point(1)-center2(1))/a22)^2 + ((point(2)-
center2(2))/b22)^2 + ((point(3)-center2(3))/c22)^2 <= 1
                        numIntersectionPoints2 = numIntersectionPoints2 + 1;
                    elseif ((point(1)-center2(1))/a12)^2 + ((point(2)-
center2(2))/b12)^2 + ((point(3)-center2(3))/c12)^2 <= 1
                        numIntersectionPoints1 = numIntersectionPoints1 + 1;
                    end
                    if ((point(1)-center3(1))/a32)^2 + ((point(2)-
center3(2))/b32)^2 + ((point(3)-center3(3))/c32)^2 <= 1
```

```
                        numIntersectionPoints3 = numIntersectionPoints3 + 1;
                    elseif ((point(1)-center3(1))/a22)^2 + ((point(2)-
center3(2))/b22)^2 + ((point(3)-center3(3))/c22)^2 <= 1
                        numIntersectionPoints2 = numIntersectionPoints2 + 1;
                    elseif ((point(1)-center3(1))/a12)^2 + ((point(2)-
center3(2))/b12)^2 + ((point(3)-center3(3))/c12)^2 <= 1
                        numIntersectionPoints1 = numIntersectionPoints1 + 1;
                    end
                end
            end

sum=numIntersectionPoints3*0.2+numIntersectionPoints2*0.8+numIntersectionPo
ints1*0.5;

            if sum>0
                num=num+1;
                hold on;
            end
```

附件 10 triple_tree_model_dark_area.m（行向间苹果树树冠阴影面积拟合模型，MATLAB 代码）

```
y1 = [2021
1181
886
748
678
677
723
815
1038
1558
] * 0.04;
y2 = [1109
822
669
595
556
554
570
639
754
968
] * 0.04;
y3 = [2043
1204
892
752
685
680
724
823
1042
1580
] * 0.04;
y4 = [2896
1636
1246
1107
1077
1165
1440
2183
] * 0.04;
x = [8 9 10 11 12 13 14 15 16 17];
x2=[9 10 11 12 13 14 15 16];

y1=y1';y2=y2';y3=y3';y4=y4';

% 设置多项式阶数
n = 4;

% 进行四条曲线的曲线拟合
p1 = polyfit(x, y1, n);
p2 = polyfit(x, y2, n);
p3 = polyfit(x, y3, n);
p4 = polyfit(x2, y4, n);

% 生成一系列 x 值
x_fit = linspace(min(x), max(x), 100);

% 计算四条拟合曲线对应的 y 值
y_fit1 = polyval(p1, x_fit);
y_fit2 = polyval(p2, x_fit);
y_fit3 = polyval(p3, x_fit);
y_fit4 = polyval(p4, x_fit);

% 绘制原始数据和拟合曲线
figure;
```

```
hold on;
scatter(x, y1, 'o','filled','MarkerFaceColor','r');
scatter(x, y2, 'o', 'filled','MarkerFaceColor',[0,0.5,0.5]);
scatter(x, y3, 'o', 'filled','MarkerFaceColor','b');
scatter(x2, y4, 'o', 'filled','MarkerFaceColor',[0.5,0,0.5]);
plot(x_fit, y_fit1,'color','r','LineWidth',1.5);
plot(x_fit, y_fit2,'color',[0,0.5,0.5],'LineWidth',1.5);
plot(x_fit, y_fit3,'color','b','LineWidth',1.5);
plot(x_fit, y_fit4,'color',[0.5,0,0.5],'LineWidth',1.5);
hold off;
xlabel('时间（单位：小时）','FontSize',12,'FontName','宋体');
ylim([0 250]);
ylabel('树冠阴影面积（单位：平方米）','FontSize',12,'FontName','宋体');
legend('','','','', '春分', '夏至', '秋分', '冬至
','FontSize',12,'FontName','宋体');
```

附件 11 trees_model.m（行列间苹果树树冠三维模型，MATLAB 代码）

```
clc,clear

%树冠 1 可视化模型
for v=-1:1:1
    for u=-1:1:1
        [theta,phi] = meshgrid(0:10:360,0:10:90);
        X = 1.5*cos(phi*pi/180).*cos(theta*pi/180)+v*5;
        Y = 1.5*cos(phi*pi/180).*sin(theta*pi/180)+u*5;
        Z = 2*sin(phi*pi/180);
        ss1=surf(X, Y, Z);
        ss1.FaceAlpha=0.4;
        hold on;
        [theta,phi2] = meshgrid(0:10:360,-90:10:0);
        X2 = 1.5*cos(phi2*pi/180).*cos(theta.*pi/180)+v*5;
        Y2 = 1.5*cos(phi2*pi/180).*sin(theta*pi/180)+u*5;
        Z2 = 1*sin(phi2*pi/180);
        ss2=surf(X2, Y2, Z2);
        ss2.FaceAlpha=0.4;
        hold on;
        [theta,phi] = meshgrid(0:10:360,0:10:90);
        X3 = 1*cos(phi*pi/180).*cos(theta*pi/180)+v*5;
        Y3 = 1*cos(phi*pi/180).*sin(theta*pi/180)+u*5;
        Z3 = 1.5*sin(phi*pi/180);
        ss3=surf(X3, Y3, Z3);
        ss3.FaceAlpha=0.4;
        hold on;
        X4 = 1*cos(phi2*pi/180).*cos(theta.*pi/180)+v*5;
        Y4 = 1*cos(phi2*pi/180).*sin(theta*pi/180)+u*5;
        Z4 = 0.75*sin(phi2*pi/180);
        ss4=surf(X4, Y4, Z4);
        ss4.FaceAlpha=0.4;
        hold on;
        X5 = 0.5*cos(phi*pi/180).*cos(theta*pi/180)+v*5;
        Y5 = 0.5*cos(phi*pi/180).*sin(theta*pi/180)+u*5;
        Z5 = 1*sin(phi*pi/180);
        ss5=surf(X5, Y5, Z5);
        ss5.FaceAlpha=0.4;
        hold on;
        X6 = 0.5*cos(phi2*pi/180).*cos(theta.*pi/180)+v*5;
        Y6 = 0.5*cos(phi2*pi/180).*sin(theta*pi/180)+u*5;
        Z6 = 0.375*sin(phi2*pi/180);
        ss6=surf(X6, Y6, Z6);
        ss6.FaceAlpha=0.4;
        hold on;
    end
end

xlabel('南北方向（单位：m）','FontSize',15,'FontName','宋体')
ylabel('东西方向（单位：m）','FontSize',15,'FontName','宋体')
axis equal;
```

附件 12 trees_model_light.m（行列间苹果树树冠受光模型，MATLAB 代码）

```
clc,clear
for d = 3.5:1.5:6.5

%树冠模型参数
a11=1.5;b11=1.5;c11=2;
a12=1.5;b12=1.5;c12=1;
center1=[0,0,0];

% 树冠模型参数
a31 = 1.5; b31 = 1.5; c31 = 2;
a32 = 1.5; b32 = 1.5; c32 = 1;
center3 = [0, -d, 0];  % 将中心点在 x 轴上移动 5 个单位

% 树冠模型参数
a21 = 1.5; b21 = 1.5; c21 = 2;
a22 = 1.5; b22 = 1.5; c22 = 1;
center2 = [0, d, 0];  % 将中心点在 x 轴上移动 5 个单位

% 树冠模型参数
```

```
a41 = 1.5; b41 = 1.5; c41 = 2;
a42 = 1.5; b42 = 1.5; c42 = 1;
center4 = [d, 0, 0];  % 将中心点在 x 轴上移动 5 个单位

% 树冠模型参数
a51 = 1.5; b51 = 1.5; c51 = 2;
a52 = 1.5; b52 = 1.5; c52 = 1;
center5 = [-d, 0,0];  % 将中心点在 x 轴上移动 5 个单位

% 树冠模型参数
a61 = 1.5; b61 = 1.5; c61 = 2;
a62 = 1.5; b62 = 1.5; c62 = 1;
center6 = [-d, -d, 0];  % 将中心点在 x 轴上移动 5 个单位

% 树冠模型参数
a71 = 1.5; b71 = 1.5; c71 = 2;
a72 = 1.5; b72 = 1.5; c72 = 1;
center7 = [d, -d, 0];  % 将中心点在 x 轴上移动 5 个单位

a81 = 1.5; b81 = 1.5; c81 = 2;
a82 = 1.5; b82 = 1.5; c82 = 1;
center8 = [-d, d, 0];  % 将中心点在 x 轴上移动 5 个单位

% 树冠模型参数
a91 = 1.5; b91 = 1.5; c91 = 2;
a92 = 1.5; b92 = 1.5; c92 = 1;
center9 = [d, d, 0];  % 将中心点在 x 轴上移动 5 个单位



%两个角度
elevation_angle=[15.25329928
26.74728536
37.30069887
46.07805656
51.73268607
52.76041559
48.80598458
41.08804954
31.11833248
19.92684826
28.8681958
40.86994525
52.88480367
```

```
64.46002994
74.10822164
76.37468044
68.72145662
57.6097592
45.69154488
33.64999697
15.00342485
26.47644953
36.99593591
45.72815553
51.34071796
52.35927901
48.43739643
40.7661971
30.83552072
19.67006902
10.43771148
18.92008084
25.41121023
29.24919056
29.9130712
27.30034527
21.79243076
14.02900813
]';
azimuth_angle=[102.3059678
112.6554838
125.2734277
141.6402462
162.8883742
187.4492199
210.4519166
228.7179887
242.6460181
253.7456587
80.96109056
89.19354329
99.08022618
113.6313597
142.3663618
197.7927875
237.5568977
255.9546618
```

```
267.1545624
275.8472996
102.6339155
112.9897418
125.6048415
141.9284839
163.0410832
187.380392
210.2042401
228.3963008
242.3106454
253.415112
129.7484118
140.9571165
154.0680759
168.9518354
184.7639009
200.1609288
214.0333748
225.9820887
]';

    % 将角度转换为弧度

for n=1:1:38
    elevation_rad = deg2rad(elevation_angle(n));
    azimuth_rad = deg2rad(azimuth_angle(n));

    % 光线方向向量
    direction_vector = [cos(azimuth_rad) * cos(elevation_rad);
                   sin(azimuth_rad) * cos(elevation_rad);
                   sin(elevation_rad)];
    %光线参数
    point1=[0,0,-2.5];
    point22=point1+direction_vector';
    p=1;
    while point22(3)<5
        point22=point22+direction_vector';
        p=p+1;
    end

    direction=point22-point1;
    num=0;
%步长计算
```

```
numIntersectionPoints1=0;
S=0;
for x =-10:0.1:10
    for y =-10:0.1:10
        point1=[0,0,-2.5];
        point1=point1+[x,0,0]+[0,y,0];
        point2=point22+[x,0,0]+[0,y,0];
        %plot3([point1(1),point2(1)],[point1(2),point2(2)],[point1(2),point2
(2)])
        bool = 0;
        for t = 0:0.01:1  % 步长为0.01
            point = point2 - t * direction;
            % 判断点是否在半椭球内
            if point(3)>=0
                    bool1 = (((point(1)-center3(1))/a31)^2 + ((point(2)-
center3(2))/b31)^2 + ((point(3)-center3(3))/c31)^2 <= 1) & bool == 0;
                    bool2 = (((point(1)-center2(1))/a21)^2 + ((point(2)-
center2(2))/b21)^2 + ((point(3)-center2(3))/c21)^2 <= 1) & bool == 0;
                    bool3 = (((point(1)-center1(1))/a11)^2 + ((point(2)-
center1(2))/b11)^2 + ((point(3)-center1(3))/c11)^2 <= 1) & bool == 0;
                    bool4 = (((point(1)-center4(1))/a41)^2 + ((point(2)-
center4(2))/b41)^2 + ((point(3)-center4(3))/c41)^2 <= 1) & bool == 0;
                    bool5 = (((point(1)-center5(1))/a51)^2 + ((point(2)-
center5(2))/b51)^2 + ((point(3)-center5(3))/c51)^2 <= 1) & bool == 0;
                    bool6 = (((point(1)-center6(1))/a61)^2 + ((point(2)-
center6(2))/b61)^2 + ((point(3)-center6(3))/c61)^2 <= 1) & bool == 0;
                    bool7 = (((point(1)-center7(1))/a71)^2 + ((point(2)-
center7(2))/b71)^2 + ((point(3)-center7(3))/c71)^2 <= 1) & bool == 0;
                    bool8 = (((point(1)-center8(1))/a81)^2 + ((point(2)-
center8(2))/b81)^2 + ((point(3)-center8(3))/c81)^2 <= 1) & bool == 0;
                    bool9 = (((point(1)-center9(1))/a91)^2 + ((point(2)-
center9(2))/b91)^2 + ((point(3)-center9(3))/c91)^2 <= 1) & bool == 0;
                    if bool1 || bool2 || bool3 ||bool4 ||bool5||bool6 ||
bool7 ||bool8 ||bool9
                        numIntersectionPoints1 = numIntersectionPoints1 +
bool3;
                        if bool3 ==1
                            A = sqrt((point(1)/a11^2)^2 + (point(2)/b11^2)^2 +
(point(3)/c11^2)^2);
                            sintheta=(point(2)/b11^2)/A;
                            sintheta=sqrt(1-sintheta^2);
                            S=S+0.01*abs(sintheta);
                        end
                        bool = 1;
                    break
                end

            elseif point(3)<0
                bool1 = (((point(1)-center3(1))/a32)^2 + ((point(2)-
center3(2))/b32)^2 + ((point(3)-center3(3))/c32)^2 <= 1) & bool == 0;
                bool2 = (((point(1)-center2(1))/a22)^2 + ((point(2)-
center2(2))/b22)^2 + ((point(3)-center2(3))/c22)^2 <= 1) & bool == 0;
                bool3 = (((point(1)-center1(1))/a12)^2 + ((point(2)-
center1(2))/b12)^2 + ((point(3)-center1(3))/c12)^2 <= 1) & bool == 0;
                bool4 = (((point(1)-center4(1))/a42)^2 + ((point(2)-
center4(2))/b42)^2 + ((point(3)-center4(3))/c42)^2 <= 1) & bool == 0;
                bool5 = (((point(1)-center5(1))/a52)^2 + ((point(2)-
center5(2))/b52)^2 + ((point(3)-center5(3))/c52)^2 <= 1) & bool == 0;
                bool6 = (((point(1)-center6(1))/a62)^2 + ((point(2)-
center6(2))/b62)^2 + ((point(3)-center6(3))/c62)^2 <= 1) & bool == 0;
                bool7 = (((point(1)-center7(1))/a72)^2 + ((point(2)-
center7(2))/b72)^2 + ((point(3)-center7(3))/c72)^2 <= 1) & bool == 0;
                bool8 = (((point(1)-center8(1))/a82)^2 + ((point(2)-
center8(2))/b82)^2 + ((point(3)-center8(3))/c82)^2 <= 1) & bool == 0;
                bool9 = (((point(1)-center9(1))/a92)^2 + ((point(2)-
center9(2))/b92)^2 + ((point(3)-center9(3))/c92)^2 <= 1) & bool == 0;

                if bool1 || bool2 || bool3 ||bool4 ||bool5||bool6 ||
bool7 ||bool8 ||bool9
                    numIntersectionPoints1 = numIntersectionPoints1 +
bool3;
                    if bool3 ==1
                        A = sqrt((point(1)/a11^2)^2 + (point(2)/b11^2)^2 +
(point(3)/c11^2)^2);
                        sintheta=(point(2)/b11^2)/A;
                        sintheta=sqrt(1-sintheta^2);
                        S=S+0.01*abs(sintheta);
                    end
                    bool = 1;
                    %scatter3(point(1), point(2), point(3), 'filled',
'MarkerFaceColor','y', 'MarkerFaceAlpha', 1);
                    break
                end

            hold on;
            end
        end
    end
```

```
end
%disp(numIntersectionPoints1)
disp([num2str(n),'受光面积为: ',num2str(S)])
end
end
axis equal;
```

附件 13 trees_model_light_area.m（行列间苹果树树冠受光面积拟合模型，MATLAB 代码）

```
y1 = [
0.0068002
10.3097
10.622
9.438
9.2057
10.2259
12.7375
7.9064
] ;
y2 = [
0.96758
14.7647
13.0191
11.8685
11.6315
12.3926
14.5989
13.2969
];
y3 = [1.5723
16.0132
13.0191
11.8685
11.6315
12.3926
14.5989
13.4217
];
%y4 = [
%] * 0.04;
x = [8 9 10 11 12 13 14 15 16 17];
x2=[9 10 11 12 13 14 15 16];
```

```
y1=y1';y2=y2';y3=y3';

%y4=y4';

% 设置多项式阶数
n = 6;

% 进行四条曲线的曲线拟合
p1 = polyfit(x2, y1, n);
p2 = polyfit(x2, y2, n);
p3 = polyfit(x2, y3, n);
%p4 = polyfit(x2, y4, n);

% 生成一系列 x 值
x_fit = linspace(min(x), max(x), 100);

% 计算四条拟合曲线对应的 y 值
y_fit1 = polyval(p1, x_fit);
y_fit2 = polyval(p2, x_fit);
y_fit3 = polyval(p3, x_fit);
%y_fit4 = polyval(p4, x_fit);

% 绘制原始数据和拟合曲线
figure;
hold on;
scatter(x2, y1, 'o','filled','MarkerFaceColor',[0.5,0,0.5]);
scatter(x2, y2, 'o', 'filled','MarkerFaceColor',[0,0.5,0.5]);
scatter(x2, y3, 'o', 'filled','MarkerFaceColor','b');
%scatter(x2, y4, 'o', 'filled','MarkerFaceColor',[0.5,0,0.5]);
plot(x_fit, y_fit1,'color',[0.5,0,0.5],'LineWidth',1.5);
plot(x_fit, y_fit2,'color',[0,0.5,0.5],'LineWidth',1.5);
plot(x_fit, y_fit3,'color','b','LineWidth',1.5);
%plot(x_fit, y_fit4,'color',[0.5,0,0.5],'LineWidth',1.5);
hold off;
xlabel('时间（单位：小时）','FontSize',12,'FontName','宋体');
ylim([-3 16]);
xlim([8 17])
ylabel('树冠受光面积（单位：平方米）','FontSize',12,'FontName','宋体');
legend('','','', '3.5m 间距', '5m 间距', '6.5m 间距
','FontSize',10,'FontName','宋体','Location','southeast');
```

附件 14 trees_model_dark.m（行列间苹果树树冠阴影模型，MATLAB 代码）

```
clc,clear
%树冠 1 可视化模型
```

```
for v=-1:1:1
    for u=-1:1:1
        [theta,phi] = meshgrid(0:10:360,0:10:90);
        X = 1.5*cos(phi*pi/180).*cos(theta*pi/180)+v*5;
        Y = 1.5*cos(phi*pi/180).*sin(theta*pi/180)+u*5;
        Z = 2*sin(phi*pi/180);
        ss1=surf(X, Y, Z);
        ss1.FaceAlpha=0.4;
        hold on;
        [theta,phi2] = meshgrid(0:10:360,-90:10:0);
        X2 = 1.5*cos(phi2*pi/180).*cos(theta.*pi/180)+v*5;
        Y2 = 1.5*cos(phi2*pi/180).*sin(theta*pi/180)+u*5;
        Z2 = 1*sin(phi2*pi/180);
        ss2=surf(X2, Y2, Z2);
        ss2.FaceAlpha=0.4;
        hold on;
        [theta,phi] = meshgrid(0:10:360,0:10:90);
        X3 = 1*cos(phi*pi/180).*cos(theta*pi/180)+v*5;
        Y3 = 1*cos(phi*pi/180).*sin(theta*pi/180)+u*5;
        Z3 = 1.5*sin(phi*pi/180);
        ss3=surf(X3, Y3, Z3);
        ss3.FaceAlpha=0.4;
        hold on;
        X4 = 1*cos(phi2*pi/180).*cos(theta.*pi/180)+v*5;
        Y4 = 1*cos(phi2*pi/180).*sin(theta*pi/180)+u*5;
        Z4 = 0.75*sin(phi2*pi/180);
        ss4=surf(X4, Y4, Z4);
        ss4.FaceAlpha=0.4;
        hold on;
        X5 = 0.5*cos(phi*pi/180).*cos(theta*pi/180)+v*5;
        Y5 = 0.5*cos(phi*pi/180).*sin(theta*pi/180)+u*5;
        Z5 = 1*sin(phi*pi/180);
        ss5=surf(X5, Y5, Z5);
        ss5.FaceAlpha=0.4;
        hold on;
        X6 = 0.5*cos(phi2*pi/180).*cos(theta.*pi/180)+v*5;
        Y6 = 0.5*cos(phi2*pi/180).*sin(theta*pi/180)+u*5;
        Z6 = 0.375*sin(phi2*pi/180);
        ss6=surf(X6, Y6, Z6);
        ss6.FaceAlpha=0.4;
        hold on;
    end
end
```

```
%树冠模型参数
a11=1.5;b11=1.5;c11=2;a21=1;b21=1;c21=1.5;a31=0.5;b31=0.5;c31=1;
a12=1.5;b12=1.5;c12=1;a22=1;b22=1;c22=0.75;a32=0.5;b32=0.5;c32=0.375;

center2=[6.5,6.5,0];center3=[6.5,0,0];center4=[6.5,-6.5,0];
center5=[0,6.5,0];center1=[0,0,0];center6=[0,-6.5,0];
center7=[-6.5,6.5,0];center8=[-6.5,0,0];center9=[-6.5,-6.5,0];


elevation_angle=[10.43771148
18.92008084
25.41121023
29.24919056
29.9130712
27.30034527
21.79243076
14.02900813
];
azimuth_angle=[129.7484118
140.9571165
154.0680759
168.9518354
184.7639009
200.1609288
214.0333748
225.9820887
];
elevation_angle=elevation_angle';
azimuth_angle=azimuth_angle';
for s=1:1:8
    azimuth_angle(s)=180-azimuth_angle(s);
end

for n=1:1:8
    %dark=0;
    %middle=0;
    %small=0;
    %disp(['阴影点个数为: ',num2str(n)])
    % 将角度转换为弧度
    elevation_rad = deg2rad(elevation_angle(n));
    azimuth_rad = deg2rad(azimuth_angle(n));

    % 光线方向向量
    direction_vector = [cos(azimuth_rad) * cos(elevation_rad);
                        sin(azimuth_rad) * cos(elevation_rad);
                        sin(elevation_rad)];
    %光线参数
    point1=[0,0,-2.5];
    point22=point1+direction_vector';
    p=1;
    while point22(3)<5
        point22=point22+direction_vector';
        p=p+1;
    end

    direction=point22-point1;
    num=0;
    %步长计算
    for x =-60:0.2:60
        for y =-60:0.2:60
            point1=[0,0,-2.5];
            point1=point1+[x,0,0]+[0,y,0];
            point2=point22+[x,0,0]+[0,y,0];
            %plot3([point1(1),point2(1)],[point1(2),point2(2)],[point1(3),po
int2(3)]);
            numIntersectionPoints1=0;
            numIntersectionPoints2=0;
            numIntersectionPoints3=0;
            direction=point2-point1;
            for t = 0:0.01:1  % 步长为 0.01
                point = point1 + t * direction;
                % 判断点是否在半椭球内
                if point(3)>=0
                    if ((point(1)-center1(1))/a31)^2 + ((point(2)-
center1(2))/b31)^2 + ((point(3)-center1(3))/c31)^2 <= 1
                        numIntersectionPoints3 = numIntersectionPoints3 + 1;
                    elseif ((point(1)-center1(1))/a21)^2 + ((point(2)-
center1(2))/b21)^2 + ((point(3)-center1(3))/c21)^2 <= 1
                        numIntersectionPoints2 = numIntersectionPoints2 + 1;
                    elseif ((point(1)-center1(1))/a11)^2 + ((point(2)-
center1(2))/b11)^2 + ((point(3)-center1(3))/c11)^2 <= 1
                        numIntersectionPoints1 = numIntersectionPoints1 + 1;
                    end
                    if ((point(1)-center2(1))/a31)^2 + ((point(2)-
center2(2))/b31)^2 + ((point(3)-center2(3))/c31)^2 <= 1
                        numIntersectionPoints3 = numIntersectionPoints3 + 1;
                    elseif ((point(1)-center2(1))/a21)^2 + ((point(2)-
center2(2))/b21)^2 + ((point(3)-center2(3))/c21)^2 <= 1
                        numIntersectionPoints2 = numIntersectionPoints2 + 1;
                    elseif ((point(1)-center2(1))/a11)^2 + ((point(2)-
center2(2))/b11)^2 + ((point(3)-center2(3))/c11)^2 <= 1
                        numIntersectionPoints1 = numIntersectionPoints1 + 1;
                    end
                    if ((point(1)-center3(1))/a31)^2 + ((point(2)-
center3(2))/b31)^2 + ((point(3)-center3(3))/c31)^2 <= 1
                        numIntersectionPoints3 = numIntersectionPoints3 + 1;
                    elseif ((point(1)-center3(1))/a21)^2 + ((point(2)-
center3(2))/b21)^2 + ((point(3)-center3(3))/c21)^2 <= 1
                        numIntersectionPoints2 = numIntersectionPoints2 + 1;
                    elseif ((point(1)-center3(1))/a11)^2 + ((point(2)-
center3(2))/b11)^2 + ((point(3)-center3(3))/c11)^2 <= 1
                        numIntersectionPoints1 = numIntersectionPoints1 + 1;
                    end
                    if ((point(1)-center4(1))/a31)^2 + ((point(2)-
center4(2))/b31)^2 + ((point(3)-center4(3))/c31)^2 <= 1
                        numIntersectionPoints3 = numIntersectionPoints3 + 1;
                    elseif ((point(1)-center4(1))/a21)^2 + ((point(2)-
center4(2))/b21)^2 + ((point(3)-center4(3))/c21)^2 <= 1
                        numIntersectionPoints2 = numIntersectionPoints2 + 1;
                    elseif ((point(1)-center4(1))/a11)^2 + ((point(2)-
center4(2))/b11)^2 + ((point(3)-center4(3))/c11)^2 <= 1
                        numIntersectionPoints1 = numIntersectionPoints1 + 1;
                    end
                    if ((point(1)-center5(1))/a31)^2 + ((point(2)-
center5(2))/b31)^2 + ((point(3)-center5(3))/c31)^2 <= 1
                        numIntersectionPoints3 = numIntersectionPoints3 + 1;
                    elseif ((point(1)-center5(1))/a21)^2 + ((point(2)-
center5(2))/b21)^2 + ((point(3)-center5(3))/c21)^2 <= 1
                        numIntersectionPoints2 = numIntersectionPoints2 + 1;
                    elseif ((point(1)-center5(1))/a11)^2 + ((point(2)-
center5(2))/b11)^2 + ((point(3)-center5(3))/c11)^2 <= 1
                        numIntersectionPoints1 = numIntersectionPoints1 + 1;
                    end
                    if ((point(1)-center6(1))/a31)^2 + ((point(2)-
center6(2))/b31)^2 + ((point(3)-center6(3))/c31)^2 <= 1
                        numIntersectionPoints3 = numIntersectionPoints3 + 1;
                    elseif ((point(1)-center6(1))/a21)^2 + ((point(2)-
center6(2))/b21)^2 + ((point(3)-center6(3))/c21)^2 <= 1
                        numIntersectionPoints2 = numIntersectionPoints2 + 1;
                    elseif ((point(1)-center6(1))/a11)^2 + ((point(2)-
center6(2))/b11)^2 + ((point(3)-center6(3))/c11)^2 <= 1
                        numIntersectionPoints1 = numIntersectionPoints1 + 1;
                    end
                    if ((point(1)-center7(1))/a31)^2 + ((point(2)-
center7(2))/b31)^2 + ((point(3)-center7(3))/c31)^2 <= 1
                        numIntersectionPoints3 = numIntersectionPoints3 + 1;
                    elseif ((point(1)-center7(1))/a21)^2 + ((point(2)-
center7(2))/b21)^2 + ((point(3)-center7(3))/c21)^2 <= 1
                        numIntersectionPoints2 = numIntersectionPoints2 + 1;
                    elseif ((point(1)-center7(1))/a11)^2 + ((point(2)-
center7(2))/b11)^2 + ((point(3)-center7(3))/c11)^2 <= 1
                        numIntersectionPoints1 = numIntersectionPoints1 + 1;
                    end
                    if ((point(1)-center8(1))/a31)^2 + ((point(2)-
center8(2))/b31)^2 + ((point(3)-center8(3))/c31)^2 <= 1
                        numIntersectionPoints3 = numIntersectionPoints3 + 1;
                    elseif ((point(1)-center8(1))/a21)^2 + ((point(2)-
center8(2))/b21)^2 + ((point(3)-center8(3))/c21)^2 <= 1
                        numIntersectionPoints2 = numIntersectionPoints2 + 1;
                    elseif ((point(1)-center8(1))/a11)^2 + ((point(2)-
center8(2))/b11)^2 + ((point(3)-center8(3))/c11)^2 <= 1
                        numIntersectionPoints1 = numIntersectionPoints1 + 1;
                    end
                    if ((point(1)-center9(1))/a31)^2 + ((point(2)-
center9(2))/b31)^2 + ((point(3)-center9(3))/c31)^2 <= 1
                        numIntersectionPoints3 = numIntersectionPoints3 + 1;
                    elseif ((point(1)-center9(1))/a21)^2 + ((point(2)-
center9(2))/b21)^2 + ((point(3)-center9(3))/c21)^2 <= 1
                        numIntersectionPoints2 = numIntersectionPoints2 + 1;
                    elseif ((point(1)-center9(1))/a11)^2 + ((point(2)-
center9(2))/b11)^2 + ((point(3)-center9(3))/c11)^2 <= 1
                        numIntersectionPoints1 = numIntersectionPoints1 + 1;
                    end
                elseif point(3)<0
                    if ((point(1)-center1(1))/a32)^2 + ((point(2)-
center1(2))/b32)^2 + ((point(3)-center1(3))/c32)^2 <= 1
                        numIntersectionPoints3 = numIntersectionPoints3 + 1;
                    elseif ((point(1)-center1(1))/a22)^2 + ((point(2)-
center1(2))/b22)^2 + ((point(3)-center1(3))/c22)^2 <= 1
                        numIntersectionPoints2 = numIntersectionPoints2 + 1;
                    elseif ((point(1)-center1(1))/a12)^2 + ((point(2)-
center1(2))/b12)^2 + ((point(3)-center1(3))/c12)^2 <= 1
                        numIntersectionPoints1 = numIntersectionPoints1 + 1;
                    end
                    if ((point(1)-center2(1))/a32)^2 + ((point(2)-
center2(2))/b32)^2 + ((point(3)-center2(3))/c32)^2 <= 1
                        numIntersectionPoints3 = numIntersectionPoints3 + 1;
                    elseif ((point(1)-center2(1))/a22)^2 + ((point(2)-
center2(2))/b22)^2 + ((point(3)-center2(3))/c22)^2 <= 1
                        numIntersectionPoints2 = numIntersectionPoints2 + 1;
                    elseif ((point(1)-center2(1))/a12)^2 + ((point(2)-
center2(2))/b12)^2 + ((point(3)-center2(3))/c12)^2 <= 1
                        numIntersectionPoints1 = numIntersectionPoints1 + 1;
                    end
                    if ((point(1)-center3(1))/a32)^2 + ((point(2)-
center3(2))/b32)^2 + ((point(3)-center3(3))/c32)^2 <= 1
                        numIntersectionPoints3 = numIntersectionPoints3 + 1;
                    elseif ((point(1)-center3(1))/a22)^2 + ((point(2)-
center3(2))/b22)^2 + ((point(3)-center3(3))/c22)^2 <= 1
                        numIntersectionPoints2 = numIntersectionPoints2 + 1;
                    elseif ((point(1)-center3(1))/a12)^2 + ((point(2)-
center3(2))/b12)^2 + ((point(3)-center3(3))/c12)^2 <= 1
                        numIntersectionPoints1 = numIntersectionPoints1 + 1;
                    end
                    if ((point(1)-center4(1))/a31)^2 + ((point(2)-
center4(2))/b31)^2 + ((point(3)-center4(3))/c31)^2 <= 1
                        numIntersectionPoints3 = numIntersectionPoints3 + 1;
                    elseif ((point(1)-center4(1))/a21)^2 + ((point(2)-
center4(2))/b21)^2 + ((point(3)-center4(3))/c21)^2 <= 1
                        numIntersectionPoints2 = numIntersectionPoints2 + 1;
                    elseif ((point(1)-center4(1))/a11)^2 + ((point(2)-
center4(2))/b11)^2 + ((point(3)-center4(3))/c11)^2 <= 1
                        numIntersectionPoints1 = numIntersectionPoints1 + 1;
                    end
                    if ((point(1)-center5(1))/a31)^2 + ((point(2)-
center5(2))/b31)^2 + ((point(3)-center5(3))/c31)^2 <= 1
                        numIntersectionPoints3 = numIntersectionPoints3 + 1;
                    elseif ((point(1)-center5(1))/a21)^2 + ((point(2)-
center5(2))/b21)^2 + ((point(3)-center5(3))/c21)^2 <= 1
                        numIntersectionPoints2 = numIntersectionPoints2 + 1;
                    elseif ((point(1)-center5(1))/a11)^2 + ((point(2)-
center5(2))/b11)^2 + ((point(3)-center5(3))/c11)^2 <= 1
                        numIntersectionPoints1 = numIntersectionPoints1 + 1;
                    end
                    if ((point(1)-center6(1))/a31)^2 + ((point(2)-
center6(2))/b31)^2 + ((point(3)-center6(3))/c31)^2 <= 1
                        numIntersectionPoints3 = numIntersectionPoints3 + 1;
                    elseif ((point(1)-center6(1))/a21)^2 + ((point(2)-
center6(2))/b21)^2 + ((point(3)-center6(3))/c21)^2 <= 1
                        numIntersectionPoints2 = numIntersectionPoints2 + 1;
                    elseif ((point(1)-center6(1))/a11)^2 + ((point(2)-
center6(2))/b11)^2 + ((point(3)-center6(3))/c11)^2 <= 1
                        numIntersectionPoints1 = numIntersectionPoints1 + 1;
                    end
                    if ((point(1)-center7(1))/a31)^2 + ((point(2)-
center7(2))/b31)^2 + ((point(3)-center7(3))/c31)^2 <= 1
                        numIntersectionPoints3 = numIntersectionPoints3 + 1;
                    elseif ((point(1)-center7(1))/a21)^2 + ((point(2)-
center7(2))/b21)^2 + ((point(3)-center7(3))/c21)^2 <= 1
                        numIntersectionPoints2 = numIntersectionPoints2 + 1;
                    elseif ((point(1)-center7(1))/a11)^2 + ((point(2)-
center7(2))/b11)^2 + ((point(3)-center7(3))/c11)^2 <= 1
                        numIntersectionPoints1 = numIntersectionPoints1 + 1;
                    end
                    if ((point(1)-center8(1))/a31)^2 + ((point(2)-
center8(2))/b31)^2 + ((point(3)-center8(3))/c31)^2 <= 1
                        numIntersectionPoints3 = numIntersectionPoints3 + 1;
                    elseif ((point(1)-center8(1))/a21)^2 + ((point(2)-
center8(2))/b21)^2 + ((point(3)-center8(3))/c21)^2 <= 1
                        numIntersectionPoints2 = numIntersectionPoints2 + 1;
                    elseif ((point(1)-center8(1))/a11)^2 + ((point(2)-
center8(2))/b11)^2 + ((point(3)-center8(3))/c11)^2 <= 1
                        numIntersectionPoints1 = numIntersectionPoints1 + 1;
                    end
                    if ((point(1)-center9(1))/a31)^2 + ((point(2)-
center9(2))/b31)^2 + ((point(3)-center9(3))/c31)^2 <= 1
                        numIntersectionPoints3 = numIntersectionPoints3 + 1;
                    elseif ((point(1)-center9(1))/a21)^2 + ((point(2)-
center9(2))/b21)^2 + ((point(3)-center9(3))/c21)^2 <= 1
                        numIntersectionPoints2 = numIntersectionPoints2 + 1;
                    elseif ((point(1)-center9(1))/a11)^2 + ((point(2)-
center9(2))/b11)^2 + ((point(3)-center9(3))/c11)^2 <= 1
                        numIntersectionPoints1 = numIntersectionPoints1 + 1;
                    end
                end
            end
            %disp(['直线与半椭球 1 相交的点个数为：',
num2str(numIntersectionPoints1)]);
            %disp(['直线与半椭球 2 相交的点个数为：',
num2str(numIntersectionPoints2)]);
```

```
            %disp(['直线与半椭球 3 相交的点个数为: ', 
num2str(numIntersectionPoints3)]);
            %disp(['坐标为: ',point1])

sum=numIntersectionPoints3*0.2+numIntersectionPoints2*0.8+numIntersectionPo
ints1*0.5;

            if sum>0
                num=num+1;
                %if sum>12
                    %dark=dark+1;
                 %   scatter3([point1(1)], [point1(2)],[point1(3)],'filled',
'MarkerFaceColor','k','MarkerFaceAlpha', 0.8);
                %elseif sum>6
                 %   %middle=middle+1;
                  %  scatter3([point1(1)], [point1(2)],[point1(3)],'filled',
'MarkerFaceColor','k','MarkerFaceAlpha', 0.5);
                %elseif sum>0
                    %small=small+1;
                 %   scatter3([point1(1)], [point1(2)],[point1(3)],'filled',
'MarkerFaceColor','k','MarkerFaceAlpha', 0.2);
                %end
                %{
                % 平面方程
                % 假设平面方程为 Ax + By + Cz + D = 0
                A = 0;
                B = 0;
                C = 1;
                D = 2.5;

                syms m
                point = point1 + m * direction;
                eqn = A * point(1) + B * point(2) + C * point(3) + D == 0;
                t_sol = solve(eqn, m);
                %}
            % 计算交点坐标
        %{
                intersectionPoints = point1 + double(t_sol) * direction;
                if sum>=12
                    scatter3([intersectionPoints(1)], 
[intersectionPoints(2)],[intersectionPoints(3)],'filled', 
'MarkerFaceColor','k','MarkerFaceAlpha', 0.8);
                elseif sum>=6
```

```
                    scatter3([intersectionPoints(1)],
[intersectionPoints(2)],[intersectionPoints(3)],'filled',
'MarkerFaceColor','k','MarkerFaceAlpha', 0.5);
                elseif sum>0
                    scatter3([intersectionPoints(1)],
[intersectionPoints(2)],[intersectionPoints(3)],'filled',
'MarkerFaceColor','k','MarkerFaceAlpha', 0.2);
                end
                %disp(['阴影程度为: ',num2str(sum)])
        %}
                hold on;
            end

        end
    end
    %S=dark*1+middle*0.7
    disp([num2str(num)])
end

xlabel('南北方向（单位：m）','FontSize',15,'FontName','宋体')
ylabel('东西方向（单位：m）','FontSize',15,'FontName','宋体')
axis equal;
```

附件 15 trees_model_dark_area.m（行列间苹果树树冠阴影面积拟合模型，MATLAB 代码）

```
y1 = [6659
4240
3345
2826
2681
3137
3799
5314
] * 0.04;
y2 = [8200
5293
4304
3390
3244
3850
4918
6361
] * 0.04;
y3 = [
8861
5704
4389
3773
3616
4090
5109
6893
] * 0.04;
%y4 = [
%] * 0.04;
x = [8 9 10 11 12 13 14 15 16 17];
x2=[9 10 11 12 13 14 15 16];

y1=y1';y2=y2';y3=y3';
%y4=y4';

% 设置多项式阶数
n = 4;

% 进行四条曲线的曲线拟合
p1 = polyfit(x2, y1, n);
p2 = polyfit(x2, y2, n);
p3 = polyfit(x2, y3, n);
%p4 = polyfit(x2, y4, n);

% 生成一系列 x 值
x_fit = linspace(min(x2), max(x2), 100);

% 计算四条拟合曲线对应的 y 值
y_fit1 = polyval(p1, x_fit);
y_fit2 = polyval(p2, x_fit);
y_fit3 = polyval(p3, x_fit);
%y_fit4 = polyval(p4, x_fit);

% 绘制原始数据和拟合曲线
figure;
hold on;
scatter(x2, y1, 'o','filled','MarkerFaceColor',[0.5,0,0.5]);
scatter(x2, y2, 'o', 'filled','MarkerFaceColor',[0,0.5,0.5]);
scatter(x2, y3, 'o', 'filled','MarkerFaceColor','b');
%scatter(x2, y4, 'o', 'filled','MarkerFaceColor',[0.5,0,0.5]);
plot(x_fit, y_fit1,'color',[0.5,0,0.5],'LineWidth',1.5);
plot(x_fit, y_fit2,'color',[0,0.5,0.5],'LineWidth',1.5);
plot(x_fit, y_fit3,'color','b','LineWidth',1.5);
%plot(x_fit, y_fit4,'color',[0.5,0,0.5],'LineWidth',1.5);
hold off;
xlabel('时间（单位：小时）','FontSize',12,'FontName','宋体');
ylim([0 360]);
ylabel('树冠阴影面积（单位：平方米）','FontSize',12,'FontName','宋体');
legend('','','', '3.5m 间距', '5m 间距', '6.5m 间距
','FontSize',10,'FontName','宋体','Location','southeast');
```

附件 16 trees_model_10.m（带坡度的行列间苹果树树冠三维模型，MATLAB 程序）

```
clc,clear
%树冠 1 可视化模型

[theta,phi] = meshgrid(0:10:360,0:10:90);
X = 1.5*cos(phi*pi/180).*cos(theta*pi/180)+3.5;
Y = 1.5*cos(phi*pi/180).*sin(theta*pi/180)+3.5;
Z = 2*sin(phi*pi/180)+sqrt(2)*tan(deg2rad(10))*3.5;
ss1=surf(X, Y, Z);
ss1.FaceAlpha=0.4;
hold on;
[theta,phi2] = meshgrid(0:10:360,-90:10:0);
X2 = 1.5*cos(phi2*pi/180).*cos(theta.*pi/180)+3.5;
Y2 = 1.5*cos(phi2*pi/180).*sin(theta*pi/180)+3.5;
Z2 = 1*sin(phi2*pi/180)+sqrt(2)*tan(deg2rad(10))*3.5;
ss2=surf(X2, Y2, Z2);
ss2.FaceAlpha=0.4;
hold on;
[theta,phi] = meshgrid(0:10:360,0:10:90);
X3 = 1*cos(phi*pi/180).*cos(theta*pi/180)+3.5;
Y3 = 1*cos(phi*pi/180).*sin(theta*pi/180)+3.5;
Z3 = 1.5*sin(phi*pi/180)+sqrt(2)*tan(deg2rad(10))*3.5;
ss3=surf(X3, Y3, Z3);
ss3.FaceAlpha=0.4;
hold on;
X4 = 1*cos(phi2*pi/180).*cos(theta.*pi/180)+3.5;
Y4 = 1*cos(phi2*pi/180).*sin(theta*pi/180)+3.5;
Z4 = 0.75*sin(phi2*pi/180)+sqrt(2)*tan(deg2rad(10))*3.5;
ss4=surf(X4, Y4, Z4);
ss4.FaceAlpha=0.4;
hold on;
X5 = 0.5*cos(phi*pi/180).*cos(theta*pi/180)+3.5;
Y5 = 0.5*cos(phi*pi/180).*sin(theta*pi/180)+3.5;
Z5 = 1*sin(phi*pi/180)+sqrt(2)*tan(deg2rad(10))*3.5;
```

```
ss5=surf(X5, Y5, Z5);
ss5.FaceAlpha=0.4;
hold on;
X6 = 0.5*cos(phi2*pi/180).*cos(theta.*pi/180)+3.5;
Y6 = 0.5*cos(phi2*pi/180).*sin(theta*pi/180)+3.5;
Z6 = 0.375*sin(phi2*pi/180)+sqrt(2)*tan(deg2rad(10))*3.5;
ss6=surf(X6, Y6, Z6);
ss6.FaceAlpha=0.4;
hold on;

[theta,phi] = meshgrid(0:10:360,0:10:90);
X = 1.5*cos(phi*pi/180).*cos(theta*pi/180);
Y = 1.5*cos(phi*pi/180).*sin(theta*pi/180);
Z = 2*sin(phi*pi/180);
ss1=surf(X, Y, Z);
ss1.FaceAlpha=0.4;
hold on;
[theta,phi2] = meshgrid(0:10:360,-90:10:0);
X2 = 1.5*cos(phi2*pi/180).*cos(theta.*pi/180);
Y2 = 1.5*cos(phi2*pi/180).*sin(theta*pi/180);
Z2 = 1*sin(phi2*pi/180);
ss2=surf(X2, Y2, Z2);
ss2.FaceAlpha=0.4;
hold on;
[theta,phi] = meshgrid(0:10:360,0:10:90);
X3 = 1*cos(phi*pi/180).*cos(theta*pi/180);
Y3 = 1*cos(phi*pi/180).*sin(theta*pi/180);
Z3 = 1.5*sin(phi*pi/180);
ss3=surf(X3, Y3, Z3);
ss3.FaceAlpha=0.4;
hold on;
X4 = 1*cos(phi2*pi/180).*cos(theta.*pi/180);
Y4 = 1*cos(phi2*pi/180).*sin(theta*pi/180);
Z4 = 0.75*sin(phi2*pi/180);
ss4=surf(X4, Y4, Z4);
ss4.FaceAlpha=0.4;
hold on;
X5 = 0.5*cos(phi*pi/180).*cos(theta*pi/180);
Y5 = 0.5*cos(phi*pi/180).*sin(theta*pi/180);
Z5 = 1*sin(phi*pi/180);
ss5=surf(X5, Y5, Z5);
ss5.FaceAlpha=0.4;
hold on;
X6 = 0.5*cos(phi2*pi/180).*cos(theta.*pi/180);
```

```
Y6 = 0.5*cos(phi2*pi/180).*sin(theta*pi/180);
Z6 = 0.375*sin(phi2*pi/180);
ss6=surf(X6, Y6, Z6);
ss6.FaceAlpha=0.4;
hold on;

[theta,phi] = meshgrid(0:10:360,0:10:90);
X = 1.5*cos(phi*pi/180).*cos(theta*pi/180);
Y = 1.5*cos(phi*pi/180).*sin(theta*pi/180);
Z = 2*sin(phi*pi/180);
ss1=surf(X, Y, Z);
ss1.FaceAlpha=0.4;
hold on;
[theta,phi2] = meshgrid(0:10:360,-90:10:0);
X2 = 1.5*cos(phi2*pi/180).*cos(theta.*pi/180);
Y2 = 1.5*cos(phi2*pi/180).*sin(theta*pi/180);
Z2 = 1*sin(phi2*pi/180);
ss2=surf(X2, Y2, Z2);
ss2.FaceAlpha=0.4;
hold on;
[theta,phi] = meshgrid(0:10:360,0:10:90);
X3 = 1*cos(phi*pi/180).*cos(theta*pi/180);
Y3 = 1*cos(phi*pi/180).*sin(theta*pi/180);
Z3 = 1.5*sin(phi*pi/180);
ss3=surf(X3, Y3, Z3);
ss3.FaceAlpha=0.4;
hold on;
X4 = 1*cos(phi2*pi/180).*cos(theta.*pi/180);
Y4 = 1*cos(phi2*pi/180).*sin(theta*pi/180);
Z4 = 0.75*sin(phi2*pi/180);
ss4=surf(X4, Y4, Z4);
ss4.FaceAlpha=0.4;
hold on;
X5 = 0.5*cos(phi*pi/180).*cos(theta*pi/180);
Y5 = 0.5*cos(phi*pi/180).*sin(theta*pi/180);
Z5 = 1*sin(phi*pi/180);
ss5=surf(X5, Y5, Z5);
ss5.FaceAlpha=0.4;
hold on;
X6 = 0.5*cos(phi2*pi/180).*cos(theta.*pi/180);
Y6 = 0.5*cos(phi2*pi/180).*sin(theta*pi/180);
Z6 = 0.375*sin(phi2*pi/180);
ss6=surf(X6, Y6, Z6);
ss6.FaceAlpha=0.4;
```

```
hold on;

[theta,phi] = meshgrid(0:10:360,0:10:90);
X = 1.5*cos(phi*pi/180).*cos(theta*pi/180)-3.5;
Y = 1.5*cos(phi*pi/180).*sin(theta*pi/180)-3.5;
Z = 2*sin(phi*pi/180)-sqrt(2)*tan(deg2rad(10))*3.5;
ss1=surf(X, Y, Z);
ss1.FaceAlpha=0.4;
hold on;
[theta,phi2] = meshgrid(0:10:360,-90:10:0);
X2 = 1.5*cos(phi2*pi/180).*cos(theta.*pi/180)-3.5;
Y2 = 1.5*cos(phi2*pi/180).*sin(theta*pi/180)-3.5;
Z2 = 1*sin(phi2*pi/180)-sqrt(2)*tan(deg2rad(10))*3.5;
ss2=surf(X2, Y2, Z2);
ss2.FaceAlpha=0.4;
hold on;
[theta,phi] = meshgrid(0:10:360,0:10:90);
X3 = 1*cos(phi*pi/180).*cos(theta*pi/180)-3.5;
Y3 = 1*cos(phi*pi/180).*sin(theta*pi/180)-3.5;
Z3 = 1.5*sin(phi*pi/180)-sqrt(2)*tan(deg2rad(10))*3.5;
ss3=surf(X3, Y3, Z3);
ss3.FaceAlpha=0.4;
hold on;
X4 = 1*cos(phi2*pi/180).*cos(theta.*pi/180)-3.5;
Y4 = 1*cos(phi2*pi/180).*sin(theta*pi/180)-3.5;
Z4 = 0.75*sin(phi2*pi/180)-sqrt(2)*tan(deg2rad(10))*3.5;
ss4=surf(X4, Y4, Z4);
ss4.FaceAlpha=0.4;
hold on;
X5 = 0.5*cos(phi*pi/180).*cos(theta*pi/180)-3.5;
Y5 = 0.5*cos(phi*pi/180).*sin(theta*pi/180)-3.5;
Z5 = 1*sin(phi*pi/180)-sqrt(2)*tan(deg2rad(10))*3.5;
ss5=surf(X5, Y5, Z5);
ss5.FaceAlpha=0.4;
hold on;
X6 = 0.5*cos(phi2*pi/180).*cos(theta.*pi/180)-3.5;
Y6 = 0.5*cos(phi2*pi/180).*sin(theta*pi/180)-3.5;
Z6 = 0.375*sin(phi2*pi/180)-sqrt(2)*tan(deg2rad(10))*3.5;
ss6=surf(X6, Y6, Z6);
ss6.FaceAlpha=0.4;
hold on;

[theta,phi] = meshgrid(0:10:360,0:10:90);
X = 1.5*cos(phi*pi/180).*cos(theta*pi/180)+3.5;
```

```
Y = 1.5*cos(phi*pi/180).*sin(theta*pi/180)-3.5;
Z = 2*sin(phi*pi/180);
ss1=surf(X, Y, Z);
ss1.FaceAlpha=0.4;
hold on;
[theta,phi2] = meshgrid(0:10:360,-90:10:0);
X2 = 1.5*cos(phi2*pi/180).*cos(theta.*pi/180)+3.5;
Y2 = 1.5*cos(phi2*pi/180).*sin(theta*pi/180)-3.5;
Z2 = 1*sin(phi2*pi/180);
ss2=surf(X2, Y2, Z2);
ss2.FaceAlpha=0.4;
hold on;
[theta,phi] = meshgrid(0:10:360,0:10:90);
X3 = 1*cos(phi*pi/180).*cos(theta*pi/180)+3.5;
Y3 = 1*cos(phi*pi/180).*sin(theta*pi/180)-3.5;
Z3 = 1.5*sin(phi*pi/180);
ss3=surf(X3, Y3, Z3);
ss3.FaceAlpha=0.4;
hold on;
X4 = 1*cos(phi2*pi/180).*cos(theta.*pi/180)+3.5;
Y4 = 1*cos(phi2*pi/180).*sin(theta*pi/180)-3.5;
Z4 = 0.75*sin(phi2*pi/180);
ss4=surf(X4, Y4, Z4);
ss4.FaceAlpha=0.4;
hold on;
X5 = 0.5*cos(phi*pi/180).*cos(theta*pi/180)+3.5;
Y5 = 0.5*cos(phi*pi/180).*sin(theta*pi/180)-3.5;
Z5 = 1*sin(phi*pi/180);
ss5=surf(X5, Y5, Z5);
ss5.FaceAlpha=0.4;
hold on;
X6 = 0.5*cos(phi2*pi/180).*cos(theta.*pi/180)+3.5;
Y6 = 0.5*cos(phi2*pi/180).*sin(theta*pi/180)-3.5;
Z6 = 0.375*sin(phi2*pi/180);
ss6=surf(X6, Y6, Z6);
ss6.FaceAlpha=0.4;
hold on;

[theta,phi] = meshgrid(0:10:360,0:10:90);
X = 1.5*cos(phi*pi/180).*cos(theta*pi/180)-3.5;
Y = 1.5*cos(phi*pi/180).*sin(theta*pi/180)+3.5;
Z = 2*sin(phi*pi/180);
ss1=surf(X, Y, Z);
ss1.FaceAlpha=0.4;
```

```
hold on;
[theta,phi2] = meshgrid(0:10:360,-90:10:0);
X2 = 1.5*cos(phi2*pi/180).*cos(theta.*pi/180)-3.5;
Y2 = 1.5*cos(phi2*pi/180).*sin(theta*pi/180)+3.5;
Z2 = 1*sin(phi2*pi/180);
ss2=surf(X2, Y2, Z2);
ss2.FaceAlpha=0.4;
hold on;
[theta,phi] = meshgrid(0:10:360,0:10:90);
X3 = 1*cos(phi*pi/180).*cos(theta*pi/180)-3.5;
Y3 = 1*cos(phi*pi/180).*sin(theta*pi/180)+3.5;
Z3 = 1.5*sin(phi*pi/180);
ss3=surf(X3, Y3, Z3);
ss3.FaceAlpha=0.4;
hold on;
X4 = 1*cos(phi2*pi/180).*cos(theta.*pi/180)-3.5;
Y4 = 1*cos(phi2*pi/180).*sin(theta*pi/180)+3.5;
Z4 = 0.75*sin(phi2*pi/180);
ss4=surf(X4, Y4, Z4);
ss4.FaceAlpha=0.4;
hold on;
X5 = 0.5*cos(phi*pi/180).*cos(theta*pi/180)-3.5;
Y5 = 0.5*cos(phi*pi/180).*sin(theta*pi/180)+3.5;
Z5 = 1*sin(phi*pi/180);
ss5=surf(X5, Y5, Z5);
ss5.FaceAlpha=0.4;
hold on;
X6 = 0.5*cos(phi2*pi/180).*cos(theta.*pi/180)-3.5;
Y6 = 0.5*cos(phi2*pi/180).*sin(theta*pi/180)+3.5;
Z6 = 0.375*sin(phi2*pi/180);
ss6=surf(X6, Y6, Z6);
ss6.FaceAlpha=0.4;
hold on;

[theta,phi] = meshgrid(0:10:360,0:10:90);
X = 1.5*cos(phi*pi/180).*cos(theta*pi/180)+3.5;
Y = 1.5*cos(phi*pi/180).*sin(theta*pi/180);
Z = 2*sin(phi*pi/180)+sqrt(2)*tan(deg2rad(10))*3.5/2;
ss1=surf(X, Y, Z);
ss1.FaceAlpha=0.4;
hold on;
[theta,phi2] = meshgrid(0:10:360,-90:10:0);
X2 = 1.5*cos(phi2*pi/180).*cos(theta.*pi/180)+3.5;
Y2 = 1.5*cos(phi2*pi/180).*sin(theta*pi/180);
```

```
Z2 = 1*sin(phi2*pi/180)+sqrt(2)*tan(deg2rad(10))*3.5/2;
ss2=surf(X2, Y2, Z2);
ss2.FaceAlpha=0.4;
hold on;
[theta,phi] = meshgrid(0:10:360,0:10:90);
X3 = 1*cos(phi*pi/180).*cos(theta*pi/180)+3.5;
Y3 = 1*cos(phi*pi/180).*sin(theta*pi/180);
Z3 = 1.5*sin(phi*pi/180)+sqrt(2)*tan(deg2rad(10))*3.5/2;
ss3=surf(X3, Y3, Z3);
ss3.FaceAlpha=0.4;
hold on;
X4 = 1*cos(phi2*pi/180).*cos(theta.*pi/180)+3.5;
Y4 = 1*cos(phi2*pi/180).*sin(theta*pi/180);
Z4 = 0.75*sin(phi2*pi/180)+sqrt(2)*tan(deg2rad(10))*3.5/2;
ss4=surf(X4, Y4, Z4);
ss4.FaceAlpha=0.4;
hold on;
X5 = 0.5*cos(phi*pi/180).*cos(theta*pi/180)+3.5;
Y5 = 0.5*cos(phi*pi/180).*sin(theta*pi/180);
Z5 = 1*sin(phi*pi/180)+sqrt(2)*tan(deg2rad(10))*3.5/2;
ss5=surf(X5, Y5, Z5);
ss5.FaceAlpha=0.4;
hold on;
X6 = 0.5*cos(phi2*pi/180).*cos(theta.*pi/180)+3.5;
Y6 = 0.5*cos(phi2*pi/180).*sin(theta*pi/180);
Z6 = 0.375*sin(phi2*pi/180)+sqrt(2)*tan(deg2rad(10))*3.5/2;
ss6=surf(X6, Y6, Z6);
ss6.FaceAlpha=0.4;
hold on;

[theta,phi] = meshgrid(0:10:360,0:10:90);
X = 1.5*cos(phi*pi/180).*cos(theta*pi/180);
Y = 1.5*cos(phi*pi/180).*sin(theta*pi/180)+3.5;
Z = 2*sin(phi*pi/180)+sqrt(2)*tan(deg2rad(10))*3.5/2;
ss1=surf(X, Y, Z);
ss1.FaceAlpha=0.4;
hold on;
[theta,phi2] = meshgrid(0:10:360,-90:10:0);
X2 = 1.5*cos(phi2*pi/180).*cos(theta.*pi/180);
Y2 = 1.5*cos(phi2*pi/180).*sin(theta*pi/180)+3.5;
Z2 = 1*sin(phi2*pi/180)+sqrt(2)*tan(deg2rad(10))*3.5/2;
ss2=surf(X2, Y2, Z2);
ss2.FaceAlpha=0.4;
hold on;
```

```
[theta,phi] = meshgrid(0:10:360,0:10:90);
X3 = 1*cos(phi*pi/180).*cos(theta*pi/180);
Y3 = 1*cos(phi*pi/180).*sin(theta*pi/180)+3.5;
Z3 = 1.5*sin(phi*pi/180)+sqrt(2)*tan(deg2rad(10))*3.5/2;
ss3=surf(X3, Y3, Z3);
ss3.FaceAlpha=0.4;
hold on;
X4 = 1*cos(phi2*pi/180).*cos(theta.*pi/180);
Y4 = 1*cos(phi2*pi/180).*sin(theta*pi/180)+3.5;
Z4 = 0.75*sin(phi2*pi/180)+sqrt(2)*tan(deg2rad(10))*3.5/2;
ss4=surf(X4, Y4, Z4);
ss4.FaceAlpha=0.4;
hold on;
X5 = 0.5*cos(phi*pi/180).*cos(theta*pi/180);
Y5 = 0.5*cos(phi*pi/180).*sin(theta*pi/180)+3.5;
Z5 = 1*sin(phi*pi/180)+sqrt(2)*tan(deg2rad(10))*3.5/2;
ss5=surf(X5, Y5, Z5);
ss5.FaceAlpha=0.4;
hold on;
X6 = 0.5*cos(phi2*pi/180).*cos(theta.*pi/180);
Y6 = 0.5*cos(phi2*pi/180).*sin(theta*pi/180)+3.5;
Z6 = 0.375*sin(phi2*pi/180)+sqrt(2)*tan(deg2rad(10))*3.5/2;
ss6=surf(X6, Y6, Z6);
ss6.FaceAlpha=0.4;
hold on;

[theta,phi] = meshgrid(0:10:360,0:10:90);
X = 1.5*cos(phi*pi/180).*cos(theta*pi/180);
Y = 1.5*cos(phi*pi/180).*sin(theta*pi/180)-3.5;
Z = 2*sin(phi*pi/180)-sqrt(2)*tan(deg2rad(10))*3.5/2;
ss1=surf(X, Y, Z);
ss1.FaceAlpha=0.4;
hold on;
[theta,phi2] = meshgrid(0:10:360,-90:10:0);
X2 = 1.5*cos(phi2*pi/180).*cos(theta.*pi/180);
Y2 = 1.5*cos(phi2*pi/180).*sin(theta*pi/180)-3.5;
Z2 = 1*sin(phi2*pi/180)-sqrt(2)*tan(deg2rad(10))*3.5/2;
ss2=surf(X2, Y2, Z2);
ss2.FaceAlpha=0.4;
hold on;
[theta,phi] = meshgrid(0:10:360,0:10:90);
X3 = 1*cos(phi*pi/180).*cos(theta*pi/180);
Y3 = 1*cos(phi*pi/180).*sin(theta*pi/180)-3.5;
Z3 = 1.5*sin(phi*pi/180)-sqrt(2)*tan(deg2rad(10))*3.5/2;
```

```
ss3=surf(X3, Y3, Z3);
ss3.FaceAlpha=0.4;
hold on;
X4 = 1*cos(phi2*pi/180).*cos(theta.*pi/180);
Y4 = 1*cos(phi2*pi/180).*sin(theta*pi/180)-3.5;
Z4 = 0.75*sin(phi2*pi/180)-sqrt(2)*tan(deg2rad(10))*3.5/2;
ss4=surf(X4, Y4, Z4);
ss4.FaceAlpha=0.4;
hold on;
X5 = 0.5*cos(phi*pi/180).*cos(theta*pi/180);
Y5 = 0.5*cos(phi*pi/180).*sin(theta*pi/180)-3.5;
Z5 = 1*sin(phi*pi/180)-sqrt(2)*tan(deg2rad(10))*3.5/2;
ss5=surf(X5, Y5, Z5);
ss5.FaceAlpha=0.4;
hold on;
X6 = 0.5*cos(phi2*pi/180).*cos(theta.*pi/180);
Y6 = 0.5*cos(phi2*pi/180).*sin(theta*pi/180)-3.5;
Z6 = 0.375*sin(phi2*pi/180)-sqrt(2)*tan(deg2rad(10))*3.5/2;
ss6=surf(X6, Y6, Z6);
ss6.FaceAlpha=0.4;
hold on;

[theta,phi] = meshgrid(0:10:360,0:10:90);
X = 1.5*cos(phi*pi/180).*cos(theta*pi/180)-3.5;
Y = 1.5*cos(phi*pi/180).*sin(theta*pi/180);
Z = 2*sin(phi*pi/180)-sqrt(2)*tan(deg2rad(10))*3.5/2;
ss1=surf(X, Y, Z);
ss1.FaceAlpha=0.4;
hold on;
[theta,phi2] = meshgrid(0:10:360,-90:10:0);
X2 = 1.5*cos(phi2*pi/180).*cos(theta.*pi/180)-3.5;
Y2 = 1.5*cos(phi2*pi/180).*sin(theta*pi/180);
Z2 = 1*sin(phi2*pi/180)-sqrt(2)*tan(deg2rad(10))*3.5/2;
ss2=surf(X2, Y2, Z2);
ss2.FaceAlpha=0.4;
hold on;
[theta,phi] = meshgrid(0:10:360,0:10:90);
X3 = 1*cos(phi*pi/180).*cos(theta*pi/180)-3.5;
Y3 = 1*cos(phi*pi/180).*sin(theta*pi/180);
Z3 = 1.5*sin(phi*pi/180)-sqrt(2)*tan(deg2rad(10))*3.5/2;
ss3=surf(X3, Y3, Z3);
ss3.FaceAlpha=0.4;
hold on;
X4 = 1*cos(phi2*pi/180).*cos(theta.*pi/180)-3.5;
```

```
Y4 = 1*cos(phi2*pi/180).*sin(theta*pi/180);
Z4 = 0.75*sin(phi2*pi/180)-sqrt(2)*tan(deg2rad(10))*3.5/2;
ss4=surf(X4, Y4, Z4);
ss4.FaceAlpha=0.4;
hold on;
X5 = 0.5*cos(phi*pi/180).*cos(theta*pi/180)-3.5;
Y5 = 0.5*cos(phi*pi/180).*sin(theta*pi/180);
Z5 = 1*sin(phi*pi/180)-sqrt(2)*tan(deg2rad(10))*3.5/2;
ss5=surf(X5, Y5, Z5);
ss5.FaceAlpha=0.4;
hold on;
X6 = 0.5*cos(phi2*pi/180).*cos(theta.*pi/180)-3.5;
Y6 = 0.5*cos(phi2*pi/180).*sin(theta*pi/180);
Z6 = 0.375*sin(phi2*pi/180)-sqrt(2)*tan(deg2rad(10))*3.5/2;
ss6=surf(X6, Y6, Z6);
ss6.FaceAlpha=0.4;
hold on;

xlabel('南北方向（单位：m）','FontSize',15,'FontName','宋体')
ylabel('东西方向（单位：m）','FontSize',15,'FontName','宋体')
axis equal;
```

附件 17 trees_model_10_light.m（带坡度的行列间苹果树树冠受光模型，MATLAB 代码）

```
clc,clear
for d = 5:1.5:6.5

%树冠模型参数
a11=1.5;b11=1.5;c11=2;
a12=1.5;b12=1.5;c12=1;
center1=[0,0,0];

% 树冠模型参数
a31 = 1.5; b31 = 1.5; c31 = 2;
a32 = 1.5; b32 = 1.5; c32 = 1;
center3 = [0, -d, -sqrt(2)*tan(deg2rad(10))*d/2];  % 将中心点在 x 轴上移动 5 个单位

% 树冠模型参数
a21 = 1.5; b21 = 1.5; c21 = 2;
a22 = 1.5; b22 = 1.5; c22 = 1;
center2 = [0, d, sqrt(2)*tan(deg2rad(10))*d/2];  % 将中心点在 x 轴上移动 5 个单位
```

```
% 树冠模型参数
a41 = 1.5; b41 = 1.5; c41 = 2;
a42 = 1.5; b42 = 1.5; c42 = 1;
center4 = [d, 0, sqrt(2)*tan(deg2rad(10))*d/2];  % 将中心点在 x 轴上移动 5 个单
位

% 树冠模型参数
a51 = 1.5; b51 = 1.5; c51 = 2;
a52 = 1.5; b52 = 1.5; c52 = 1;
center5 = [-d, 0, -sqrt(2)*tan(deg2rad(10))*d/2];  % 将中心点在 x 轴上移动 5 个
单位

% 树冠模型参数
a61 = 1.5; b61 = 1.5; c61 = 2;
a62 = 1.5; b62 = 1.5; c62 = 1;
center6 = [-d, -d, -sqrt(2)*tan(deg2rad(10))*d];  % 将中心点在 x 轴上移动 5 个
单位

% 树冠模型参数
a71 = 1.5; b71 = 1.5; c71 = 2;
a72 = 1.5; b72 = 1.5; c72 = 1;
center7 = [d, -d, 0];  % 将中心点在 x 轴上移动 5 个单位

a81 = 1.5; b81 = 1.5; c81 = 2;
a82 = 1.5; b82 = 1.5; c82 = 1;
center8 = [-d, d, 0];  % 将中心点在 x 轴上移动 5 个单位

% 树冠模型参数
a91 = 1.5; b91 = 1.5; c91 = 2;
a92 = 1.5; b92 = 1.5; c92 = 1;
center9 = [d, d, sqrt(2)*tan(deg2rad(10))*d];  % 将中心点在 x 轴上移动 5 个单位



%两个角度
elevation_angle=[15.25329928
26.74728536
37.30069887
46.07805656
51.73268607
52.76041559
48.80598458
41.08804954
31.11833248
```

```
19.92684826
28.8681958
40.86994525
52.88480367
64.46002994
74.10822164
76.37468044
68.72145662
57.6097592
45.69154488
33.64999697
15.00342485
26.47644953
36.99593591
45.72815553
51.34071796
52.35927901
48.43739643
40.7661971
30.83552072
19.67006902
10.43771148
18.92008084
25.41121023
29.24919056
29.9130712
27.30034527
21.79243076
14.02900813
]';
azimuth_angle=[102.3059678
112.6554838
125.2734277
141.6402462
162.8883742
187.4492199
210.4519166
228.7179887
242.6460181
253.7456587
80.96109056
89.19354329
99.08022618
113.6313597
```

```
142.3663618
197.7927875
237.5568977
255.9546618
267.1545624
275.8472996
102.6339155
112.9897418
125.6048415
141.9284839
163.0410832
187.380392
210.2042401
228.3963008
242.3106454
253.415112
129.7484118
140.9571165
154.0680759
168.9518354
184.7639009
200.1609288
214.0333748
225.9820887
]';

    % 将角度转换为弧度

for n=1:1:38
    elevation_rad = deg2rad(elevation_angle(n));
    azimuth_rad = deg2rad(azimuth_angle(n));

    % 光线方向向量
    direction_vector = [cos(azimuth_rad) * cos(elevation_rad);
                        sin(azimuth_rad) * cos(elevation_rad);
                        sin(elevation_rad)];
    %光线参数
    point1=[0,0,-2.5];
    point22=point1+direction_vector';
    p=1;
    while point22(3)<5
        point22=point22+direction_vector';
        p=p+1;
    end
    direction=point22-point1;
    num=0;
%步长计算
numIntersectionPoints1=0;
S=0;
for x =-10:0.1:10
    for y =-10:0.1:10
        point1=[0,0,-2.5];
        point1=point1+[x,0,0]+[0,y,0];
        point2=point22+[x,0,0]+[0,y,0];
        %plot3([point1(1),point2(1)],[point1(2),point2(2)],[point1(2),point2
(2)])
        bool = 0;
        for t = 0:0.01:1  % 步长为0.01
            point = point2 - t * direction;
            % 判断点是否在半椭球内
            if point(3)>=0
                    bool1 = (((point(1)-center3(1))/a31)^2 + ((point(2)-
center3(2))/b31)^2 + ((point(3)-center3(3))/c31)^2 <= 1) & bool == 0;
                    bool2 = (((point(1)-center2(1))/a21)^2 + ((point(2)-
center2(2))/b21)^2 + ((point(3)-center2(3))/c21)^2 <= 1) & bool == 0;
                    bool3 = (((point(1)-center1(1))/a11)^2 + ((point(2)-
center1(2))/b11)^2 + ((point(3)-center1(3))/c11)^2 <= 1) & bool == 0;
                    bool4 = (((point(1)-center4(1))/a41)^2 + ((point(2)-
center4(2))/b41)^2 + ((point(3)-center4(3))/c41)^2 <= 1) & bool == 0;
                    bool5 = (((point(1)-center5(1))/a51)^2 + ((point(2)-
center5(2))/b51)^2 + ((point(3)-center5(3))/c51)^2 <= 1) & bool == 0;
                    bool6 = (((point(1)-center6(1))/a61)^2 + ((point(2)-
center6(2))/b61)^2 + ((point(3)-center6(3))/c61)^2 <= 1) & bool == 0;
                    bool7 = (((point(1)-center7(1))/a71)^2 + ((point(2)-
center7(2))/b71)^2 + ((point(3)-center7(3))/c71)^2 <= 1) & bool == 0;
                    bool8 = (((point(1)-center8(1))/a81)^2 + ((point(2)-
center8(2))/b81)^2 + ((point(3)-center8(3))/c81)^2 <= 1) & bool == 0;
                    bool9 = (((point(1)-center9(1))/a91)^2 + ((point(2)-
center9(2))/b91)^2 + ((point(3)-center9(3))/c91)^2 <= 1) & bool == 0;
                    if bool1 || bool2 || bool3 ||bool4 ||bool5||bool6 ||
bool7 ||bool8 ||bool9
                        numIntersectionPoints1 = numIntersectionPoints1 +
bool3;
                        if bool3 ==1
                            A = sqrt((point(1)/a11^2)^2 + (point(2)/b11^2)^2 +
(point(3)/c11^2)^2);
                            sintheta=(point(2)/b11^2)/A;
                            sintheta=sqrt(1-sintheta^2);
                            S=S+0.01*abs(sintheta);
                        end
                        bool = 1;
                        break
                    end

            elseif point(3)<0
                    bool1 = (((point(1)-center3(1))/a32)^2 + ((point(2)-
center3(2))/b32)^2 + ((point(3)-center3(3))/c32)^2 <= 1) & bool == 0;
                    bool2 = (((point(1)-center2(1))/a22)^2 + ((point(2)-
center2(2))/b22)^2 + ((point(3)-center2(3))/c22)^2 <= 1) & bool == 0;
                    bool3 = (((point(1)-center1(1))/a12)^2 + ((point(2)-
center1(2))/b12)^2 + ((point(3)-center1(3))/c12)^2 <= 1) & bool == 0;
                    bool4 = (((point(1)-center4(1))/a42)^2 + ((point(2)-
center4(2))/b42)^2 + ((point(3)-center4(3))/c42)^2 <= 1) & bool == 0;
                    bool5 = (((point(1)-center5(1))/a52)^2 + ((point(2)-
center5(2))/b52)^2 + ((point(3)-center5(3))/c52)^2 <= 1) & bool == 0;
                    bool6 = (((point(1)-center6(1))/a62)^2 + ((point(2)-
center6(2))/b62)^2 + ((point(3)-center6(3))/c62)^2 <= 1) & bool == 0;
                    bool7 = (((point(1)-center7(1))/a72)^2 + ((point(2)-
center7(2))/b72)^2 + ((point(3)-center7(3))/c72)^2 <= 1) & bool == 0;
                    bool8 = (((point(1)-center8(1))/a82)^2 + ((point(2)-
center8(2))/b82)^2 + ((point(3)-center8(3))/c82)^2 <= 1) & bool == 0;
                    bool9 = (((point(1)-center9(1))/a92)^2 + ((point(2)-
center9(2))/b92)^2 + ((point(3)-center9(3))/c92)^2 <= 1) & bool == 0;

                    if bool1 || bool2 || bool3 ||bool4 ||bool5||bool6 ||
bool7 ||bool8 ||bool9
                        numIntersectionPoints1 = numIntersectionPoints1 +
bool3;
                        if bool3 ==1
                            A = sqrt((point(1)/a11^2)^2 + (point(2)/b11^2)^2 +
(point(3)/c11^2)^2);
                            sintheta=(point(2)/b11^2)/A;
                            sintheta=sqrt(1-sintheta^2);
                            S=S+0.01*abs(sintheta);
                        end
                        bool = 1;
                        %scatter3(point(1), point(2), point(3), 'filled',
'MarkerFaceColor','y', 'MarkerFaceAlpha', 1);
                        break
                    end
            hold on;
            end
        end
    end
end
%disp(numIntersectionPoints1)
disp([num2str(n),'受光面积为：',num2str(S)])
end
end
axis equal;
```

附件 18 trees_model_10_light_area.m（带坡度的行列间苹果树树冠受光面积拟合模型，MATLAB 代码）

```
y1 = [
0.0068002
10.3097
10.622
9.438
9.2057
10.2259
12.7375
7.9064
] ;
y2 = [
0.96758
14.7647
13.0191
11.8685
11.6315
12.3926
14.5989
13.2969
];
y3 = [1.5723
16.0132
13.0191
11.8685
11.6315
12.3926
14.5989
13.4217
];

y11=[
0
9.6175
9.4585
8.0014
7.6944
8.9229
9.9125
2.1648]';
y21=[0.96758
14.7647
12.9061
10.912
10.5601
11.9382
14.5328
7.0898]';
y31=[1.5723
16.0132
13.0191
11.8685
11.6315
12.3926
14.5989
10.2338]';

%y4 = [
%] * 0.04;
x = [8 9 10 11 12 13 14 15 16 17];
x2=[9 10 11 12 13 14 15 16];


y1=y1';y2=y2';y3=y3';

%y4=y4';

% 设置多项式阶数
n = 6;

% 进行四条曲线的曲线拟合
p1 = polyfit(x2, y1, n);
p2 = polyfit(x2, y2, n);
p3 = polyfit(x2, y3, n);
p4 = polyfit(x2, y11, n);
p5 = polyfit(x2, y21, n);
```

```
p6 = polyfit(x2, y31, n);
%p4 = polyfit(x2, y4, n);

% 生成一系列 x 值
x_fit = linspace(min(x), max(x), 100);

% 计算四条拟合曲线对应的 y 值
y_fit1 = polyval(p1, x_fit);
y_fit2 = polyval(p2, x_fit);
y_fit3 = polyval(p3, x_fit);
y_fit4 = polyval(p4, x_fit);
y_fit5 = polyval(p5, x_fit);
y_fit6 = polyval(p6, x_fit);
%y_fit4 = polyval(p4, x_fit);

% 绘制原始数据和拟合曲线
figure;
hold on;
scatter(x2, y1, 'o','filled','MarkerFaceColor',[0.5,0,0.5]);
scatter(x2, y2, 'o', 'filled','MarkerFaceColor',[0,0.5,0.5]);
scatter(x2, y3, 'o', 'filled','MarkerFaceColor','b');
%scatter(x2, y4, 'o', 'filled','MarkerFaceColor',[0.5,0,0.5]);
plot(x_fit, y_fit1,'color',[0.5,0,0.5],'LineWidth',1.5);
plot(x_fit, y_fit2,'color',[0,0.5,0.5],'LineWidth',1.5);
plot(x_fit, y_fit3,'color','b','LineWidth',1.5);
plot(x_fit, y_fit4,'color',[0.5,0,0.5],'LineWidth',1,'LineStyle','--');
plot(x_fit, y_fit5,'color',[0,0.5,0.5],'LineWidth',1,'LineStyle','--');
plot(x_fit, y_fit6,'color','b','LineWidth',1,'LineStyle','--');
%plot(x_fit, y_fit4,'color',[0.5,0,0.5],'LineWidth',1.5);
hold off;
xlabel('时间（单位：小时）','FontSize',12,'FontName','宋体');
ylim([-3 16]);
xlim([8 17])
ylabel('树冠受光面积（单位：平方米）','FontSize',12,'FontName','宋体');
legend('','','', '3.5m 间距（有坡度）', '5m 间距（有坡度）', '6.5m 间距（有坡
度）', '3.5m 间距(无坡度)', '5m 间距（无坡度）', '6.5m 间距（无坡度）
','FontSize',10,'FontName','宋体','Location','southeast');
```

附件 19 trees_model_10_dark.m（带坡度的行列间苹果树树冠阴影模型，MATLAB 程序）

```
%树冠模型参数
a11=1.5;b11=1.5;c11=2;a21=1;b21=1;c21=1.5;a31=0.5;b31=0.5;c31=1;
a12=1.5;b12=1.5;c12=1;a22=1;b22=1;c22=0.75;a32=0.5;b32=0.5;c32=0.375;
```

```
center2=[3.5,3.5,sqrt(2)*tan(deg2rad(10))*3.5];center3=[3.5,0,sqrt(2)*tan(d
eg2rad(10))*3.5/2];center4=[3.5,-3.5,0];
center5=[0,3.5,sqrt(2)*tan(deg2rad(10))*3.5/2];center1=[0,0,0];center6=[0,-
3.5,-sqrt(2)*tan(deg2rad(10))*3.5/2];
center7=[-3.5,3.5,0];center8=[-3.5,0,-
sqrt(2)*tan(deg2rad(10))*3.5/2];center9=[-3.5,-3.5,-
sqrt(2)*tan(deg2rad(10))*3.5];


elevation_angle=[10.43771148
18.92008084
25.41121023
29.24919056
29.9130712
27.30034527
21.79243076
14.02900813
];
azimuth_angle=[129.7484118
140.9571165
154.0680759
168.9518354
184.7639009
200.1609288
214.0333748
225.9820887
];
elevation_angle=elevation_angle';
azimuth_angle=azimuth_angle';
for s=1:1:8
    azimuth_angle(s)=180-azimuth_angle(s);
end

for n=1:1:10
    % 将角度转换为弧度
    elevation_rad = deg2rad(elevation_angle(n));
    azimuth_rad = deg2rad(azimuth_angle(n));

    % 光线方向向量
    direction_vector = [cos(azimuth_rad) * cos(elevation_rad);
                        sin(azimuth_rad) * cos(elevation_rad);
                        sin(elevation_rad)];
    %光线参数
    point1=[0,0,-2.5];
    point22=point1+direction_vector';
    p=1;
    while point22(3)<5
        point22=point22+direction_vector';
        p=p+1;
    end

    direction=point22-point1;
    num=0;
    %步长计算
    for x =-60:0.2:60
        for y =-60:0.2:60
            point1=[0,0,-2.5];
            point1=point1+[x,0,0]+[0,y,0];
            point2=point22+[x,0,0]+[0,y,0];
            numIntersectionPoints1=0;
            numIntersectionPoints2=0;
            numIntersectionPoints3=0;
            direction=point2-point1;
            for t = 0:0.01:1  % 步长为 0.01
                point = point1 + t * direction;
                % 判断点是否在半椭球内
                if point(3)>=0
                    if ((point(1)-center1(1))/a31)^2 + ((point(2)-
center1(2))/b31)^2 + ((point(3)-center1(3))/c31)^2 <= 1
                        numIntersectionPoints3 = numIntersectionPoints3 + 1;
                    elseif ((point(1)-center1(1))/a21)^2 + ((point(2)-
center1(2))/b21)^2 + ((point(3)-center1(3))/c21)^2 <= 1
                        numIntersectionPoints2 = numIntersectionPoints2 + 1;
                    elseif ((point(1)-center1(1))/a11)^2 + ((point(2)-
center1(2))/b11)^2 + ((point(3)-center1(3))/c11)^2 <= 1
                        numIntersectionPoints1 = numIntersectionPoints1 + 1;
                    end
                    if ((point(1)-center2(1))/a31)^2 + ((point(2)-
center2(2))/b31)^2 + ((point(3)-center2(3))/c31)^2 <= 1
                        numIntersectionPoints3 = numIntersectionPoints3 + 1;
                    elseif ((point(1)-center2(1))/a21)^2 + ((point(2)-
center2(2))/b21)^2 + ((point(3)-center2(3))/c21)^2 <= 1
                        numIntersectionPoints2 = numIntersectionPoints2 + 1;
                    elseif ((point(1)-center2(1))/a11)^2 + ((point(2)-
center2(2))/b11)^2 + ((point(3)-center2(3))/c11)^2 <= 1
                        numIntersectionPoints1 = numIntersectionPoints1 + 1;
                    end
                    if ((point(1)-center3(1))/a31)^2 + ((point(2)-
center3(2))/b31)^2 + ((point(3)-center3(3))/c31)^2 <= 1
                        numIntersectionPoints3 = numIntersectionPoints3 + 1;
                    elseif ((point(1)-center3(1))/a21)^2 + ((point(2)-
center3(2))/b21)^2 + ((point(3)-center3(3))/c21)^2 <= 1
                        numIntersectionPoints2 = numIntersectionPoints2 + 1;
                    elseif ((point(1)-center3(1))/a11)^2 + ((point(2)-
center3(2))/b11)^2 + ((point(3)-center3(3))/c11)^2 <= 1
                        numIntersectionPoints1 = numIntersectionPoints1 + 1;
                    end
                    if ((point(1)-center4(1))/a31)^2 + ((point(2)-
center4(2))/b31)^2 + ((point(3)-center4(3))/c31)^2 <= 1
                        numIntersectionPoints3 = numIntersectionPoints3 + 1;
                    elseif ((point(1)-center4(1))/a21)^2 + ((point(2)-
center4(2))/b21)^2 + ((point(3)-center4(3))/c21)^2 <= 1
                        numIntersectionPoints2 = numIntersectionPoints2 + 1;
                    elseif ((point(1)-center4(1))/a11)^2 + ((point(2)-
center4(2))/b11)^2 + ((point(3)-center4(3))/c11)^2 <= 1
                        numIntersectionPoints1 = numIntersectionPoints1 + 1;
                    end
                    if ((point(1)-center5(1))/a31)^2 + ((point(2)-
center5(2))/b31)^2 + ((point(3)-center5(3))/c31)^2 <= 1
                        numIntersectionPoints3 = numIntersectionPoints3 + 1;
                    elseif ((point(1)-center5(1))/a21)^2 + ((point(2)-
center5(2))/b21)^2 + ((point(3)-center5(3))/c21)^2 <= 1
                        numIntersectionPoints2 = numIntersectionPoints2 + 1;
                    elseif ((point(1)-center5(1))/a11)^2 + ((point(2)-
center5(2))/b11)^2 + ((point(3)-center5(3))/c11)^2 <= 1
                        numIntersectionPoints1 = numIntersectionPoints1 + 1;
                    end
                    if ((point(1)-center6(1))/a31)^2 + ((point(2)-
center6(2))/b31)^2 + ((point(3)-center6(3))/c31)^2 <= 1
                        numIntersectionPoints3 = numIntersectionPoints3 + 1;
                    elseif ((point(1)-center6(1))/a21)^2 + ((point(2)-
center6(2))/b21)^2 + ((point(3)-center6(3))/c21)^2 <= 1
                        numIntersectionPoints2 = numIntersectionPoints2 + 1;
                    elseif ((point(1)-center6(1))/a11)^2 + ((point(2)-
center6(2))/b11)^2 + ((point(3)-center6(3))/c11)^2 <= 1
                        numIntersectionPoints1 = numIntersectionPoints1 + 1;
                    end
                    if ((point(1)-center7(1))/a31)^2 + ((point(2)-
center7(2))/b31)^2 + ((point(3)-center7(3))/c31)^2 <= 1
                        numIntersectionPoints3 = numIntersectionPoints3 + 1;
```

```
                elseif ((point(1)-center7(1))/a21)^2 + ((point(2)-
center7(2))/b21)^2 + ((point(3)-center7(3))/c21)^2 <= 1
                    numIntersectionPoints2 = numIntersectionPoints2 + 1;
                elseif ((point(1)-center7(1))/a11)^2 + ((point(2)-
center7(2))/b11)^2 + ((point(3)-center7(3))/c11)^2 <= 1
                    numIntersectionPoints1 = numIntersectionPoints1 + 1;
                end
                if ((point(1)-center8(1))/a31)^2 + ((point(2)-
center8(2))/b31)^2 + ((point(3)-center8(3))/c31)^2 <= 1
                    numIntersectionPoints3 = numIntersectionPoints3 + 1;
                elseif ((point(1)-center8(1))/a21)^2 + ((point(2)-
center8(2))/b21)^2 + ((point(3)-center8(3))/c21)^2 <= 1
                    numIntersectionPoints2 = numIntersectionPoints2 + 1;
                elseif ((point(1)-center8(1))/a11)^2 + ((point(2)-
center8(2))/b11)^2 + ((point(3)-center8(3))/c11)^2 <= 1
                    numIntersectionPoints1 = numIntersectionPoints1 + 1;
                end
                if ((point(1)-center9(1))/a31)^2 + ((point(2)-
center9(2))/b31)^2 + ((point(3)-center9(3))/c31)^2 <= 1
                    numIntersectionPoints3 = numIntersectionPoints3 + 1;
                elseif ((point(1)-center9(1))/a21)^2 + ((point(2)-
center9(2))/b21)^2 + ((point(3)-center9(3))/c21)^2 <= 1
                    numIntersectionPoints2 = numIntersectionPoints2 + 1;
                elseif ((point(1)-center9(1))/a11)^2 + ((point(2)-
center9(2))/b11)^2 + ((point(3)-center9(3))/c11)^2 <= 1
                    numIntersectionPoints1 = numIntersectionPoints1 + 1;
                end
            elseif point(3)<0
                if ((point(1)-center1(1))/a32)^2 + ((point(2)-
center1(2))/b32)^2 + ((point(3)-center1(3))/c32)^2 <= 1
                    numIntersectionPoints3 = numIntersectionPoints3 + 1;
                elseif ((point(1)-center1(1))/a22)^2 + ((point(2)-
center1(2))/b22)^2 + ((point(3)-center1(3))/c22)^2 <= 1
                    numIntersectionPoints2 = numIntersectionPoints2 + 1;
                elseif ((point(1)-center1(1))/a12)^2 + ((point(2)-
center1(2))/b12)^2 + ((point(3)-center1(3))/c12)^2 <= 1
                    numIntersectionPoints1 = numIntersectionPoints1 + 1;
                end
                if ((point(1)-center2(1))/a32)^2 + ((point(2)-
center2(2))/b32)^2 + ((point(3)-center2(3))/c32)^2 <= 1
                    numIntersectionPoints3 = numIntersectionPoints3 + 1;
                elseif ((point(1)-center2(1))/a22)^2 + ((point(2)-
center2(2))/b22)^2 + ((point(3)-center2(3))/c22)^2 <= 1
                    numIntersectionPoints2 = numIntersectionPoints2 + 1;
                elseif ((point(1)-center2(1))/a12)^2 + ((point(2)-
center2(2))/b12)^2 + ((point(3)-center2(3))/c12)^2 <= 1
                    numIntersectionPoints1 = numIntersectionPoints1 + 1;
                end
                if ((point(1)-center3(1))/a32)^2 + ((point(2)-
center3(2))/b32)^2 + ((point(3)-center3(3))/c32)^2 <= 1
                    numIntersectionPoints3 = numIntersectionPoints3 + 1;
                elseif ((point(1)-center3(1))/a22)^2 + ((point(2)-
center3(2))/b22)^2 + ((point(3)-center3(3))/c22)^2 <= 1
                    numIntersectionPoints2 = numIntersectionPoints2 + 1;
                elseif ((point(1)-center3(1))/a12)^2 + ((point(2)-
center3(2))/b12)^2 + ((point(3)-center3(3))/c12)^2 <= 1
                    numIntersectionPoints1 = numIntersectionPoints1 + 1;
                end
                if ((point(1)-center4(1))/a31)^2 + ((point(2)-
center4(2))/b31)^2 + ((point(3)-center4(3))/c31)^2 <= 1
                    numIntersectionPoints3 = numIntersectionPoints3 + 1;
                elseif ((point(1)-center4(1))/a21)^2 + ((point(2)-
center4(2))/b21)^2 + ((point(3)-center4(3))/c21)^2 <= 1
                    numIntersectionPoints2 = numIntersectionPoints2 + 1;
                elseif ((point(1)-center4(1))/a11)^2 + ((point(2)-
center4(2))/b11)^2 + ((point(3)-center4(3))/c11)^2 <= 1
                    numIntersectionPoints1 = numIntersectionPoints1 + 1;
                end
                if ((point(1)-center5(1))/a31)^2 + ((point(2)-
center5(2))/b31)^2 + ((point(3)-center5(3))/c31)^2 <= 1
                    numIntersectionPoints3 = numIntersectionPoints3 + 1;
                elseif ((point(1)-center5(1))/a21)^2 + ((point(2)-
center5(2))/b21)^2 + ((point(3)-center5(3))/c21)^2 <= 1
                    numIntersectionPoints2 = numIntersectionPoints2 + 1;
                elseif ((point(1)-center5(1))/a11)^2 + ((point(2)-
center5(2))/b11)^2 + ((point(3)-center5(3))/c11)^2 <= 1
                    numIntersectionPoints1 = numIntersectionPoints1 + 1;
                end
                if ((point(1)-center6(1))/a31)^2 + ((point(2)-
center6(2))/b31)^2 + ((point(3)-center6(3))/c31)^2 <= 1
                    numIntersectionPoints3 = numIntersectionPoints3 + 1;
                elseif ((point(1)-center6(1))/a21)^2 + ((point(2)-
center6(2))/b21)^2 + ((point(3)-center6(3))/c21)^2 <= 1
                    numIntersectionPoints2 = numIntersectionPoints2 + 1;
                elseif ((point(1)-center6(1))/a11)^2 + ((point(2)-
center6(2))/b11)^2 + ((point(3)-center6(3))/c11)^2 <= 1
                    numIntersectionPoints1 = numIntersectionPoints1 + 1;
                end
                if ((point(1)-center7(1))/a31)^2 + ((point(2)-
center7(2))/b31)^2 + ((point(3)-center7(3))/c31)^2 <= 1
                    numIntersectionPoints3 = numIntersectionPoints3 + 1;
                elseif ((point(1)-center7(1))/a21)^2 + ((point(2)-
center7(2))/b21)^2 + ((point(3)-center7(3))/c21)^2 <= 1
                    numIntersectionPoints2 = numIntersectionPoints2 + 1;
                elseif ((point(1)-center7(1))/a11)^2 + ((point(2)-
center7(2))/b11)^2 + ((point(3)-center7(3))/c11)^2 <= 1
                    numIntersectionPoints1 = numIntersectionPoints1 + 1;
                end
                if ((point(1)-center8(1))/a31)^2 + ((point(2)-
center8(2))/b31)^2 + ((point(3)-center8(3))/c31)^2 <= 1
                    numIntersectionPoints3 = numIntersectionPoints3 + 1;
                elseif ((point(1)-center8(1))/a21)^2 + ((point(2)-
center8(2))/b21)^2 + ((point(3)-center8(3))/c21)^2 <= 1
                    numIntersectionPoints2 = numIntersectionPoints2 + 1;
                elseif ((point(1)-center8(1))/a11)^2 + ((point(2)-
center8(2))/b11)^2 + ((point(3)-center8(3))/c11)^2 <= 1
                    numIntersectionPoints1 = numIntersectionPoints1 + 1;
                end
                if ((point(1)-center9(1))/a31)^2 + ((point(2)-
center9(2))/b31)^2 + ((point(3)-center9(3))/c31)^2 <= 1
                    numIntersectionPoints3 = numIntersectionPoints3 + 1;
                elseif ((point(1)-center9(1))/a21)^2 + ((point(2)-
center9(2))/b21)^2 + ((point(3)-center9(3))/c21)^2 <= 1
                    numIntersectionPoints2 = numIntersectionPoints2 + 1;
                elseif ((point(1)-center9(1))/a11)^2 + ((point(2)-
center9(2))/b11)^2 + ((point(3)-center9(3))/c11)^2 <= 1
                    numIntersectionPoints1 = numIntersectionPoints1 + 1;
                end
            end
        end

sum=numIntersectionPoints3*0.2+numIntersectionPoints2*0.8+numIntersectionPo
ints1*0.5;

        if sum>0
            num=num+1;
            hold on;
        end

    end
end
disp([num2str(num)])
```

```
end
```

附件 20 trees_model_10_dark_area.m（带坡度的行列间苹果树树冠阴影面积拟合模型，MATLAB 程序）

```
y1 = [
3649
2951
2601
2439
2945
3705
5382
] * 0.04;
y2 = [
4588
4064
3167
2956
3700
4974
6454
] * 0.04;
y3 = [5018
4498
3624
3401
4123
5196
6983
] * 0.04;
%y4 = [
%] * 0.04;
x = [8 9 10 11 12 13 14 15 16 17];
x2=[10 11 12 13 14 15 16];
q=[
2.0655
1.5317
1.3259
1.1999
1.098
0.9878
0.8148
];
```

```
y1=y1';y2=y2';y3=y3';
for i=1:1:10
    y1(i)=q(i)*y1(i);
    y2(i)=q(i)*y2(i);
    y3(i)=q(i)*y3(i);
end
%y4=y4';

% 设置多项式阶数
n = 4;

% 进行四条曲线的曲线拟合
p1 = polyfit(x, y1, n);
p2 = polyfit(x, y2, n);
p3 = polyfit(x, y3, n);
%p4 = polyfit(x2, y4, n);

% 生成一系列 x 值
x_fit = linspace(min(x), max(x), 100);

% 计算四条拟合曲线对应的 y 值
y_fit1 = polyval(p1, x_fit);
y_fit2 = polyval(p2, x_fit);
y_fit3 = polyval(p3, x_fit);
%y_fit4 = polyval(p4, x_fit);

% 绘制原始数据和拟合曲线
figure;
hold on;
scatter(x, y1, 'o','filled','MarkerFaceColor',[0.5,0,0.5]);
scatter(x, y2, 'o', 'filled','MarkerFaceColor',[0,0.5,0.5]);
scatter(x, y3, 'o', 'filled','MarkerFaceColor','b');
%scatter(x2, y4, 'o', 'filled','MarkerFaceColor',[0.5,0,0.5]);
plot(x_fit, y_fit1,'color',[0.5,0,0.5],'LineWidth',1.5);
plot(x_fit, y_fit2,'color',[0,0.5,0.5],'LineWidth',1.5);
plot(x_fit, y_fit3,'color','b','LineWidth',1.5);
%plot(x_fit, y_fit4,'color',[0.5,0,0.5],'LineWidth',1.5);
hold off;
xlabel('时间（单位：小时）','FontSize',12,'FontName','宋体');
ylim([0 450]);
xlim([8 16])
ylabel('树冠阴影面积（单位：平方米）','FontSize',12,'FontName','宋体');
legend('','','', '3.5m 间距', '5m 间距', '6.5m 间距
','FontSize',10,'FontName','宋体','Location','northeast');
```