为了实现wirtz pump的模拟，我们首先做了如下假设：

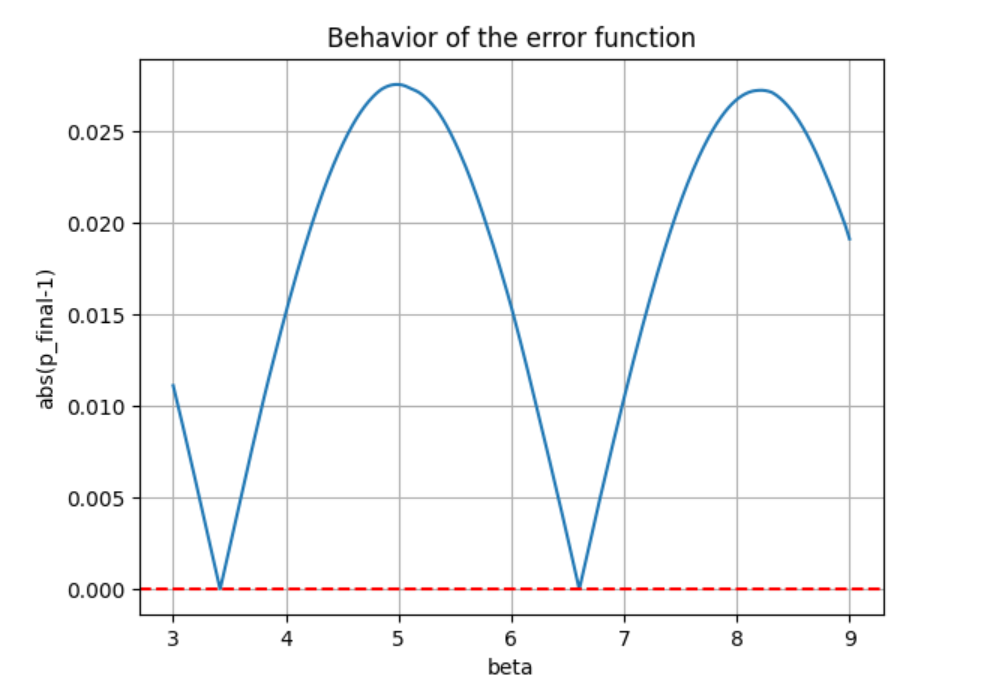
1. 水泵系统保持空间位置固定
2. 转轴维持恒定角速度旋转
3. 排除介质混合效应（空气柱间无渗混，水柱间无融合）

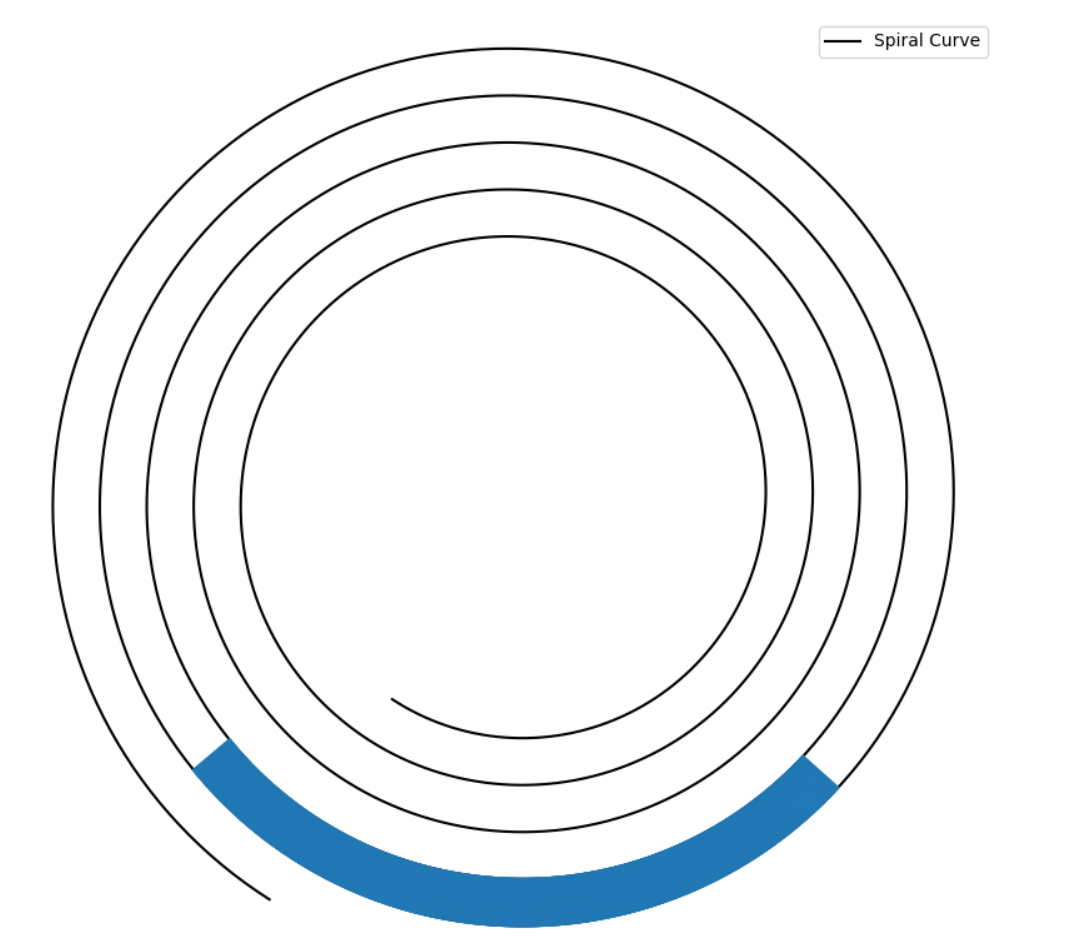
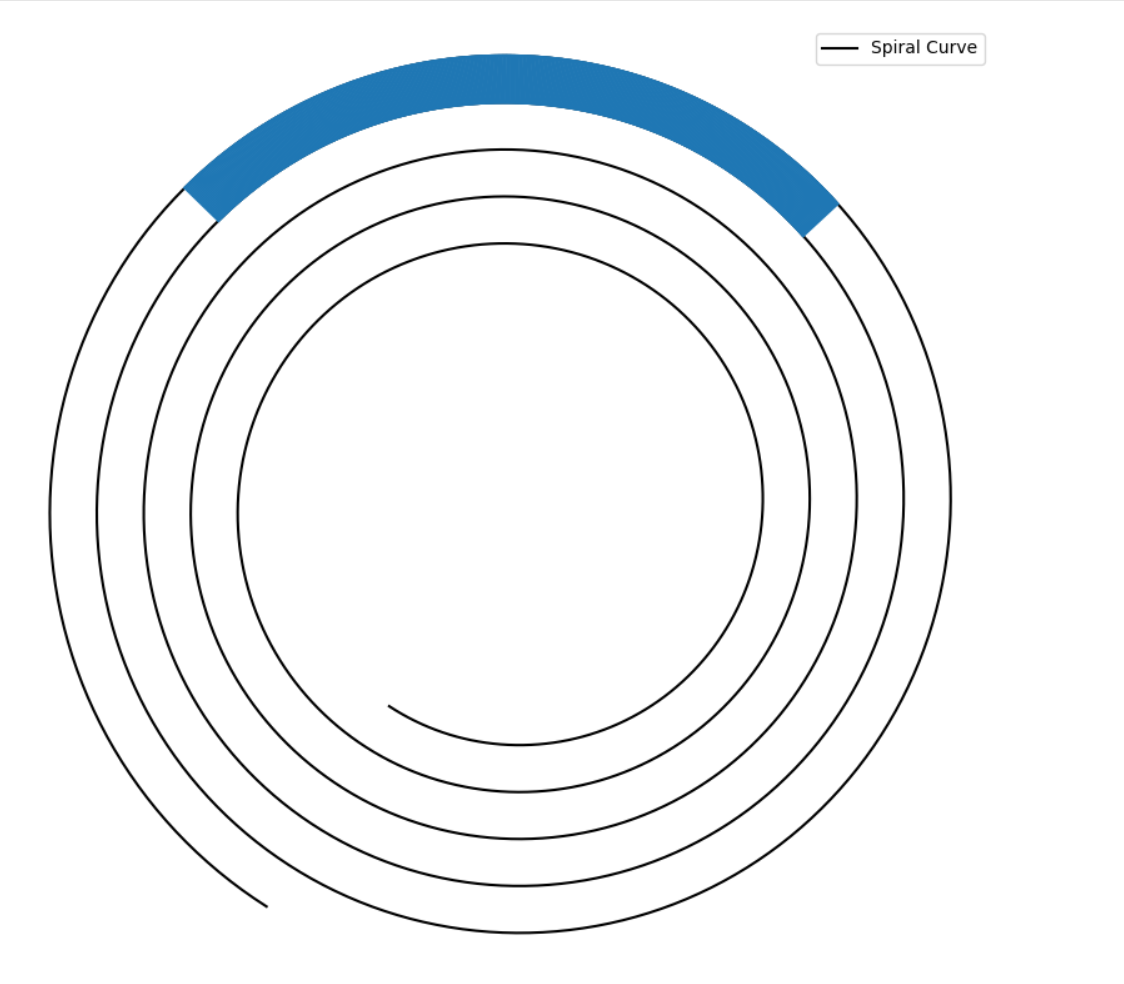
代码的实现逻辑如下：

|  |
| --- |
| 伪代码 |
| total\_cycles: 水柱总数量  dictor1: 存储每一段空气柱的PV值  dictor2: 存储每一段水柱的长度  *alpha*: 每一段水柱近端口的角度  *beta*: 每一段水柱远端口的角度 |
| **FOR** cycle **IN** total\_cycles:  **IF** current\_time == port\_entering\_water:  dictor1.store(p\_air \* V\_air)    **IF** current\_time == port\_exiting\_water:  dictor2.store(water\_column\_length)    **IF** port\_submerged:  *alpha* = 0  **ELSE**:  *alpha* = calculate\_alpha(dictor2, beta)    **FOR** water\_column **IN** 1 **TO** num\_water\_columns:  *alpha* = fetch\_from\_dictor1()  *beta* = compute\_beta(*alpha*, dictor2)  p\_current = update\_pressure(*alpha*, *beta*, p\_previous)    # Rotation termination condition  **IF** current\_angle >= threshold\_angle:  BREAK    **RETURN** abs(p\_current - 1) |

利用上述error function和python的minimize\_scalar， 我们容易计算出整个系统在每一时刻的状态。

当我们绘制这个结果函数的图像时，会发现存在多个解使得函数值达到预期，也就是所谓的多解性现象。如 下图所展示的情况，其中有一个解对应着水几乎倒置的状态，虽然从数学上严格推导，这个解满足我们所建立的基本方程，但在实际物理场景中，这种情况是无法实现的。原因在于，要使水达到这种倒置状态，系统首先要经过一段水柱几乎竖直的状态，而两段空气柱的压强差显然无法支持这么高的水柱；此外，这样的状态是高度不稳定的，很小的扰动就会破坏之。



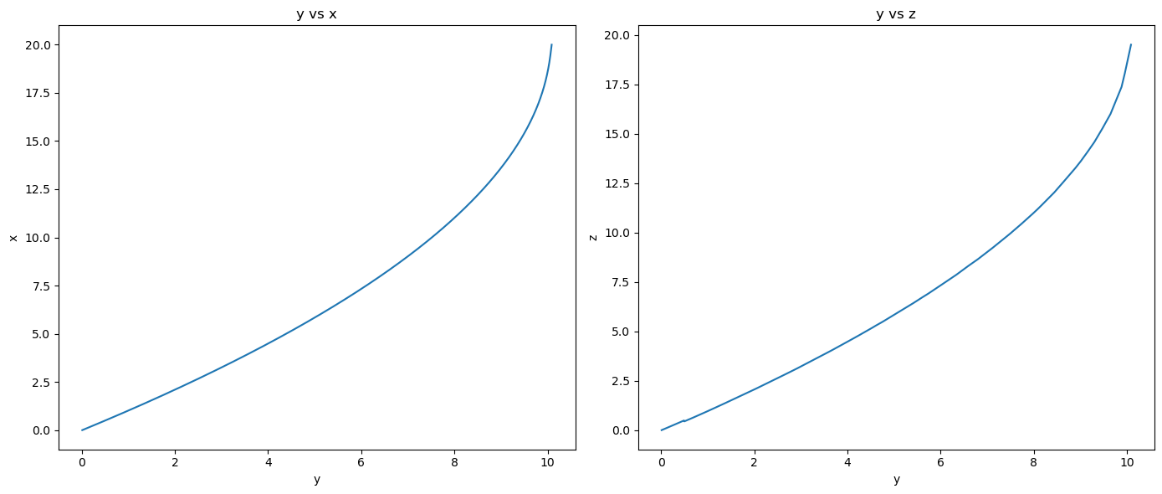
面对多解性带来的困扰，一种较为直接的解决方法是模拟每一个时刻的状态，即对整个过程进行细致入微的瞬态模拟。这种方法能够精确地追踪系统在每一个时间点的演变，由于两个时刻之间角度不会发生跃变，所以我们能够通过上一时刻的角度来确定下一时刻的解，从而确保所得到的解是符合物理实际的。然而，这种严格的瞬态模拟往往需要耗费大量的计算资源和时间，因此我们提出了一个更加便捷的计算方法——二分法。具体而言，就是通过不断二分来逼近多解出现的瞬间状态，这样既能够节省计算成本，也确保了求解过程的严谨性和准确性。

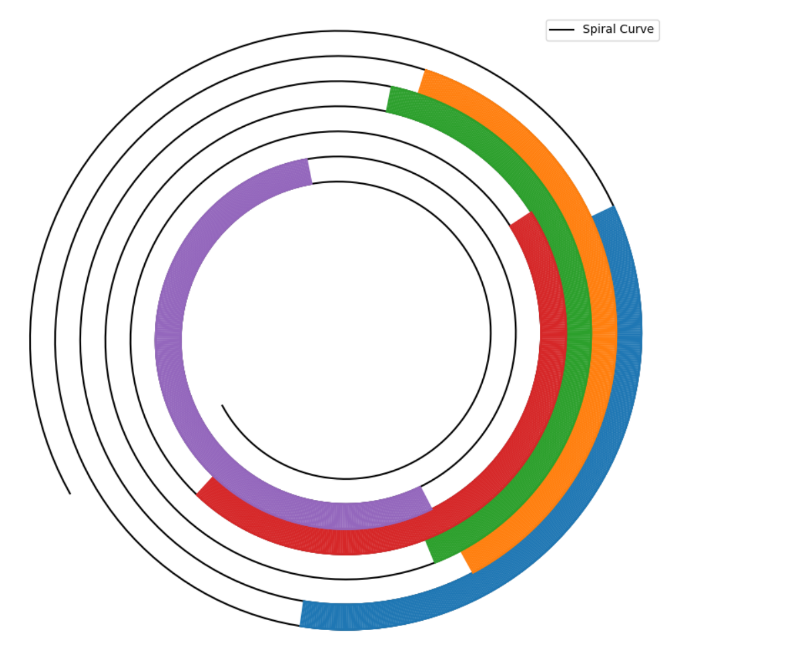
在进行模拟的过程中，需要频繁地计算两个角度之间水管的长度，但是由于 Wirtz 泵的复杂曲线特性，很难导出直接的方程得到精确解。

我们以最简单的Archimedean curve为例，其正向求解函数如下

代码中需要频繁利用到，而这个复杂函数的反函数，也并不好解。

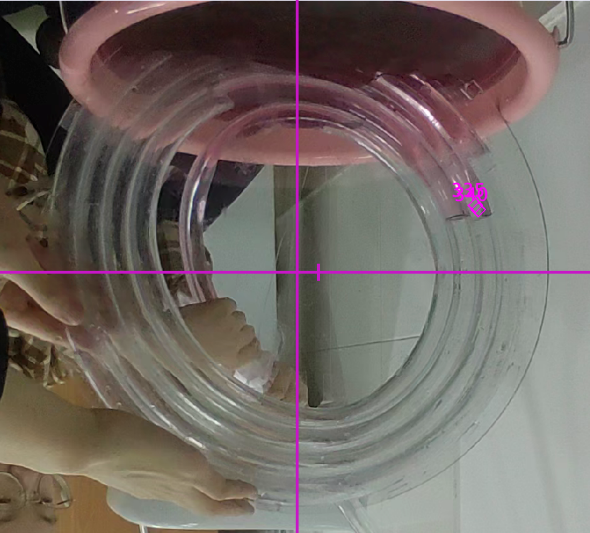
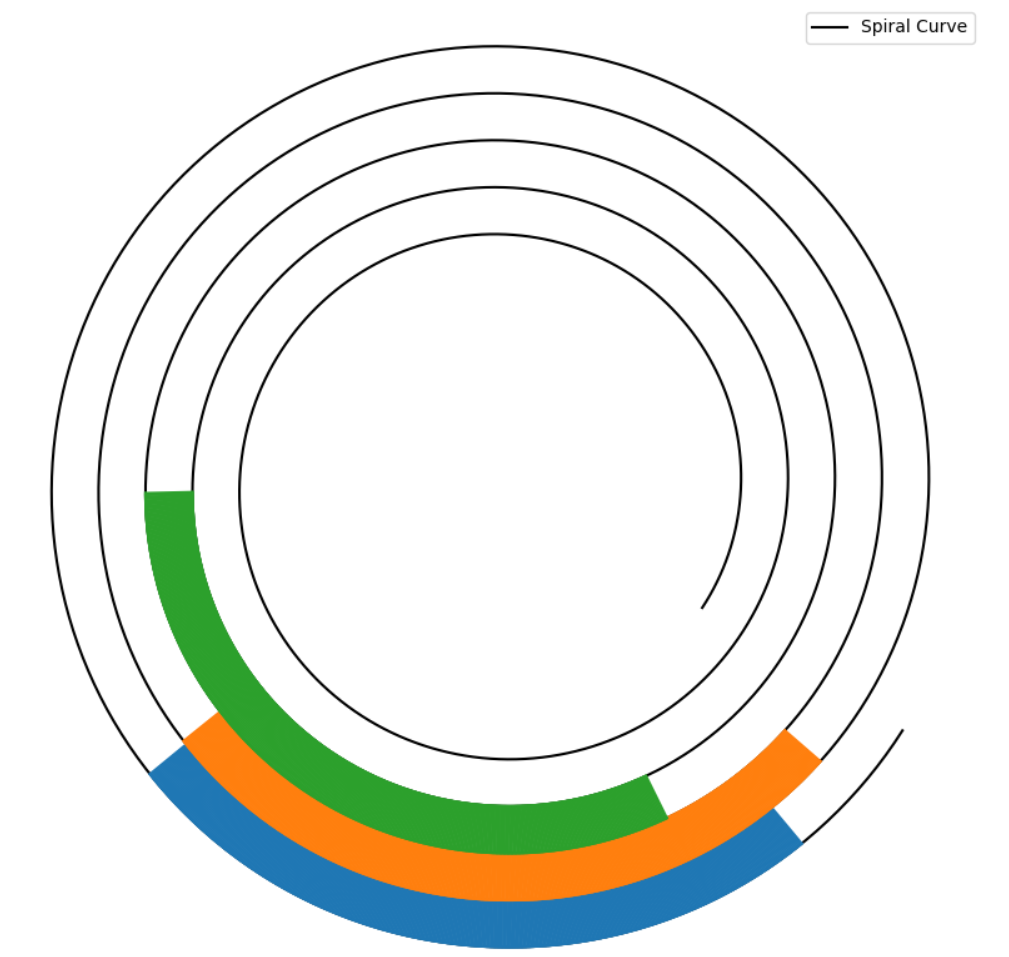
通常的方法是使用牛顿迭代法或在解空间中进行遍历搜索，但过于费时费力；预先存储一个字典则很难覆盖全部空间。为了解决这一问题，我们借用了机器学习中的多层感知机（MLP），凭借其强大的拟合能力，可以出色完成这一传统方法难以解决的问题。为了达到更好的精度，我们将求解区域划分为(0,0.5) 和[0.5,10]，MLP在这两个区域分别能够达到和的MSE loss。



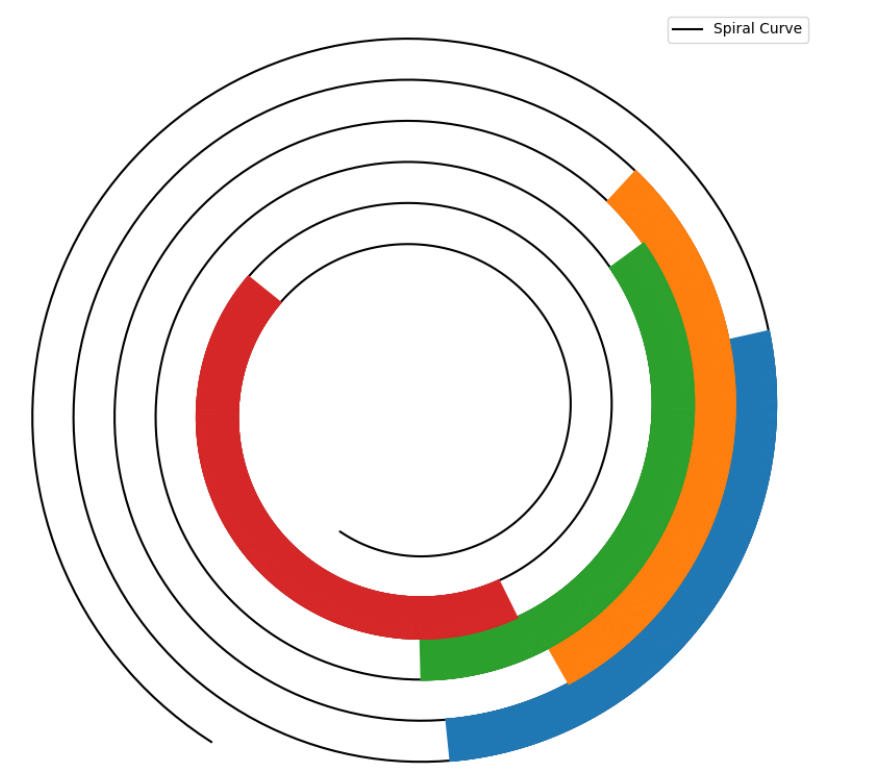
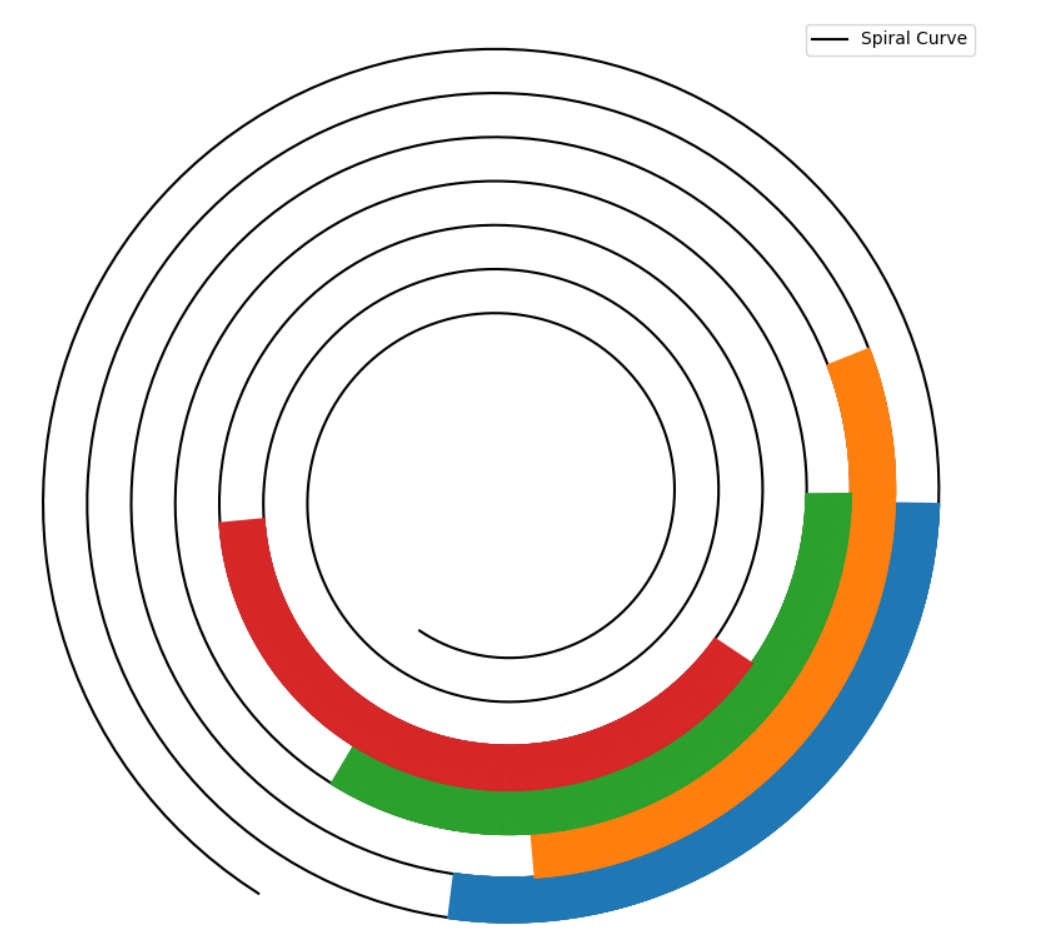
为了更直观地展示水柱和空气柱在系统中的分布变化，我们实现了一个可视化功能模块。该模块的核心在于记录每个水柱在不同时刻的两个关键参数alpha 和 beta。它能够将代码结果与真实的实验视频进行直接对比，来评估我们的代码与实际的差距和相似性。同时，我们也可以利用可视化模块来弥补代码中缺失的物理约束部分，直接排除一些不稳定的状态，从而更好解决前面提到的多解问题。

实验中，我们观察到系统存在显著的漏水和漏气现象，具体指两段相邻的水柱和两段相邻的空气柱会发生合并，这些现象对系统的性能产生了复杂的影响：虽然在短期内会暂时降低水柱的最大长度，但令人意外的是，最终却使得水能够被泵送到更高的位置。为了使代码更贴近实际情况，我们在代码中简单模拟了这两个现象——逐步减小字典中每一个键值对的值，这样的拟合虽然和实际的漏水/漏气机制有较大差异，但是在最终可视化和结果中均表现出了不错的效果。

接下来，我们将展示代码实际运行的效果和对比：



|  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- |
| 组别 | 螺旋泵最大半径R/m | 螺旋常数 | 最大转动圈数 | 螺旋圈数N | 实验测得水柱最高高度 | 代码模拟得水柱最高  高度 |
| 1 | 0.2000 | 0.00301 | 31 | 7 | 1.1227 | 1.1229 |
| 2 | 0.2080 | 0.00350 | 31 | 6 | 0.8640 | 0.9662 |



最终的拟合误差会相差在10cm水柱范围之内，并且每次拟合均可以在五分钟之内完成计算，证明了程序拟合的精度和效率。