基于天文学与三维模型的苹果树树冠优化设计

摘要

本研究基于图形三维建模和仿真技术，探讨了优化苹果树树冠形状以提升其日照接收能力的可能性，旨在通过改进树冠设计，根据受光时长显著增强苹果树的光合作用效率及果实产量。通过数学建模对果树树冠的各个形状参数进行设计，使得洛川县苹果树树冠超受光面积充分大具有现实意义，这也是我们亟待解决的问题。

针对问题一，本文建立了一种**天文学模型**。首先，需要确定洛川县的纬度，再查询夏至日的日期并计算出该日的**太阳赤纬**，然后运用**太阳时角**和**太阳高度角**的公式。从而可以计算出洛川县夏至日间隔一小时的太阳高度角。**结果详见表 1**。

针对问题二，本我们构建了**苹果树球形和圆锥形树冠超受光面积模型**。首先，建立**笛卡尔坐标系**并在其中建立球形和圆锥形的**几何模型**，并对几何模型进行**网格化离散**处理。接着，采用问题一中建立的太阳高度角公式结合太阳方位角公式建立出**太阳位置模型**：将上述模型结合进行**光线追踪**计算，找出累计受光超过八小时的面元，再对这些面元运用**累加法**求解，最终求得了**球形树冠**一天**超过八小时**的**受光面积为 11.38m²，光效率**为 **90.6%，圆锥形树冠**一天**超过八小时**的**受光面积**为 **3.58 m²，受光效率**为 **47.2%，**认为**苹果树球形树冠相较于圆锥形树冠的一天超过八小时的受光面积更大。**

针对问题三，本文沿用了问题二中的**苹果树球形树冠超受光面积模型**。相较于问题二，问题三将太阳位置的**时间参量**由夏至日扩大到 5 月 5 日(立夏)到 10 月 8 日(寒露)，代入时间参量即可求得**立夏到寒露期间受光面积**的**变化趋势，**其**先单增**再**单减，结果详见图 5。**

针对问题四，本文构建了**苹果树树冠超受光面积梯度下降优化模型**。相较于问题三，问题四将**时间参量**扩大到全年。本文决定采用**梯度下降算法**根据时间进行重复**迭代**，以得出苹果树树冠的最优形状模型。为了简化模型，参照问题二中得出的结论，将树冠的初始模型建立为球形，先找到决定树冠形状的参数并将其设置为梯度下降算法的**梯度参数**，从而进行模型的**形状迭代**。再将问题二中算得的太阳轨迹代入此模型，仿照其模型进行**梯度更新**并更新球面每个点的位置，在迭代中树冠的形状与受光面积都不断增加，最终求得了一种最优的树冠形状，其大概是一个**北上往南下倾斜的椭球体，结果详见图 8**。

关键词： 太阳视运动； 网格化； 三维模拟； 光线追踪； 梯度下降

一、问题重述

1.1 问题背景

随着精准农业的发展，提高作物的生长效率与产量和优化自然资源的使用变得尤为重要。特别是对于光合作用极为依赖的果树种植，如何通过科学设计树冠形状以最大限度地捕捉阳光，是提高果树生长质量和效率的关键因素。

陕北地区由于其独特的地理和气候条件，成为苹果等干鲜果的优质生产区。尤其是洛川县，其地理位置和气候条件为苹果树提供了极佳的生长环境。在苹果树生长过程中，太阳光照的时间是影响苹果质量及生长与成熟的关键因素之一。传统的树冠形状多依赖经验设计，而缺乏科学的量化分析与优化，这在一定程度上限制了产量和质量的提升。因此，研究和优化苹果树树冠的设计来最大化树冠的受光效率，是提高苹果产量和质量的重要手段。

1.2 问题重述

本研究旨在通过数学建模和模拟方法探索苹果树树冠的最优形状设计，以增强其在不同季节下的日照接收能力。具体研究问题包括：

问题一：量化分析洛川县在夏至日白天每小时的太阳高度角，为后文建立模型基础。

问题二：运用特殊的三维建模方法求出并比较球形与圆锥形树冠在全天超八小时的受光面积，并比较分析球形树冠与圆锥形树冠哪一种受光面积更大，确定最适合提高光合作用效率的树冠形状。

问题三：分析球形树冠从立夏至寒露期间每日超受光面积的季节性变化，在这里，可以简单地沿用问题二的定义，认为超受光面积是指累计受光超过八小时的面积，从而可以探讨季节变化如何影响树冠的超受光面积，并分析出 5 月 5 日(立夏)到 10 月 8 日(寒露)球形树冠一天超受光面积的变化规律。

问题四：树冠现在的形状不能确定，目标是开发一种理想的树冠结构，以最大化一天中超过八小时的受光面积。

二、问题分析

本研究围绕苹果树的树冠设计展开，目标是通过数学建模和仿真分析，找到能最大化日照接收的树冠形状。以下是对各个子问题的详细分析：

2.1 问题一分析

在第一问中，我们需要确定洛川县夏至日每小时的太阳高度角。太阳高度角是决定光照强度和持续时间的重要因素，对树木的光合作用和生长至关重要。这一问题的解决依赖于天文算法，需要准确计算太阳的位置变化。通过天文学原理，我们可以得出太阳赤纬和时角的公式，进而求解太阳高度角。这一过程将涉及对洛川县特定纬度的日照模型的建立，为后续的树冠光照分析提供基础数据。

2.2 问题二分析

第二问要求比较不同树冠形状(球形与圆锥形)的受光效果。这需要建立详细的三维几何模型，并使用光线追踪技术来模拟光线如何在不同形状的树冠上被吸收。通过模拟，可以计算出不同形状树冠在一天中的累积受光面积，这对于理解哪种树冠形状有更长的被照射时间具有实际意义，为后续问题提供理论支撑。

2.3 问题三分析

第三问涉及到季节性变化对光照模式的影响，特别是从立夏到寒露期间球形树冠受光面积的变化规律。这需要对整个生长季节的日照进行动态模拟，分析不同季节太阳轨迹变化对光照面积的影响。该分析将帮助理解在不同生长阶段树冠如何响应自然光条件的变化，以调整农业管理和树冠修剪策略。

2.4 问题四分析

最后一个问题是设计一种新的树冠形状，以最大化其日照接收面积。这一问题的核心是使用优化技术，如梯度下降的算法，来调整树冠的参数，使得受光面积最大化。这涉及到复杂的参数化树冠模型和优化算法的开发，需要考虑到树冠的生物学特性和光学特性，确保模型的实用性和生物学的可行性。

通过这些分析，不仅能够解决具体的数学问题，还能为实际的农业生产提供理论指导和技术方案。

三、模型假设

1、忽略树冠等其他部位，只建立出树冠的模型。

2、假设树冠表面光滑、形状规则，各处对太阳光的吸收能力相同，且能完全吸收阳光。同时不考虑光的折射反射等情况。

3、忽略降雨等天气因素对树冠接收光的影响，太阳光线不被云层覆盖。

4、假设该地完全平坦，且忽略海拔对该模型的影响。

5、果树树冠均由边长为 2 的正方体修剪而成。

6、忽略黄道面及黄赤交角的微小变化。

四、符号说明

符号	说明	单位
ϕ	观测地点的纬度	度(°)
δ	太阳赤纬	度(°)
ω	太阳时角	度(°)
α	太阳高度角	度(°)
γ	太阳方位角	度(°)
T	一天中的时间	-
N	一年中的日序数	-
R_s	球形树冠的半径	米(m)
h_s	球形树冠的中心高度	米(m)
R_c	圆锥形树冠的底面半径	米(m)
h_c	圆锥形树冠的高度	米(m)
θ	球面坐标中的极角	度(°)
ϕ_s	球面坐标中的方位角	度(°)
$\vec{S}$	太阳的位置向量	-
$\vec{n}$	面元的法向量	-
A	面元的面积	平方米(m^2)
ϵ	用于计算数值梯度的微小增量	-
α_{lr}	学习率	-
$J(\theta)$	目标函数	-
η	受光效率	-

五、模型建立与求解

5.1 问题一模型的建立与求解

针对问题一，我们需要在特定日期（夏至）计算并给出洛川县每小时的太阳高度角。为此，我们建立一个基于天文学原理的模型，考虑地球的公转、自转和轴倾角对太阳高度角的影响。

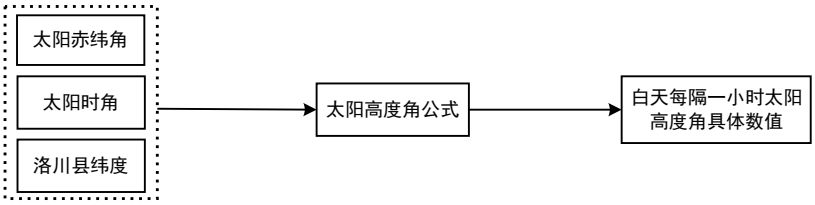


图 1 问题一模型的建立与求解的求解思路

5.1.1 模型的建立

太阳高度角是描述太阳光线与地平面之间角度的度量，对于给定的地理位置和时间，其计算公式依赖于太阳的赤纬 δ 和太阳时角 ω 。赤纬表示直射太阳光线与赤道平面之间的夹角，而太阳时角则表示给定地理位置的太阳相对于当地正午的角度差。

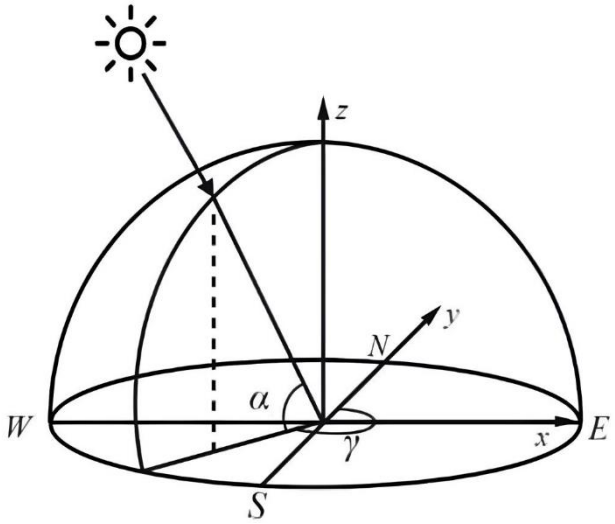


图 2 太阳高度角α示意图

①太阳赤纬角的计算：

太阳的赤纬角 δ 表征了给定时刻日地连线与地球赤道面之间的夹角[1]，可以使用下面的公式近似计算，该公式关联了一年中的某一天 N：

$$\delta = 23.45 \times \sin\left(\frac{360}{365} \times (N - 81)\right) \tag{1}$$

在这里， N 是年积日，对于夏至日(通常是 $N = 172$ ，且可以认为 1 月 1 日时 N=1)，可以将其代入上述公式计算得到赤纬角的值。

②太阳时角的计算：

太阳时角 ω 定义为从当地正午开始至特定时刻的角度变化，每小时改变 15 度[2]，可以按以下公式计算:

$$\omega = 15^\circ \times (T - 12) \tag{2}$$

这里 T 表示从 0 时开始计算的小时数。

③太阳高度角的计算:

结合赤纬角和太阳时角，太阳高度角 α 可以通过下面的方程计算[2]:

$$\sin\ \alpha = \sin\ \varphi \sin\ \delta + \cos\ \varphi \cos\ \delta \cos\ \omega \tag{3}$$

其中， φ 是观测地点的纬度(洛川县约为 35.77°N)， δ 是赤纬角， ω 是太阳时角。

④日出日落时间的计算

为了给出白天的太阳高度角，需要计算具体的日出日落时间。将式(2)(3)结合，根据太阳高度角为 0°时的反解出时角，再带入时角公式即可解出日出日落的时刻。需要注意的是，要确保正确地识别了日出和日落的零交叉点(从负到正表示日出，从正到负表示日落)。

5.1.2 模型的求解

使用 MATLAB 对上述模型进行数值求解，计算出夏至日从日出到日落每个整点的太阳高度角。具体实施步骤如下:

①使用第 1 部分的公式计算赤纬角 δ 。同时，为了简化计算，可以查阅资料获得夏至日太阳的赤纬角，在这里，可以认为夏至日的赤纬角$\delta_{solstice} = 23.44°$。

②对于每个小时 T ，使用第 2 部分的公式计算太阳时角 ω 。

③使用第 3 部分的公式计算对应的太阳高度角 α 。

④使用第 4 部分的公式计算对应日出日落时刻 T。

求解过程中，我们将从 0 时 24 时进行迭代计算，在正负高度角变化时用“interpolateTime”函数执行线性插值，确保准确计算日出日落时刻以及洛川县每小时的太阳高度角。结果如表 1。

表 1 白天间隔一小时的太阳高度角

T	α	T	α	T	α	T	α
4:47	0°	9:00	49.32°	14:00	61.34°	19:00	2.30°
5:00	2.23°	10:00	61.26°	15:00	49.40°	19:13	0°
6:00	13.39°	11:00	72.06°	16:00	37.24°		
7:00	25.11°	12:00	77.67°	17:00	25.19°		
8:00	37.16°	13:00	72.12°	18:00	13.46°		

此模型为洛川县在夏至日的光照模拟提供了精确的科学依据，对于研究苹果树的生长环境和改善农业生产具有重要意义。

5.2 问题二模型的建立与求解

为了计算苹果树的球形树冠一天超过八小时的受光面积，并比较苹果树的圆锥形树冠与球形树冠哪一种一天内超过八小时的受光面积更大。首先需要定义这一天，本文依据问题一，认为该天是夏至日。然后需要建立出一个笛卡尔坐标系，并在其中计算出太阳的周日时角运动轨迹；然后将锥体与球体的表面

进行了网格化离散处理[3]，其次对于光照时间的计算，本文借鉴了计算机中光线追踪的算法[4]。

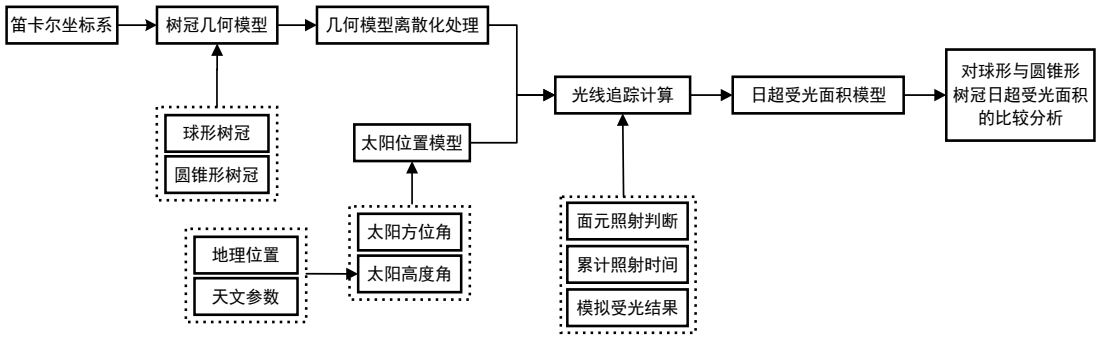


图 3 问题二模型的建立与求解的流程图

5.2 .1 模型的建立

为了准确比较苹果树的球形树冠和圆锥形树冠在接收日照方面的效率，我们采用了几何模型并借鉴了计算机处理物体的光线追踪技术。

①地理和天文参数:

地理位置：洛川县，纬度 $\varphi = 35.77°N$。

天文参数：使用太阳赤纬 δ 和时角 ω ，这些值从问题一继承。但为了提高面积计算的精度，在此将 T 精确到了每分钟，即$\Delta T = 1min$ 且 $T_{total} = 1440min$。

②笛卡尔坐标系建立:

坐标系统: 以正东方向为 X 轴正方向，正北方向为 Y 轴正方向，竖直向上为 Z 轴正方向，建立三维笛卡尔坐标系 $O - XYZ$ 。

③球形树冠和圆锥形树冠几何模型:

球形树冠：具有半径 R_s ，中心设在 $(0,0,R_s)$ 。

圆锥形树冠：底面半径 R_c ，高度 h_c ，顶点C设在 $(0,0,h_c)$ 。

④球体和圆雉的网格化处理:

Step1：球体网格化:

将球体表面依据经纬线划分为多个面元。每个面元由三个点定义，这些点的坐标根据球面坐标系给出:

$$\begin{aligned} X &= R_s \sin(\theta)\cos(\phi) \\ Y &= R_s \sin(\theta)\sin(\phi) \\ Z &= R_s \cos(\theta) \end{aligned} \tag{4}$$

其中， θ 为极角， ϕ 为方位角。

Step2：圆锥网格化:

将圆锥侧面划分为扇形区域 (三角形) 。底面上的点与顶点连接形成多个三角形。每个三角形的顶点坐标按照以下方式确定:

$$X_i = R_c \cos\left(\frac{2\pi i}{n}\right), Y_i = R_c \sin\left(\frac{2\pi i}{n}\right), Z = 0 \tag{5}$$

其中， i 是索引$(i = 1,2,3\cdots)$； n 是总的分割数量。

⑤太阳位置模型：

Step1：太阳高度角的计算

根据问题一，可以将公式中每小时的间隔周期步进为每分钟。从而可以得到一天内近似连续变化的太阳高度角。

Step2：太阳方位角的计算

根据赤纬 δ 和时角 ω ，太阳的方位 γ 可以计算为:

$$\cos\gamma = \frac{\sin\delta - \sin\alpha\sin\varphi}{\cos\alpha\cos\varphi} \tag{6}$$

其中，γ 为太阳方位角，其范围为$[0, 360°]$,δ为太阳赤纬，α为太阳高度角，φ 为当地纬度。

⑥法向量和面元面积计算：

Step1：球体:

每个面元的法向量 $\vec{n}$ 是从球心到该面元中心点的向量。面元面积 A 由球面坐标系下的微元面积计算得出:

$$A = R_s^2 \sin(\theta) d\theta d\phi \tag{7}$$

Step2：圆锥:

每个三角形面元的法向量 $\vec{n}$ 通过向量叉积计算，将圆锥的顶点坐标可以表示为:$A_i = (X_i, Y_i, Z_i)$；$B_{i+1} = (X_{i+1}, Y_{i+1}, Z_{i+1})$；$C = (0,0,h_c)$。则法向量可以通过计算两个向量 $\overrightarrow{AB}$ 和 $\overrightarrow{AC}$ 的叉积得到：

$$\begin{gathered} \overrightarrow{AB} = (X_{i+1} - X_i, Y_{i+1} - Y_i, Z_{i+1} - Z_i) \\ \overrightarrow{AC} = (-X_i, -Y_i, h_c - Z_i) \\ \vec{n} = \overrightarrow{AB} \times \overrightarrow{AC} = \begin{vmatrix} \mathbf{i} & \mathbf{j} & \mathbf{k} \\ X_{i+1} - X_i & Y_{i+1} - Y_i & Z_{i+1} - Z_i \\ -X_i & -Y_i & h_c - Z_i \end{vmatrix} \end{gathered} \tag{8}$$

最终得到的法向量 $\vec{n}$ 需要进行归一化处理，公式为:

$$\vec{n} = \frac{\vec{n}}{\| \vec{n} \|} \tag{9}$$

面积 A 使用向量的模长公式:

$$A = \frac{1}{2} |\overrightarrow{AB} \times \overrightarrow{AC}| \tag{10}$$

5.2.2 模型的求解

①光线追踪计算：

Step1：面元照射判断：

通过比较太阳光方向 $\vec{S}$ (由太阳的方位和高度角计算)

$$\vec{S} = \begin{bmatrix} \cos(\alpha)\cos(\gamma) \\ \cos(\alpha)\sin(\gamma) \\ \sin(\alpha) \end{bmatrix} \tag{11}$$

与各面元法向量 $\vec{n}$ 的点积确定是否被照射。

$$\text{Illuminated} = \begin{cases} \text{True}, & \text{if } \vec{n} \cdot \vec{S} > 0 \\ \text{False}, & \text{otherwise} \end{cases} \tag{12}$$

Step1：累积照射时间:

通过时间步进模拟一天内太阳的运动，记录每个面元被照射的时间。将所有满足被照射超过八小时的面元面积累加，以得到每种树冠的总有效受光面积。

初始化一个变量 A_{total} 来存储总面积，然后对每个面元执行以下操作:

$$\text{if IlluminatedTime } [i] \geq T \text{ then } A_{\text{total}} \mathrel{+}= A_i \tag{13}$$

其中 A_i 是面元 i 的面积。

Step2：模拟受光结果：

通过在 MATLAB 中运行代码。我们可以得到苹果树的球形与锥形树冠一天超过八小时的受光面积及模拟视图，如图 4 所示。相应的，苹果树的球形树冠一天超过八小时的受光面积为 11.38m^2，苹果树的锥形树冠一天超过八小时的受光面积为 3.58m^2。

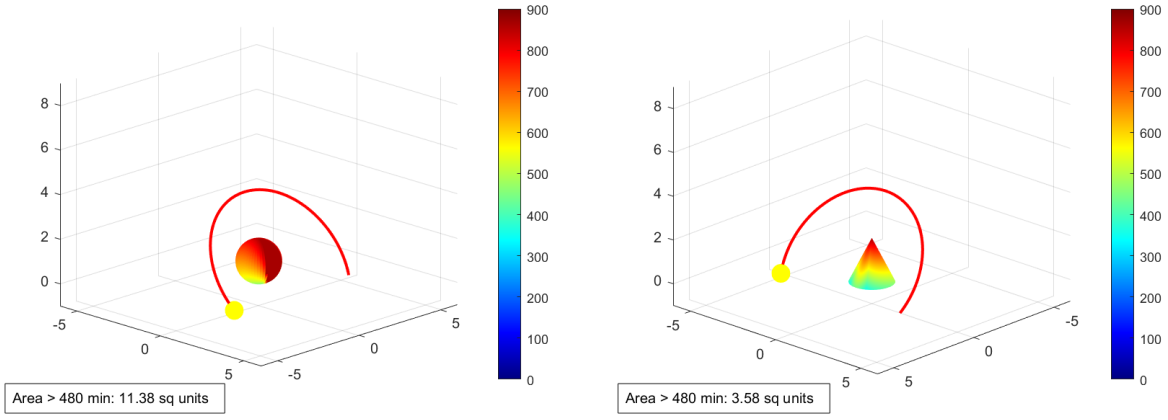


图 4 苹果树的球形与锥形树冠一天超过八小时的受光面积及模拟视图

②比较和分析:

对于给定的地理位置和日期范围，分别计算球形和圆锥形树冠的有效受光面积，前者的受光效率为 90.6%，后者的受光效率为 47.2%，受光效率 η 可用以下公式计算：

$$\eta = \frac{\text{受光超 8 小时面积}}{\text{总面积}} \times 100\% \tag{14}$$

通过这一详尽的模型分析，本文认为苹果树球形树冠相较于圆锥形树冠的一天超过八小时的受光面积更大。

5.3 问题三模型的建立与求解

对于问题三，需要建立一个太阳赤纬的全年变化模型，从而可以解出太阳在不同日期，不同赤纬对某物体的照射时长。

5.3.1 模型的建立

在问题三中，我们的目标是分析球形树冠在 5 月 5 日(立夏)到 10 月 8 日(寒露)期间每日的有效受光面积变化。这要求我们建立一个模型来详细计算每日超受光面积的季节性变化。在这里，本文借鉴了问题二中的模型，将超受光面积定义为一天内受光超八小时的面积。

①太阳赤纬的季节变化：

太阳赤纬 δ 的变化反映了地球公转带来的季节性变化，可以使用式(1)计算。

②日照时间计算：

太阳高度角 α 是关键因素，同样可以利用式(3)计算。

5.3.2 模型的求解

①累积照射时间计算：

基于 MATLAB 代码，我们采用时间步进法来模拟一天中的太阳运动，对每个离散化的面元累积其受光时间。

太阳光照的时间累积可以通过检查每个时间步中太阳光向量 $\vec{S}$ 与球面元的法向量 $\vec{n}$ 的点积是否大于零来实现。如果大于零，表示该面元在该时间步被照射。

$$\text{Illuminated } = \vec{n} \cdot \vec{S} > 0 \tag{15}$$

通过整日的模拟，我们记录下每个面元被照射的总时间。将所有满足被照射超过八小时的面元的面积累加，得到每天的有效受光面积。

$$A_{\text{day}} = \sum_{i \in \text{ Illuminated}} A_i \tag{16}$$

其中 A_i 是第 i 个面元的面积，且满足照射时间超过 480 分钟(8 小时)。

②分析季节变化：

通过迭代计算从立夏到寒露的每一天，使用上述方法计算每日的有效受光面积。结果如图 5 所示:

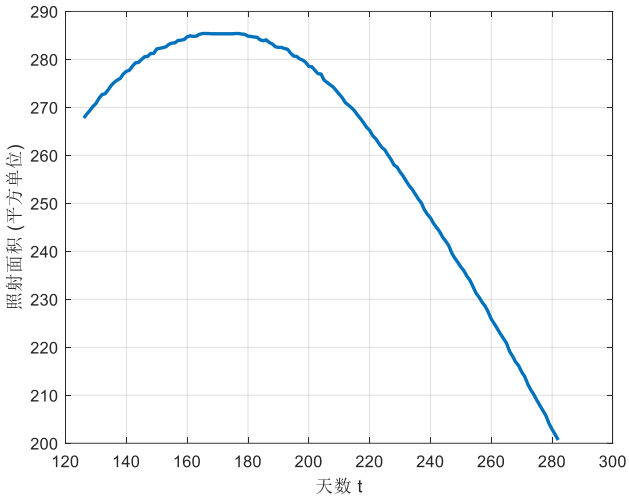


图 5 5 月 5 日（立夏）到 10 月 8 日（寒露）球形树冠一天超受光面积的变化规律

通过绘图展示，可以发现从立夏到寒露期间受光面积的变化趋势是在立夏到夏至不断增大，在夏至到寒露不断减小。通过图 5，可以直观地理解季节变化对苹果树生长的潜在影响。

5.4 问题四模型的建立与求解

为了求解问题四，我们沿用了问题二中的结论：相同照射条件下，球形树冠的超八小时受光面积比圆锥形树冠的超八小时受光面积更大。所以，针对问题四，本文进行优化算法的初始化时仅讨论球体。

5.4.1 模型的建立

问题四要求设计一种树冠形状，以使其在该地一天超过八小时的受光面积尽可能大。为了达到这一目标，我们采用梯度下降算法[5]优化树冠的形状。

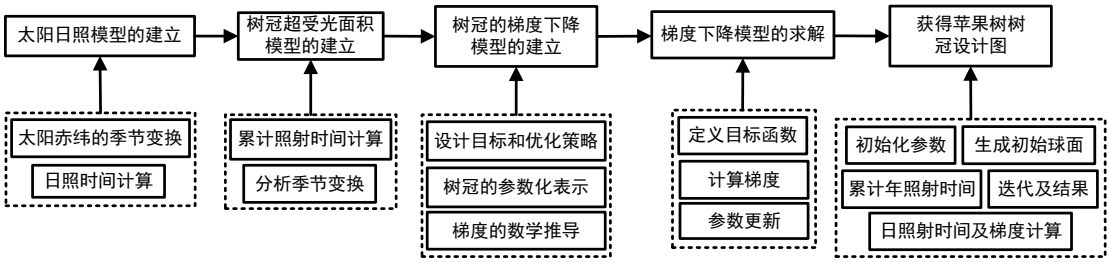


图 6 问题四模型的建立与求解的流程图

①设计目标和优化策略：

梯度下降算法是一种常用的优化方法，通过迭代调整参数以最小化(或最大化)目标函数。在本问题中，目标函数是一天中超过八小时受光面积的负值，我们的目标是最大化这个面积。

②树冠的参数化表示：

假设树冠可以被表示为一系列的参数(如顶点的位置、形状的曲率等)，这些参数将决定树冠的几何形状。通过调整这些参数，我们可以改变树冠的形状，并观察受光面积的变化。

③梯度计算：

梯度是目标函数相对于各参数的偏导数向量。在梯度下降步骤中，我们需要计算目标函数关于每个参数的梯度，然后用这些梯度来更新参数。在本案例中，梯度表示受光面积对树冠形状参数的敏感度。

5.4.2 模型的求解

①梯度的数学推导：

Step1：定义目标函数：

$$J(\theta) = -\text{ Total Illuminated Area }(\theta) \tag{17}$$

其中，θ 表示树冠形状的参数集，总受光面积是所有被照射超过八小时的面元的面积总和。

Step2：计算梯度:

梯度 $\nabla_\theta J(\theta)$ 是目标函数对于每个参数的偏导数。每个参数的偏导数可以通过以下方式近似计算:

$$\frac{\partial J}{\partial \theta_i} \approx \frac{J(\theta_i + \epsilon) - J(\theta_i)}{\epsilon} \tag{18}$$

其中 ϵ 是一个小的增量，用于计算数值梯度。

Step3：参数更新:

使用梯度和学习率 α 来更新参数:

$$\theta := \theta - \alpha \nabla_\theta J(\theta) \tag{19}$$

这一步是迭代进行的，直到达到收敛条件或完成指定的迭代次数。

②实现细节：

在 MATLAB 环境中，我们编写函数来模拟一天中的太阳轨迹和树冠的光照接收情况。对每个参数化的树冠形状，我们计算其受光面积，并使用上述梯度计算方法来迭代优化参数。

Step1：初始化参数:

在梯度下降算法中，有很多因素会影响迭代效果，这需要提前对它们进行定义，如表 2 所示。同时，要考虑树冠的生物学特性来设定迭代次数和学习率。

表 2 梯度下降的初始化参数

参数	含义
R	初始树冠半径
`num_lat, num_lon`	纬线和经线的数量, 用于离散化树冠表面
`daily_iterations`	每天进行的迭代次数
`days_in_year`	一年中的天数
`learning_rate`	学习率，控制梯度下降步长的参数

Step2：生成初始球面:

使用 MATLAB 的“sphere”函数生成一个标准球面，该球面随后将根据梯度下降算法进行调整。

Step3：累积年度照射时间:

定义一个数组 annual_illumination_time 来存储每个表面点在一年中的累积照射时间。

Step4：每天的照射时间及梯度计算:

太阳赤纬和方位角计算：根据当天的日期计算太阳赤纬和每分钟的太阳方位及高度角，具体的实现方法可参考 5.3 问题三模型的建立与求解。

照射判断和梯度更新：对于球面上的每一个点，计算其法向量与太阳向量的点积。如果点积大于零，表示该点在当前时刻被太阳照射。对于被照射的点，根据点积结果更新梯度。具体的实现方法可参考 5.2 问题二模型的建立与求解。

梯度的计算公式如下:

$$\begin{aligned} grad\_X(i,j) &+= sun\_vector(1) * dot\_product; \\ grad\_Y(i,j) &+= sun\_vector(2) * dot\_product; \\ grad\_Z(i,j) &+= sun\_vector(3) * dot\_product; \end{aligned} \tag{20}$$

这里的“dot_product”是法向量和太阳向量的点积结果。

Step5：每天的照射时间及梯度计算:

通过梯度下降算法，利用前一步计算得到的梯度和学习率更新球面的每个点:

$$\begin{aligned} X &= X + learning\_rate * gradients.X; \\ Y &= Y + learning\_rate * gradients.Y; \\ Z &= Z + learning\_rate * gradients.Z; \end{aligned} \tag{21}$$

此过程在一年中重复进行，每天根据梯度更新球面的形状，以最大化被照射时间。

Step6：优化过程和结果:

通过多次迭代，我们观察到树冠形状的逐步变化和受光面积的增加，如图 7 所示。每一步的参数更新都基于前一步的梯度计算，不断接近最优形状。

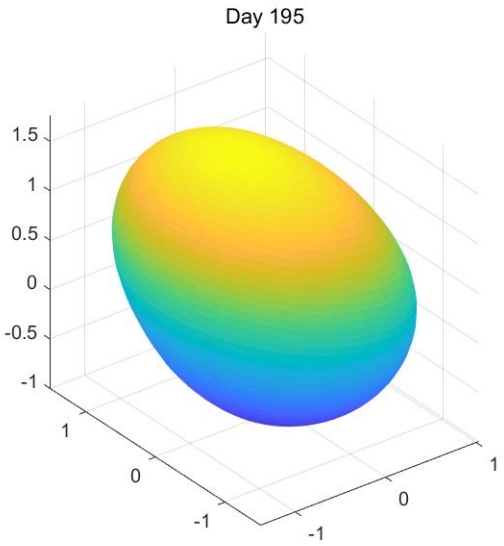
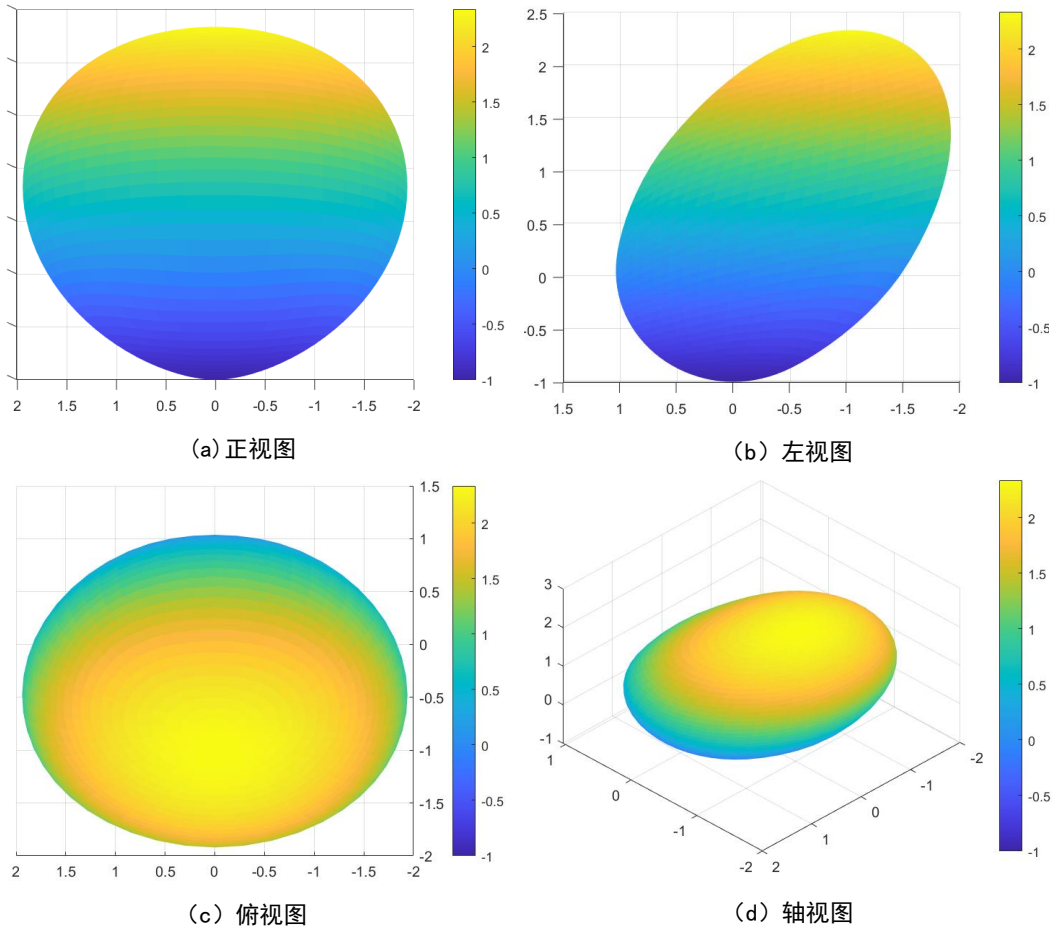


图 7 树冠的迭代过程

③苹果树树冠设计图

通过梯度下降算法，我们成功设计了一种最优的树冠形状，显著增加了树冠一天中受光时长超八小时的面积，如图 8 所示，其形状大概是一个北上往南下倾斜的椭球体。这种方法的实施为农业生产中树冠设计提供了一种有效的科学依据和技术路径。



(a) 正视图　　(b) 左视图

(c) 俯视图　　(d) 轴视图

图 8 树冠的三视图及轴视图

六、模型的检验

6.1 问题一模型的检验

针对问题一，本文考虑了夏至日的太阳赤纬、太阳时角，以及洛川县的纬度，最后给出了该地夏至白天每隔一小时的太阳高度角。但是从实际出发，地球的黄道面夹角(黄赤交角)会有微小变化，而且每年的夏至日期并不相同。于是，我们可以查询当地的实际太阳高度角来分析模型的有效性[6]。在图 9 中，本文拟合了近五年夏至日洛川县当地的实际太阳高度角数据，形成了误差带。模型一中算出的白天间隔一小时的太阳高度角的具体数值散点均落在误差带中。这可以说明问题一所建立的模型是有效的。

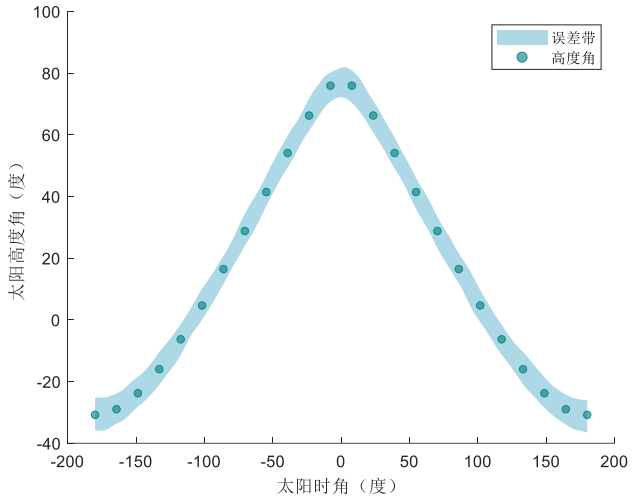


图 9 太阳高度角的误差评价

6.2 问题二模型的检验

对于问题二，不规则的太阳运动难以求出解析解。可以观察模型模拟太阳运动的光照时间来主观判断模型的精确性，如图 10 所示。

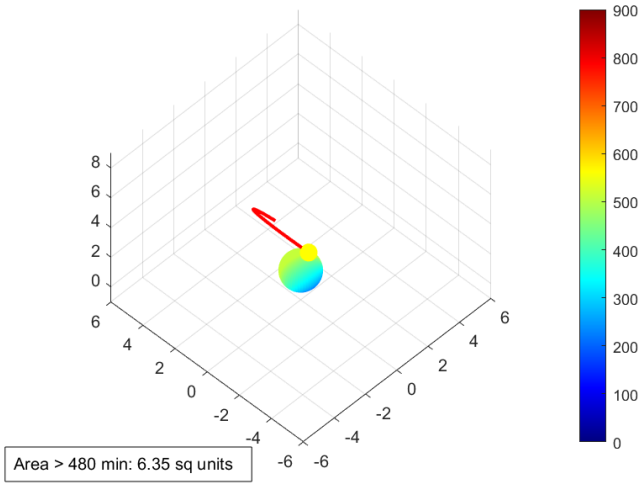


图 10 球体模型的实时模拟受光图

同时本文还采用了数值分析的方法检验模型的有效性：通过对比不同分辨率（离散化程度）下模型计算的受光面积，我们可以验证模型在不同离散化程度下的稳定性和收敛性。比如对于球体模型，将分割精度设为 2，那么一天中超八小时受光面积应该与图 11 上部的三角形面积的两倍相等，这是由于网格初始化时下部分会有微小的倾斜角度，导致终日不能受到太阳光照射。

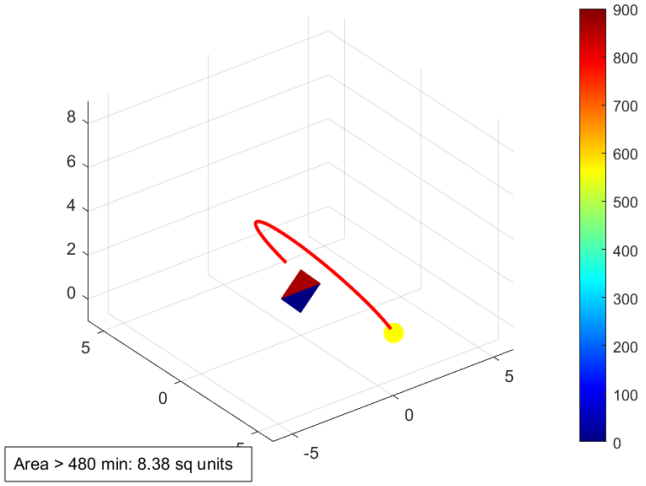


图 11 特定分割精度下的球体模拟受光图

通过观察法和数值分析法可以基本判断出该模型具有准确性和普适性。

6.3 问题三模型的检验

问题三可以采用敏感性分析的检验方法：调整模型中的参数进行敏感性分析，观察受光面积的变化情况，验证模型对不同参数的敏感度。首先我们可以在图 12 中观察到洛川县 5 月 5 日(立夏)到 10 月 8 日(寒露)超受光面积与太阳直射点纬度之间的关系。

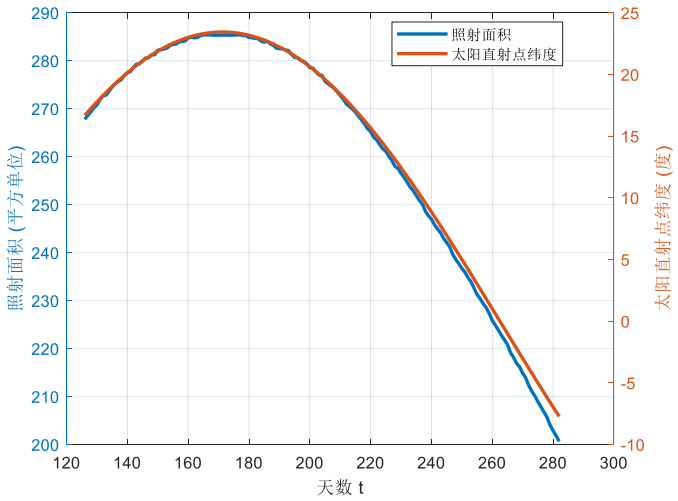


图 12 洛川县超受光面积与太阳直射点纬度之间的关系

同时，我们可以将模型三中的纬度数值调整为 35.77°S(洛川县的对趾点[1])，观察特殊情况的八小时以上受光面积的变化。

[1] 对趾点：地球表面上关于地心对称的位于地球直径两端的点。这两点的经度和为 180°，纬度数值相同，南北相反。

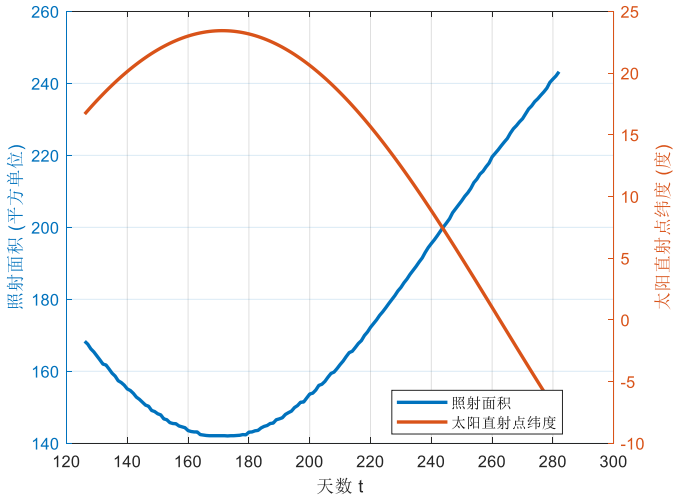
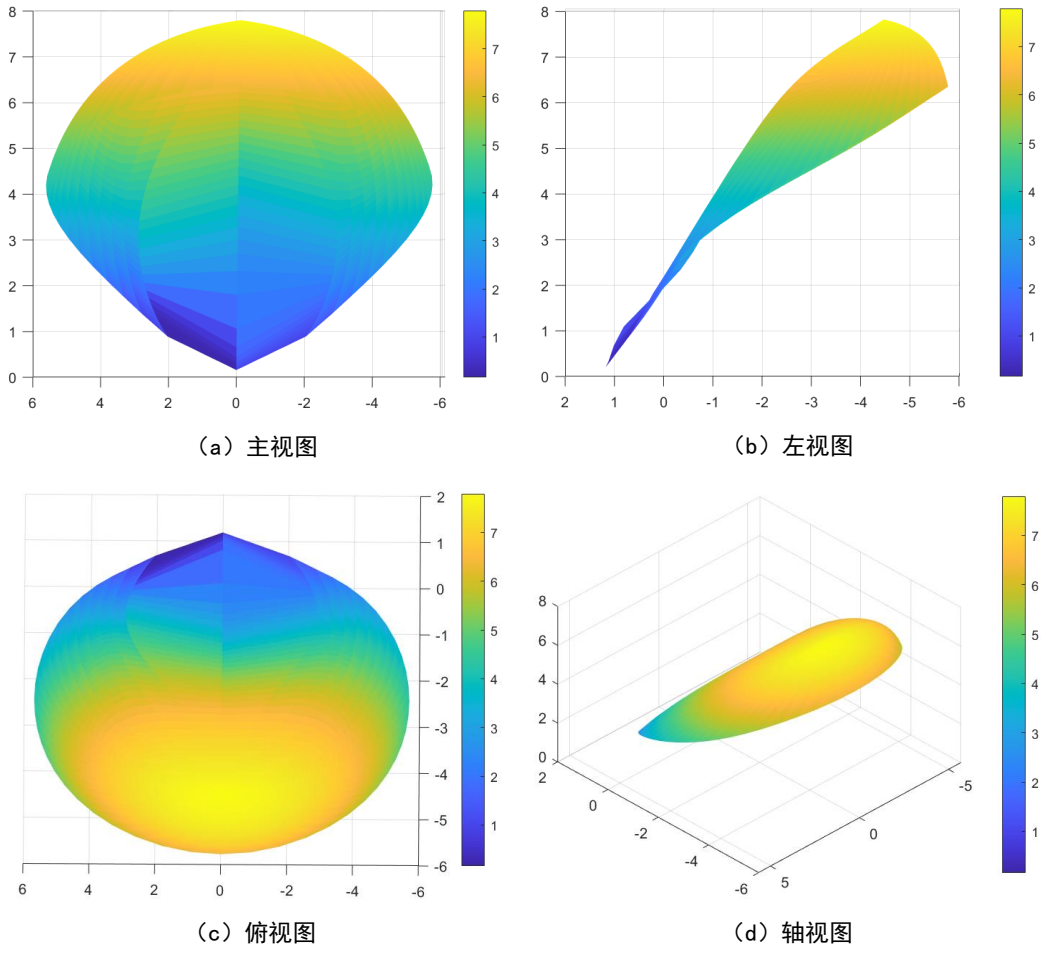


图 13 洛川县对趾点超受光面积与太阳直射点纬度之间的关系

通过季节性数据对比和敏感性分析，可以发现超受光面积与太阳赤纬和当地的纬度差成反比，如图 13。所以本文可以确认该模型在计算季节性变化方面的准确性和稳定性。

6.4 问题四模型的检验

问题四的目标是通过梯度下降算法优化树冠形状，使其在一天中受光面积超过八小时。我们可以通过修改初始三维体的方法进行多值验证，通过多组初始条件进行数值验证，观察最终优化结果是否一致，验证模型的鲁棒性和有效性。在这里，可以选用问题二中的网格圆锥进行梯度下降的迭代。



（a）主视图　　（b）左视图

（c）俯视图　　（d）轴视图

图 14 初始状态为圆锥体的梯度下降迭代结果图

将迭代次数进行合理的调整，在图 14 中可以明显看出与球体的迭代结果类似。通过多值验证，本文可以认为梯度下降算法在优化树冠形状方面具有强鲁棒性

七、模型的评价与推广

7.1 模型的评价

本文的模型展示了在特定条件下对苹果树树冠的优化设计，通过数学建模和仿真分析实现日照最大化，从而提高了光合作用效率和果实产量。以下是对模型进行评价的详细分析：

7.2.1 模型的优点

①实用性和应用范围广泛：

模型不仅适用于洛川县苹果树的优化，也可以扩展到其他需要日照优化的农作物。

②科学性强：

使用了天文学、几何建模和光线追踪等科学方法，保证了模型的精确性和可靠性。

③技术前沿：

引入了梯度下降算法来优化树冠形状，体现了模型在解决实际问题时的技术创新。

④具有验证机制：

模型包括了具体的验证步骤，如通过实际日照数据和模拟结果对比，确保了模型的有效性。同时，在 MATLAB 代码中的参数配置部分可以简易的调整参数，以便模拟不同日期，不同地点的受光结果

7.2.2 模型的缺点

①假设限制：

模型的准确性受到了初步假设的限制，如忽略了天气变化对光照的影响。

②复杂性较高：

对于非专业人员而言，模型中的高级算法和计算过程可能难以理解和应用。

③数据依赖性强：

模型的准确性高度依赖于输入数据的质量和精确性，数据的微小误差都可能影响最终结果。

④局部最优解问题：

梯度下降算法可能会导致模型陷入局部最优解，而不是全局最优解，尤其是在参数空间复杂或者存在多个最优解的情况下[7]。同时梯度下降的路径可能会进入吸引盆(attractor basins)，在这些区域内，算法会在不同的局部最优解之间徘徊，难以找到全局最优解，如图 15，从而影响模型优化的效率和结果的质量。

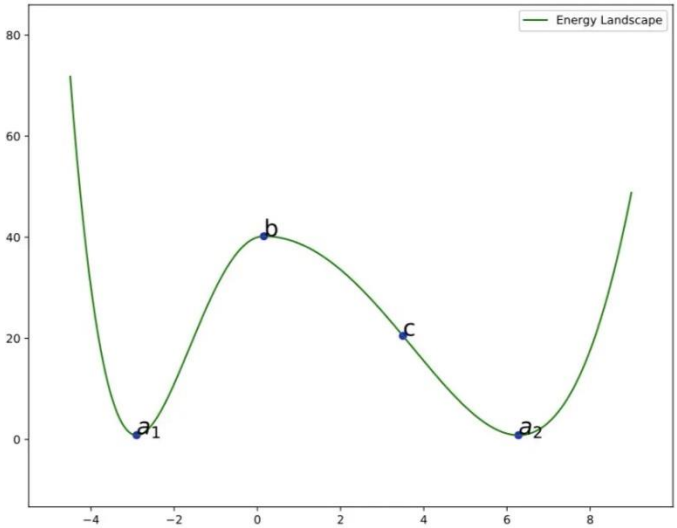


图 15 梯度下降算法的局部最优解示意图

7.2 模型的推广

①教育和培训：

通过教育培训，提高潜在用户，特别是农业技术人员对模型的理解和操作能力。

②软件开发：

开发用户友好的软件应用，使模型可以通过图形用户界面直观操作，降低技术门槛。

③政策支持：

争取政府或相关农业机构的支持，通过政策推广和资金扶持，将模型应用于实际农业生产中。

④跨学科合作：

与气象、农业科学等其他领域的专家合作，进一步提高模型的准确性和适用性。

⑤持续改进和更新：

根据用户反馈和技术进步，不断对模型进行优化和升级，保持其领先地位和实用价值。

八、参考文献

[1] 邓先武.太阳直射点周年变化图新解[J].教学月刊·中学版(教学参考),2015(Z1):107-109.

[2] 王慧,崔连延.太阳高度角和方位角的计算算法[J].电子技术与软件工程,2015,(17):167.

[3] 李学军,高永明,孙春娟.面向辐射度计算的快速网格化算法[J].指挥技术学院学报,2000(06):60-64.

[4] 徐汝彪,刘慎权.光线追踪技术综述[J].计算机研究与发展,1990(05):19-27.

[5] 李兴怡,岳洋.梯度下降算法研究综述[J].软件工程,2020,23(02):1-4.DOI:10.19644/j.cnki.issn2096-1472.2020.02.001.

[6] 太阳方位角和高度角计算器,http://www.atoolbox.net/Tool.php?Id=1060,2024.5

[7] 史加荣,王丹,尚凡华,等.随机梯度下降算法研究进展[J].自动化学报,2021,47(09):2103-2119.DOI:10.16383/j.aas.c190260.

九、附录

9.1 环境描述

运行软件及版本：MATLAB R2021b

操作系统及版本：Window 11 Version 23H2

9.2 程序源代码

9.2.1 问题一程序源代码

```
% 定义常数
latitude = 35.77;  % 纬度
declination = 23.44;  % 夏至的太阳赤纬

% 太阳时角的细分
hour_angle = linspace(-180, 180, 3600);  % 细分时角以提高精度

% 转换为弧度
lat_rad = deg2rad(latitude);
dec_rad = deg2rad(declination);
hour_angle_rad = deg2rad(hour_angle);

% 计算高度角
zenith_angle = acos(sin(lat_rad) * sin(dec_rad) + cos(lat_rad) *
cos(dec_rad) * cos(hour_angle_rad));
altitude_angle = pi/2 - zenith_angle;  % 高度角是天顶角的补角

% 寻找高度角从负到正的转变点（日出）和从正到负的转变点（日落）
zero_crossings = find(diff(sign(altitude_angle)) ~= 0);
sunrise_idx = [];
sunset_idx = [];

% 检查每个交叉点确保正确分类日出和日落
for i = 1:length(zero_crossings)
    if altitude_angle(zero_crossings(i)) < 0 &&
altitude_angle(zero_crossings(i) + 1) > 0
        sunrise_idx = [sunrise_idx, zero_crossings(i)];
    elseif altitude_angle(zero_crossings(i)) > 0 &&
altitude_angle(zero_crossings(i) + 1) < 0
        sunset_idx = [sunset_idx, zero_crossings(i)];
    end
end

% 如果存在日出和日落的索引，则使用线性插值计算时间
if ~isempty(sunrise_idx) && ~isempty(sunset_idx)
    sunrise_time = interpolateTime(hour_angle, altitude_angle,
sunrise_idx(1));
```

```
    sunset_time = interpolateTime(hour_angle, altitude_angle,
sunset_idx(1));
else
    sunrise_time = NaN;
    sunset_time = NaN;
end

% 输出结果
fprintf('日落时间：%.2f 时\n', sunrise_time-12);
fprintf('日出时间：%.2f 时\n', sunset_time+12);
fprintf('时间（小时）- 太阳高度角（度）：\n');

% 每个整点的高度角
for k = -12:12  % 从-12 到 12 小时，对应于从-180 到 180 度的时角
    idx = find(abs(hour_angle - k*15) < 0.1, 1);  % 找到最接近整点的时角
    if ~isempty(idx)
        fprintf('%d 时: %.2f 度\n', k+12,
rad2deg(altitude_angle(idx)));  % 输出整点的太阳高度角
    end
end

function time = interpolateTime(hour_angle, altitude_angle, index)
    % 使用线性插值计算具体时间点
    ha1 = hour_angle(index);
    ha2 = hour_angle(index + 1);
    alt1 = altitude_angle(index);
    alt2 = altitude_angle(index + 1);
    % 插值公式，找到高度角为 0 的时角
    ha_zero = ha1 + (0 - alt1) * (ha2 - ha1) / (alt2 - alt1);
    % 转换时角为时间
    time = mod((ha_zero / 15 + 24), 24);  % 确保时间在 24 小时内
end
```

9.2.2 问题二程序源代码

①圆锥型

```
%代码等待时长约为 20 秒（等待日出）


%%代码完整运行时长约为 80 秒

function main
    % 圆锥参数
    R = 1; % 圆锥底面半径
    H = 2; % 圆锥高度
```

```
    num_lon = 500; % 分割数量（类似经度线的数量）

    % 太阳参数
    sun_distance = 4; % 太阳到圆锥顶点的距离
    sun_radius = 0.4; % 太阳半径

    % 太阳位置计算参数
    latitude = 35.77; % 纬度
    declination = 23.44; % 夏至的太阳赤纬
    minutes_in_day = 24 * 60; % 一天的分钟数
    lat_rad = deg2rad(latitude);
    dec_rad = deg2rad(declination);

    % 创建圆锥网格
    [X, Y, Z] = cylinder([0 R], num_lon); % 使用半径为 0 和 R 的圆柱来形成圆
锥
    Z = Z * (-H); % 调整高度
    Z = Z - min(Z(:)); % 调整 Z 坐标使底面位于原点

    % 创建太阳小球网格
    [sunX, sunY, sunZ] = sphere(20);
    sunX = sun_radius * sunX;
    sunY = sun_radius * sunY;
    sunZ = sun_radius * sunZ;

    % 初始化被照射时间数组
    illuminated_time = zeros(size(X));

    % 创建图形并设置视图
    figure;
    hCone = surf(X, Y, Z, 'FaceColor', 'interp', 'EdgeColor',
'none'); % 圆锥
    hold on;
    hSun = surf(sunX, sunY, sunZ, 'FaceColor', 'y', 'EdgeColor',
'none'); % 太阳小球
    hPath = plot3(0, 0, 0, 'r-', 'LineWidth', 2); % 初始化太阳路径
    axis equal;
    view(3);
    xlim([-sun_distance, sun_distance]*1.5);
    ylim([-sun_distance, sun_distance]*1.5);
    zlim([-1, sun_distance+5]);
    caxis([0 900]); % 设置颜色范围至 900 分钟
    colormap(jet(900)); % 使用从蓝到红的色彩映射
    colorbar; % 添加颜色条
```

```
    % 实时面积显示框
    area_box = annotation('textbox', [.02 .05 .1 .05], 'String',
'Initializing', 'FitBoxToText', 'on', 'BackgroundColor', 'white');

    % 太阳位置集合
    sunPositions = [];

    % 计算每分钟的太阳位置
    for minute = 1:minutes_in_day
        hour_angle = (minute - minutes_in_day / 2) * 360 /
minutes_in_day;
        hour_angle_rad = deg2rad(hour_angle);

        % 计算太阳的方位和高度角
        zenith_distance = acos(sin(lat_rad) * sin(dec_rad) +
cos(lat_rad) * cos(dec_rad) * cos(hour_angle_rad));
        altitude_angle = pi/2 - zenith_distance;

        % 只处理太阳高度角大于 0 的情况
        if altitude_angle > 0
            B = (sin(dec_rad) - sin(lat_rad) * sin(altitude_angle)) ./
(cos(lat_rad) * cos(altitude_angle));
            B = max(min(B, 1), -1);
            if hour_angle <= 0
                azimuth_angle = acos(B);
            else
                azimuth_angle = 2 * pi - acos(B);
            end
            sun_vector = [cos(azimuth_angle) * cos(altitude_angle),
sin(azimuth_angle) * cos(altitude_angle), sin(altitude_angle)];
            sunPos = sun_distance * sun_vector;

            % 更新太阳位置
            set(hSun, 'XData', sunX + sunPos(1), 'YData', sunY +
sunPos(2), 'ZData', sunZ + sunPos(3));

            % 记录太阳位置
            sunPositions = [sunPositions; sunPos];

            % 更新太阳路径
            set(hPath, 'XData', sunPositions(:,1), 'YData',
sunPositions(:,2), 'ZData', sunPositions(:,3));
```

```
            % 检查每个圆锥网格点是否被照射
            for i = 1:size(X, 1)
                for j = 1:size(X, 2)
                    normal = [X(i,j), Y(i,j), Z(i,j)];
                    normal = normal / norm(normal); % 归一化
                    if dot(normal, sun_vector) > 0
                        illuminated_time(i, j) = illuminated_time(i, j) +
1;
                    end
                end
            end
        end

        % 更新图形数据
        set(hCone, 'CData', illuminated_time);
        drawnow;
        pause(0.05); % 暂停一段时间以减慢动画速度

        % 计算并更新显示超过 480 分钟的面积
        area_over_480 = sum(illuminated_time(:) > 480) * 2 * pi * R^2
/ numel(illuminated_time); % 面积计算假定均匀分布
        set(area_box, 'String', sprintf('Area > 480 min: %.2f sq
units', area_over_480));
    end
end
```

②球型

```
%%代码等待时长约为 30 秒（等待日出）

%%代码完整运行时长约为 120 秒

function main
    % 球体参数
    R = 1; % 球体半径
    num_lat = 50; % 纬线的数量
    num_lon = 50; % 经线的数量
    sun_distance = 4; % 太阳到球体中心的距离
    sun_radius = 0.4; % 太阳的半径

    % 太阳位置计算参数
    latitude = 35.77;  % 纬度
    declination = 23.44;  % 夏至的太阳赤纬
    minutes_in_day = 24 * 60; % 一天的分钟数
    lat_rad = deg2rad(latitude);
```

```
    dec_rad = deg2rad(declination);

    % 创建球体网格
    [X, Y, Z] = sphere(num_lat);
    X = R * X;
    Y = R * Y;
    Z = R * Z + R;  % 球心位于 (0,0,R)

    % 创建太阳小球网格
    [sunX, sunY, sunZ] = sphere(20);
    sunX = sun_radius * sunX;
    sunY = sun_radius * sunY;
    sunZ = sun_radius * sunZ;

    % 初始化被照射时间数组
    illuminated_time = zeros(num_lat+1, num_lon+1);

    % 创建图形并设置初步视图
    figure;
    h = surf(X, Y, Z, zeros(size(X)), 'EdgeColor', 'none'); % 被照射球
体
    hold on;
    hSun = surf(sunX, sunY, sunZ, 'FaceColor', 'y', 'EdgeColor',
'none'); % 太阳小球
    hPath = plot3(0, 0, 0, 'r-', 'LineWidth', 2); % 初始化太阳路径
    axis equal;
    view(3); % 设置 3D 视图
    xlim([-sun_distance, sun_distance]*1.5);
    ylim([-sun_distance, sun_distance]*1.5);
    zlim([-1, sun_distance+5]);
    caxis([0 900]); % 设置颜色范围至 900 分钟
    colormap(jet(900)); % 使用从蓝到红的色彩映射
    colorbar; % 添加颜色条以解释颜色映射

    % 实时面积显示框
    annotation('textbox', [.02 .05 .1 .05], 'String', 'Initializing',
'FitBoxToText', 'on', 'BackgroundColor', 'white');

    % 太阳位置集合
    sunPositions = [];

    % 计算每分钟的太阳位置
    for minute = 1:minutes_in_day
```

```
        hour_angle = (minute - minutes_in_day / 2) * 360 /
minutes_in_day;
        hour_angle_rad = deg2rad(hour_angle);

        % 计算太阳的方位和高度角
        zenith_distance = acos(sin(lat_rad) * sin(dec_rad) +
cos(lat_rad) * cos(dec_rad) * cos(hour_angle_rad));
        altitude_angle = pi/2 - zenith_distance;

        % 只处理太阳高度角大于 0 的情况
        if altitude_angle > 0
            B = (sin(dec_rad) - sin(lat_rad) * sin(altitude_angle)) ./
(cos(lat_rad) * cos(altitude_angle));
            B = max(min(B, 1), -1);
            if hour_angle <= 0
                azimuth_angle = acos(B);
            else
                azimuth_angle = 2 * pi - acos(B);
            end
            sun_vector = [cos(azimuth_angle) * cos(altitude_angle),
sin(azimuth_angle) * cos(altitude_angle), sin(altitude_angle)];
            sunPos = sun_distance * sun_vector;

            % 更新太阳位置
            set(hSun, 'XData', sunX + sunPos(1), 'YData', sunY +
sunPos(2), 'ZData', sunZ + sunPos(3));

            % 记录太阳位置
            sunPositions = [sunPositions; sunPos];

            % 更新太阳路径
            set(hPath, 'XData', sunPositions(:,1), 'YData',
sunPositions(:,2), 'ZData', sunPositions(:,3));

            % 检查每个球面网格点是否被照射
            for i = 1:num_lat+1
                for j = 1:num_lon+1
                    normal = [X(i,j), Y(i,j), Z(i,j)];
                    normal = normal / norm(normal); % 归一化
                    if dot(normal, sun_vector) > 0
                        illuminated_time(i, j) = illuminated_time(i, j) +
1;

                    end
                end
```

```
            end
        end

        % 更新图形数据
        set(h, 'CData', illuminated_time);
        drawnow;
        pause(0.01); % 暂停一段时间以减慢动画速度

        % 计算并更新显示超过 480 分钟的面积
        area_over_480 = sum(illuminated_time(:) > 480) * 4 * pi * R^2
/ numel(illuminated_time);
        annotation('textbox', [.02 .05 .1 .05], 'String',
sprintf('Area > 480 min: %.2f sq units', area_over_480),
'FitBoxToText', 'on', 'BackgroundColor', 'white');
    end
end
```

9.2.3 问题三程序源代码

```
%%代码完整运行时长数分钟

function main
    % 球体参数
    R = 5; % 球体半径
    num_lat = 100; % 纬线的数量
    num_lon = 100; % 经线的数量

    % 时间范围
    t_start = 126;
    t_end = 282;

    % 初始化数组以存储结果
    illuminated_areas = zeros(t_end - t_start + 1, 1);
    days = t_start:t_end;

    % 循环计算每一天的面积
    for t = t_start:t_end
        % 计算太阳赤纬
        declination = 23.44 * sin(deg2rad(360/365 * (t - 80)));
        latitude = 35.77;  % 纬度
        minutes_in_day = 24 * 60; % 一天的分钟数
        lat_rad = deg2rad(latitude);
        dec_rad = deg2rad(declination);

        % 创建球体网格
```

```
        [X, Y, Z] = sphere(num_lat);
        X = R * X;
        Y = R * Y;
        Z = R * Z + R;  % 球心位于 (0,0,R)

        % 初始化被照射时间数组
        illuminated_time = zeros(num_lat+1, num_lon+1);

        % 计算每分钟的太阳位置
        for minute = 1:minutes_in_day
            hour_angle = (minute - minutes_in_day / 2) * 360 /
minutes_in_day;
            hour_angle_rad = deg2rad(hour_angle);

            % 计算太阳的方位和高度角
            zenith_distance = acos(sin(lat_rad) * sin(dec_rad) +
cos(lat_rad) * cos(dec_rad) * cos(hour_angle_rad));
            altitude_angle = pi/2 - zenith_distance;

            % 只处理太阳高度角大于 0 的情况
            if altitude_angle > 0
                B = (sin(dec_rad) - sin(lat_rad) *
sin(altitude_angle)) ./ (cos(lat_rad) * cos(altitude_angle));
                B = max(min(B, 1), -1);
                if hour_angle <= 0
                    azimuth_angle = acos(B);
                else
                    azimuth_angle = 2 * pi - acos(B);
                end
                sun_vector = [cos(azimuth_angle) * cos(altitude_angle),
sin(azimuth_angle) * cos(altitude_angle), sin(altitude_angle)];

                % 检查每个球面网格点是否被照射
                for i = 1:num_lat+1
                    for j = 1:num_lon+1
                        normal = [X(i,j), Y(i,j), Z(i,j)];
                        normal = normal / norm(normal); % 归一化
                        if dot(normal, sun_vector) > 0
                            illuminated_time(i, j) = illuminated_time(i,
j) + 1;
                        end
                    end
                end
            end
```

```
        end

        % 计算被照射超过 480 分钟的面积
        area_per_patch = 4 * pi * R^2 / ((num_lat + 1) * (num_lon +
1));
        total_illuminated_area = sum(illuminated_time(:) >= 480) *
area_per_patch;
        illuminated_areas(t - t_start + 1) = total_illuminated_area;
    end

    % 绘制结果
    figure;
    plot(days, illuminated_areas);
    xlabel('天数 t');
    ylabel('照射面积 (平方单位)');
    grid on;
end
```

9.2.4 问题四程序源代码

```
%%代码完整运行时长数分钟

function optimize_shape_over_year
    % 初始化参数
    R = 1;
    num_lat = 50;
    num_lon = 50;
    daily_iterations = 2;
    days_in_year = 365;
    learning_rate = 0.01;

    % 生成初始球面
    [X, Y, Z] = sphere(num_lat);

    % 初始化年度照射时间累计数组
    annual_illumination_time = zeros(size(X));

    % 循环整个年度
    figure;
    for day = 1:days_in_year
        % 每天的形状调整
        for iter = 1:daily_iterations
            % 计算当前形状的照射时间及梯度
```

```
            [daily_illumination_time, gradients] =
compute_daily_illumination_and_gradients(X, Y, Z, day, num_lat,
num_lon);

            % 累积年度照射时间
            annual_illumination_time = annual_illumination_time +
daily_illumination_time;

            % 更新球面形状
            X = X + learning_rate * gradients.X;
            Y = Y + learning_rate * gradients.Y;
            Z = Z + learning_rate * gradients.Z;
        end

        % 更新图形以显示每天的形状变化
        clf;
        surf(X, Y, Z, 'EdgeColor', 'none');
        axis equal;
        title(sprintf('Day %d', day));
        pause(0.00001);  % 暂停以便观察每日变化
    end

    % 显示最终形状
    surf(X, Y, Z, 'EdgeColor', 'none');
    colorbar;
    title('Final Shape After One Year');
end

function [daily_illumination_time, gradients] =
compute_daily_illumination_and_gradients(X, Y, Z, day, num_lat,
num_lon)
    % 定义天数和纬度
    latitude = 35.77;  % 纬度
    declination = 23.44 * sin(deg2rad(360/365 * (day - 80))); % 太阳赤
纬
    dec_rad = deg2rad(declination);
    lat_rad = deg2rad(latitude);

    % 初始化照射时间和梯度矩阵
    daily_illumination_time = zeros(num_lat+1, num_lon+1);
    grad_X = zeros(num_lat+1, num_lon+1);
    grad_Y = zeros(num_lat+1, num_lon+1);
    grad_Z = zeros(num_lat+1, num_lon+1);
```

```
    minutes_in_day = 1440; % 一天的分钟数

    % 循环计算每分钟的太阳位置
    for minute = 1:minutes_in_day
        hour_angle = (minute - minutes_in_day / 2) * 360 /
minutes_in_day;
        hour_angle_rad = deg2rad(hour_angle);

        % 计算太阳的方位和高度角
        zenith_distance = acos(sin(lat_rad) * sin(dec_rad) +
cos(lat_rad) * cos(dec_rad) * cos(hour_angle_rad));
        altitude_angle = pi/2 - zenith_distance;

        % 如果太阳高于地平线，计算太阳向量
        if altitude_angle > 0
            B = (sin(dec_rad) - sin(lat_rad) ...
                * sin(altitude_angle)) ./ (cos(lat_rad) *
cos(altitude_angle));
            B = max(min(B, 1), -1);
            if hour_angle <= 0
                azimuth_angle = acos(B);
            else
                azimuth_angle = 2 * pi - acos(B);
            end
            sun_vector = [cos(azimuth_angle) * cos(altitude_angle),
sin(azimuth_angle) * cos(altitude_angle), sin(altitude_angle)];

            % 检查每个点是否被照射
            for i = 1:num_lat+1
                for j = 1:num_lon+1
                    normal = [X(i,j), Y(i,j), Z(i,j)];
                    normal = normal / norm(normal); % 归一化
                    if dot(normal, sun_vector) > 0
                        daily_illumination_time(i,j) =
daily_illumination_time(i,j) + 1;

                        % 梯度计算
                        dot_product = dot(normal, sun_vector);
                        grad_X(i,j) = grad_X(i,j) + sun_vector(1) *
dot_product;
                        grad_Y(i,j) = grad_Y(i,j) + sun_vector(2) *
dot_product;
                        grad_Z(i,j) = grad_Z(i,j) + sun_vector(3) *
dot_product;
```

```
                end
            end
        end
    end
end

    % 归一化梯度
    gradients.X = grad_X / minutes_in_day;
    gradients.Y = grad_Y / minutes_in_day;
    gradients.Z = grad_Z / minutes_in_day;
end
```