# 数学实验课程第一次实验报告

PB21010479　王曹励文

2023 年 9 月 12 日

## 1　牛顿迭代法求解零点

### 1.1　题目

设

$$f_n(x) = x - \frac{x^3}{3!} + \frac{x^5}{5!} - \ldots + (-1)^n \frac{x^{2n+1}}{(2n+1)!},$$

对 $n = 3, 4, 5, 6, 7$, 依次实现牛顿切线法求出 $f_n(x)$ 在 $x = 3$ 附近的零点. 并观察随着 $n$ 的增加，所求出的零点的变化趋势，并对此做出解释.

### 1.2　解答

本题利用牛顿迭代法求解零点. 当使用牛顿迭代法来寻找方程 $f(x) = 0$ 的根时，它的原理基于以下思想:

1. 线性逼近: 假设已经有一个近似的根 $x_n$,牛顿迭代法试图找到一个更好的近似根 $x_{n+1}$,使得 $f(x_{n+1})$ 尽可能地接近零. 为了做到这一点，尝试通过一个线性逼近来估计新的根.
2. 切线的概念: 牛顿迭代法使用 $f(x_n)$ 处的切线来近似函数 $f(x)$. 切线是函数 $f(x)$ 在点 $(x_n, f(x_n))$ 处的一条线，它的斜率等于 $f'(x_n)$，即函数在 $x_n$ 处的导数值.
3. 迭代公式: 牛顿迭代法基于切线的概念，将切线与 x 轴的交点作为新的近似根. 新的近似根 $x_{n+1}$ 被计算为 $x_{n+1} = x_n - \frac{f(x_n)}{f'(x_n)}$.
4. 重复迭代过程: 这个迭代过程会不断重复，每次都会计算一个新的近似根 $x_{n+1}$，然后将其作为下一次迭代的起点，直到找到一个满足精度要求的根或达到迭代次数的限制.

这一过程的直观解释是，牛顿迭代法通过不断改进当前的猜测根，使其逐渐接近真实根. 这是一种快速收敛的方法，特别适用于那些具有单一根且导数不为零的函数. 但要注意，初始猜测值的选择可能会影响迭代的收敛性，有时甚至可能导致发散. 因此，牛顿迭代法在实际应用中需要谨慎使用.

在以上原理的思想上，我编写了如下代码，适用于求出 $n = 3$ 时，迭代 100 次，精度为 $10^{-6}$ 下的代码. 由于初始值的选取可能导致发散，因此加入了如果在迭代 100 次仍然达不到精度 $10^{-6}$ 时，输出迭代未收敛到指定的精度.

1.1 原始代码

```
f3 = @(x) x - x^3 / factorial(3);
f3_prime = @(x) 1 - x^2 / factorial(2);

x0 = 3;
tolerance = 1e-6;
max_iterations = 100;

iteration = 0;
x_current = x0;

while abs(f3(x_current)) > tolerance && iteration < max_iterations
    x_next = x_current - f3(x_current) / f3_prime(x_current);
    iteration = iteration + 1;
    x_current = x_next;
end

if iteration < max_iterations
    fprintf('零点近似值为: %f\n', x_current);
else
    fprintf('迭代未收敛到指定的精度。\n');
end
```

该尝试有以下几个缺点，并提出了相应的解决方案：

1. 只能求出一个数字 $n$ 在迭代下的零点. 解决方案是定义数组 $n\_values$ 后，利用循环结构.
2. 定义函数的时候使用了匿名函数，这通常比使用内联函数稍慢，在一些需要频繁调用函数的计算中可能会产生性能影响（虽然本题并不很复杂）. 解决方案是采用内联函数，创建两个 $.m$ 文件 $fn\_function.m$ 和 $fn\_prime\_function.m$，来定义函数与导数，方便于在代码中直接调用.
3. 将迭代初值加入进输出中，使得回答更加完善.

更新后的代码见文件夹 1.1. 实验结果如部分 1.3 所示.

从实验结果中可以注意到，随着 $n$ 的增大，牛顿迭代法的实验结果越来越接近 $\pi$. 这是由于

$$sinx = \sum_{n=0}^{\infty}(-1)^n \frac{x^{2n+1}}{(2n+1)!}.$$

为 $sinx$ 的幂级数展开式，随着 $n$ 的增加，求和表达式 $\sum_{k=0}^{n}(-1)^k \frac{x^{2k+1}}{(2k+1)!}$ 越来越逼近真实值 $sinx$. 因此随着 $n$ 的增加，得到的零点越来越接近真实的零点 $\pi$.

### 1.3 实验结果

实验结果如图 1 所示.

```
n = 3 在 x = 3 的零点近似值为: 3.078643
n = 4 在 x = 3 的零点近似值为: 3.148690
n = 5 在 x = 3 的零点近似值为: 3.141148
n = 6 在 x = 3 的零点近似值为: 3.141614
n = 7 在 x = 3 的零点近似值为: 3.141592
```

图 1: 1.1 实验结果

## 2 $\sin\frac{1}{x}$ 图像上距离给定点最近的点

### 2.1 题目

判断一点 $A_k(\frac{1}{k}, \sin k)$ 距离最近的是哪一点，要求 $k$ 为整数. 并根据最近点绘制一系列曲线.

### 2.2 解答

考虑$A_m(\frac{1}{m}, \sin m)$ 与 $A_k(\frac{1}{k}, \sin k)$ 之间的距离，其中 $k$ 与 $m$ 为整数：

$$Dist(A_m, A_k) = \sqrt{(\frac{1}{m} - \frac{1}{k})^2 + (\sin m - \sin k)^2}.$$

则

$$Dist^2 = (\frac{1}{m} - \frac{1}{k})^2 + (\sin m - \sin k)^2.$$

之后我编写了在给定 $k$ 值和 $m$ 值范围时求距离 $A_k$ 最近的点 $A_m$ 的程序，该代码的逻辑为给定范围区间对距离进行遍历，用内置函数求最小值，并返回对应的点，见文件 1.2，通过调整 $m$ 值范围的参数，发现具有 2.3 部分给出的实验结果。

出现该结果的原因为在 $m$ 与 $k$ 均较大时，$\frac{1}{m}$ 与 $\frac{1}{k}$ 充分接近，$Dist^2$ 的主要影响项为 $\sin m - \sin k)^2$，当 $m$ 与 $k$ 相差 $2k\pi$ 时，两者完全相同，但是 $\pi$ 不能被有理数表示，因此需要对 $\pi$ 做有理数逼近，圆周率的疏率与密率恰好为 $\frac{22}{7}$ 与 $\frac{355}{133}$，分子恰为 44 与 710 的 $\frac{1}{2}$.

对 $k$, $k-44$, $k-88$ 等点，$k$ 取 2000 时作图，得到的结果如图3所示，代码见文件 1.2.

### 2.3 实验结果

给定 $k$ 值位于 4000-4010，规定 $m$ 位于 $[k-99, k+100]$ 且不为 $m$，得到结果如图2所示. 发现 $m = k + 44$ 或 $m = k - 44$，与预期结果相同.

规定 $m$ 位于 $[k-999, k+1000]$ 且不为 $m$，得到结果如图3所示. 发现在较大的概率下 $m = k + 710$.

```
对于k = 4001，最近的点是 (0.000253, -0.939644)，距离最小的距离为 0.003046，对应的m为 3957
对于k = 4002，最近的点是 (0.000253, -0.795605)，距离最小的距离为 0.016320，对应的m为 3958
对于k = 4003，最近的点是 (0.000247, 0.079909)，距离最小的距离为 0.014410，对应的m为 4047
对于k = 4004，最近的点是 (0.000253, 0.881955)，距离最小的距离为 0.000554，对应的m为 3960
对于k = 4005，最近的点是 (0.000252, 0.873136)，距离最小的距离为 0.015188，对应的m为 3961
对于k = 4006，最近的点是 (0.000247, 0.061559)，距离最小的距离为 0.015717，对应的m为 4050
对于k = 4007，最近的点是 (0.000247, -0.806614)，距离最小的距离为 0.001637，对应的m为 4051
对于k = 4008，最近的点是 (0.000252, -0.933191)，距离最小的距离为 0.013753，对应的m为 3964
对于k = 4009，最近的点是 (0.000247, -0.201796)，距离最小的距离为 0.016709，对应的m为 4053
对于k = 4010，最近的点是 (0.000247, 0.715129)，距离最小的距离为 0.004108，对应的m为 4054
```

图 2: 1.2 中 $m$ 位于 $[k-99, k+100]$ 实验结果

```
对于k = 4001，最近的点是 (0.000212, -0.403652)，距离最小的距离为 0.000039，对应的m为 4711
对于k = 4002，最近的点是 (0.000212, 0.551779)，距离最小的距离为 0.000067，对应的m为 4712
对于k = 4003，最近的点是 (0.000212, 0.999906)，距离最小的距离为 0.000062，对应的m为 4713
对于k = 4004，最近的点是 (0.000240, 0.528725)，距离最小的距离为 0.000029，对应的m为 4161
对于k = 4005，最近的点是 (0.000212, -0.428564)，距离最小的距离为 0.000064，对应的m为 4715
对于k = 4006，最近的点是 (0.000212, -0.991833)，距离最小的距离为 0.000066，对应的m为 4716
对于k = 4007，最近的点是 (0.000212, -0.643215)，距离最小的距离为 0.000038，对应的m为 4717
对于k = 4008，最近的点是 (0.000212, 0.296772)，距离最小的距离为 0.000060，对应的m为 4718
对于k = 4009，最近的点是 (0.000212, 0.963908)，距离最小的距离为 0.000068，对应的m为 4719
对于k = 4010，最近的点是 (0.000212, 0.744832)，距离最小的距离为 0.000040，对应的m为 4720
```

图 3: 1.2 中 $m$ 位于 $[k-999, k+1000]$ 实验结果

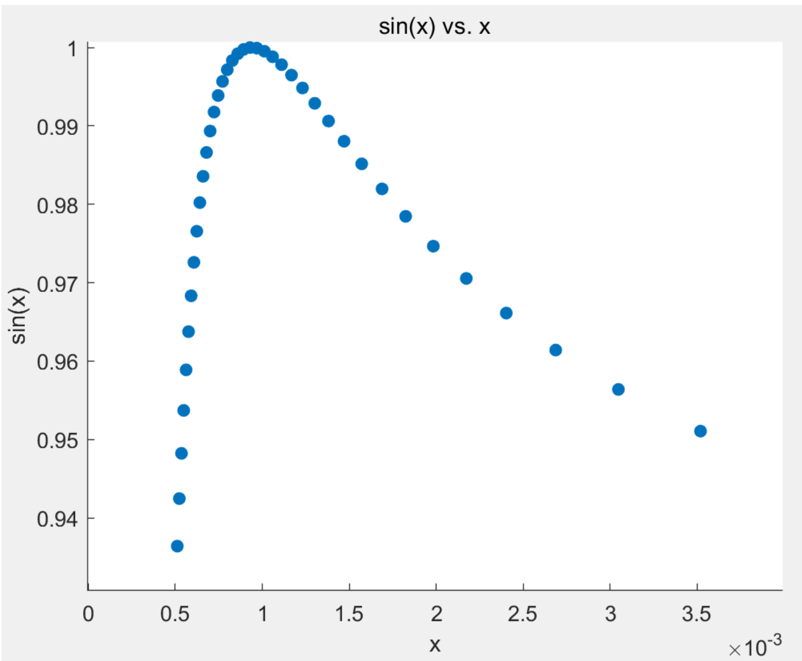



图 4: 作图

# 3 积分与自然对数

## 3.1 题目

1. 对 $n = 10^m$，$m = 3, 4, 5, 6$，用 $MATLAB$ 中的 $sum$ 语句计算“大和” $\Sigma_n$ 与“小和” $\sigma_n$ 以及他们的平均值，观察他们的变化趋势，得出 $S(2)$ 的近似值. 再用求积分语句计算 $S(2)$, 将两者的结果进行比较.

2. 画出函数 $S(x) = \int_1^x \frac{1}{t}\mathrm{d}t$ 在区间 $[0.1, 10]$ 上的图像，并利用牛顿切线法求解该函数的根，观察与自然对数有何关系.

## 3.2 解答

大和和小和具有如下定义：

$$\Sigma_n = \sum_{k=1}^{n} \frac{1}{n+k-1}.$$

$$\sigma_n = \sum_{k=1}^{n} \frac{1}{n+k}.$$

该定义来源于：作出 $\frac{1}{x}$ 的图像，在区间 $[x_{k-1},x_k]$ 上，最大高度为 $\frac{1}{x_{k-1}} = \frac{n}{n+k-1}$，最小高度为 $\frac{1}{x_k} = \frac{n}{n+k}$，区间长度为 $\frac{1}{n}$ 时，最大矩形面积和为 $\sum_{k=1}^{n} \frac{1}{n+k-1}$，最小矩形面积和为 $\sum_{k=1}^{n} \frac{1}{n+k}$. 即对应上和和下和. 利用 $Matlab$ 的 $sum$ 函数可以求出数组的和，sum((1:n)) 即表示从 1 到 $n$ 求和，类似地可以算出 $\Sigma_n$ 与 $\sigma_n$ 的结果. 为了实现对多个 $m$ 的计算，需要先定义 $m\_values$ 数组，再利用循环结构.

算积分时需要利用函数 $integral$，该函数的语法规则是 $integral(fun, xmin, xmax)$，$fun$ 表示函数，$xmin$, $xmax$ 分别表示积分上下限.

实现的代码见文件 1.3.1，实验结果如图7所示. 可注意到随着 $m$ 的增加大和逐渐减小，小和逐渐增大，平均值在保留小数点后六位数字时等于使用函数 $integral$ 算出来的结果.

大和增大，小和减小来源于微积分中的定理：当分隔变细时，达布上和单调不增，达布下和单调不减，且在函数可积时，上积分 = 下积分 = 上和的下确界 = 下和的上确界.

首先尝试用 $plot$ 函数直接绘制 $S(x)$ 的图像，回忆如下代码：

$x^2$ 函数作图

```
x = linspace(-2, 2, 100);
y = x.^2;

plot(x, y);
xlabel('X');
ylabel('Y');
title('函数␣y␣=␣x^2␣的图像');
grid on;
```

该程序可以得到如图5所示的结果. 但是倘若定义变上限积分函数，给出如下所示的代码：

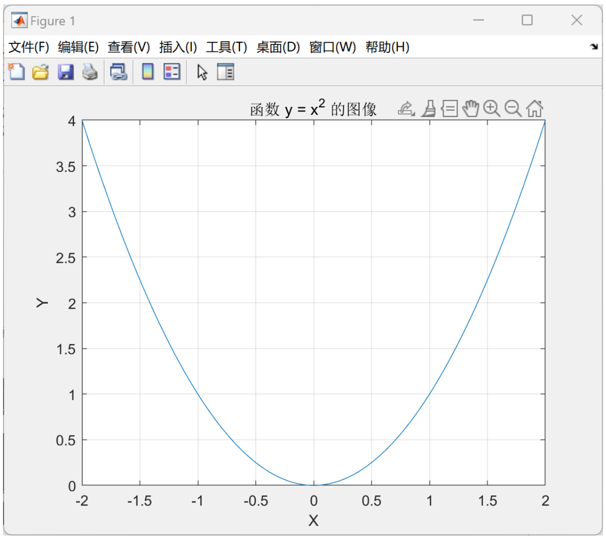



图 5: $x^2$ 函数图像

**尝试作图**

```
x = linspace(0.1, 10, 1000);
S = @(x) integral(@(t) 1/t, 1, x);

plot(x, S);
xlabel('x');
ylabel('S(x)');
title('函数␣S(x)␣=␣\int_{1}^{x}␣\frac{1}{t}␣dt␣的图像');
grid on;
```

但显示结果如图6所示.

错误使用 **plot**
数据参数无效。

图 6: 尝试作图显示的结果

对于某些函数，直接使用 *plot* 函数可以生成近似的图像，但对于积分函数等复杂函数，通常需要进行数值积分并创建一个离散的数据点集合，然后使用 *plot* 函数来绘制图像. 由于 *plot* 函数是用于绘制离散的数据点，而不是直接绘制函数的积分.

因此对代码进行修正，加入对每一个 $S(x)$ 的计算，代码如文件 1.3.2 所示，实验结果如图8所示.

接下来用牛顿迭代法求解函数的根，通过观察图8给出的图像，给予初始值 1.5，精度为 $10^{-6}$，迭代法的原理如 1 题所示，代码见文件 1.3.2，实验结果如图10所示.

注意到 1 为该函数的零点，而 $In1 = 0$，图8中的图像根据微积分基本定理应该为函数 $Inx$. 在图同时画出两个函数的图像，代码文件见 1.3.2. 可以看出两者的图像完全重合，验证了微积分基本定理.

## 3.3 实验结果

```
对于 m = 3，大和为 0.693397，小和为 0.692897，平均值为 0.693147。
对于 m = 4，大和为 0.693172，小和为 0.693122，平均值为 0.693147。
对于 m = 5，大和为 0.693150，小和为 0.693145，平均值为 0.693147。
对于 m = 6，大和为 0.693147，小和为 0.693147，平均值为 0.693147。
积分结果为 0.693147
```

图 7: 大和、小和、平均值

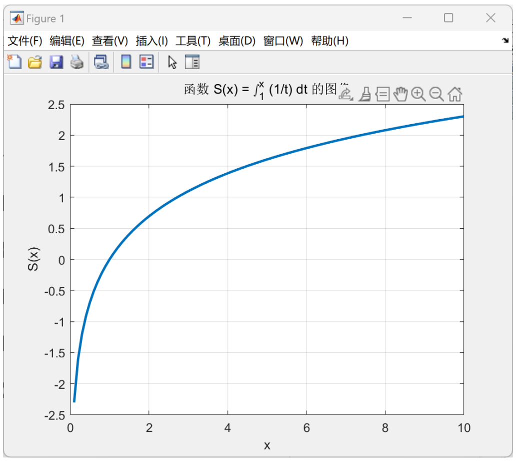



图 8: 变上限积分函数的图像

```
零点近似值为：1.000000
```

图 9: 牛顿迭代法求变上限积分函数的零点

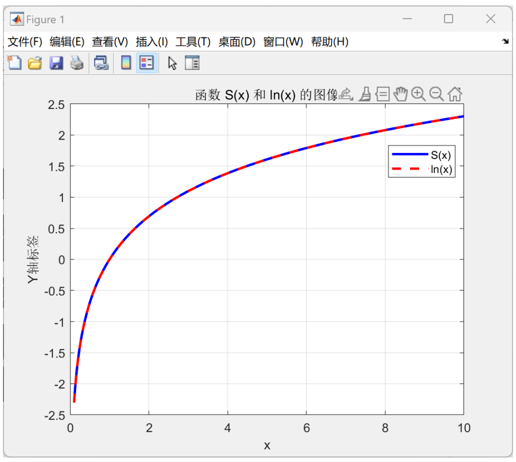



图 10: 图像的完全重合

# 4 圆周率相关

## 4.1 题目

1. 用割圆术迭代公式计算 $\pi$ 的值，根据不同的迭代步数得到表格统计结果.
2. 利用外推公式进行割圆术的迭代加速求解，并统计每步迭代结果.
3. 利用 $Borwein$ 二阶迭代算法计算圆周率.

## 4.2 解答

对于每一小问，采用如下公式进行迭代：

1. 迭代步数为 $n$，有

$$C_n = 6 \cdot 2^n.$$

$$a_0 = 1.$$

$$a_n = \sqrt{2 - 2\sqrt{1 - (\frac{a_{n-1}}{2})^2}}.$$

$$S_n = 3 \cdot 2^{n-1} \cdot a_{n-1}.$$

$$f_n = S_n.$$

$$F_n = 2S_n - S_{n-1}.$$

其中 $C_n$ 表示边数，$f_n$ 与 $F_n$ 分别表示下界与上界.

编写代码，给出计算结果表格，代码见文件 2.1，结果见图11.

2. 迭代步数为 $n$，有

$$G_n = S_{n-1} + \frac{4}{3}(S_n - S_{n-1}).$$

$G_n$ 表示 $\pi$ 的估计值. 代码见文件 2.1，结果见图12.

3. 在前两个实验中，我发现 $MATLAB$ 的索引值要求为正整数，因此在本题中修改初值，方便代码的编写，给出迭代公式：

$$y_1 = \frac{1}{\sqrt{2}}.$$

$$\alpha_1 = \frac{1}{2}.$$

$$y_{n+1} = \frac{1 - \sqrt{1 - y_n^2}}{1 + \sqrt{1 - y_n^2}}.$$

$$\alpha_{n+1} = (1 + y_{n+1})^2 \alpha_n - 2^{n+1} y_{n+1}.$$

$$G_n = \frac{1}{\alpha_n}.$$

代码见文件 2.1，结果见图13.

### 4.3 实验结果

```
对于n = 1, Cn = 12.000000, fn = 3.105829, Fn = 3.211657.
对于n = 2, Cn = 24.000000, fn = 3.132629, Fn = 3.159429.
对于n = 3, Cn = 48.000000, fn = 3.139350, Fn = 3.146072.
对于n = 4, Cn = 96.000000, fn = 3.141032, Fn = 3.142714.
对于n = 5, Cn = 192.000000, fn = 3.141452, Fn = 3.141873.
对于n = 6, Cn = 384.000000, fn = 3.141558, Fn = 3.141663.
对于n = 7, Cn = 768.000000, fn = 3.141584, Fn = 3.141610.
对于n = 8, Cn = 1536.000000, fn = 3.141590, Fn = 3.141597.
```

图 11: 2.1.1 实验结果

```
对于n = 1, Gn = 3.141105.
对于n = 2, Gn = 3.141562.
对于n = 3, Gn = 3.141591.
对于n = 4, Gn = 3.141593.
对于n = 5, Gn = 3.141593.
对于n = 6, Gn = 3.141593.
对于n = 7, Gn = 3.141593.
```

图 12: 2.1.2 实验结果

```
对于n = 1000, G_n = 3.141593.
```

图 13: 2.1.3 实验结果

# 5 蒙特卡罗方法

### 5.1 题目

使用蒙特卡罗方法计算由三个圆柱面 $x^2+y^2=1$、$x^2+z^2=1$ 和 $y^2+z^2=1$ 围成的立体的体积.

### 5.2 解答

由蒙特卡洛方法的核心思想，生成随机点阵用满足条件点阵的量度除以点阵的总量度，该算法由以下步骤实现：

1. 在一个包含立体的三维空间中生成大量的随机点，确保这些点均匀分布在立体的包围盒内，本题中选取 $x$, $y$, $z$ 均位于 [-1,1] 之间.
2. 对于每个随机点，检查它是否在三个圆柱面的内部。可以通过检查点是否满足圆柱面的方程来确定.
3. 统计在立体内的随机点数量.

4. 通过以下公式得出计算立体的体积的估计值：体积估计值 = 立体包围盒的体积 * (在立体内的随机点数量 / 总随机点数量).

5. 通过增加生成的随机点数量，来提高体积估计的精度.

6. 我又加入了可视化效果.

根据以上逻辑编写的代码如 2.2 所示，得到的模拟图如图14所示.

### 5.3 实验结果

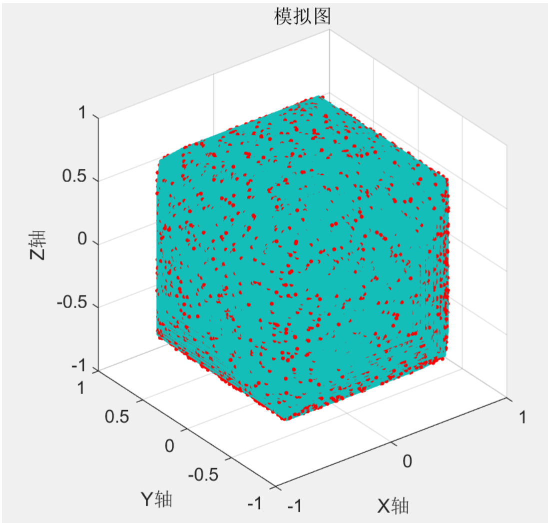



图 14: 可视化效果

该代码可以实现多次不同的模拟，且由于生成随机数，每一次模拟的结果可能存在略微差别，我模拟的三次结果分别为：4.695680、4.698400、4.681040，规定的点数为 10000.

## 6 $\sin n$ 的极限状态的规律

### 6.1 题目

研究数列 $a_n = \sin n$ 的极限状态的规律：

1. 在平面上画出点列 $(n,\ a_n)$，$n = 1,\ 2,\ \cdots N$. 如 $N$=5000；根据该图形，判断数列 $a_n$ 的极限是否存在.

2. 从上述图形中观察点列的分布规律.

3. 任取区间 $[a,\ b]\subseteq [1, 1]$，画出数列中落在区间 $[a,\ b]$ 中的点，将区间 $[a,\ b]$ 放大并取不同的 $N$，观察落在区间 $[a,\ b]$ 中的点集的变化.

## 6.2 解答

1. 本题第一问即用 plot 函数做二维图形，只需要定义数组 $n$，计算数组 $a_n$，作图即可. 如图**??**所示，可以发现数列 $a_n$ 在子列下可以收敛到不同的数字，具有多个极限点，因此数列不收敛，极限不存在.

2. 在给出的图形中，发现图形从外观看好像可以连成连续的曲线，点列的分布较为对称，形成的图案也较为美观；点列在水平轴上波动，但没有明显的趋势，反映了正弦函数的周期性质；另一方面，若选取自然数不同的子列，$a_n$ 会收敛到多个不同的值，在数学分析中被成为极限点，而最大与最小的极限点 1 和-1 被称作上极限与下极限，不过在这里由于 $n$ 为整数，不可以被严格取到，但是可以足够逼近.

3. 从任何区间中选择满足条件的点再作图，只需要定义一个满足条件的数组，利用循环与判断语句将满足条件的存储在新的数组中，代码见 4.1 所示. 注意到对于区间的修改，可以发现不同位置的疏密存在差异，越靠近边缘的区域貌似越密，靠近 0 的区域貌似越稀疏，如图16所示，也与最开始作出的图15显示的结果相同. 若改变 $N$ 的大小，发现图像并没有过多疏密与形状的变化，只是 $N$ 越大，图像中包含的元素个数更多，表明了函数的周期性，如图17所示.

## 6.3 实验结果

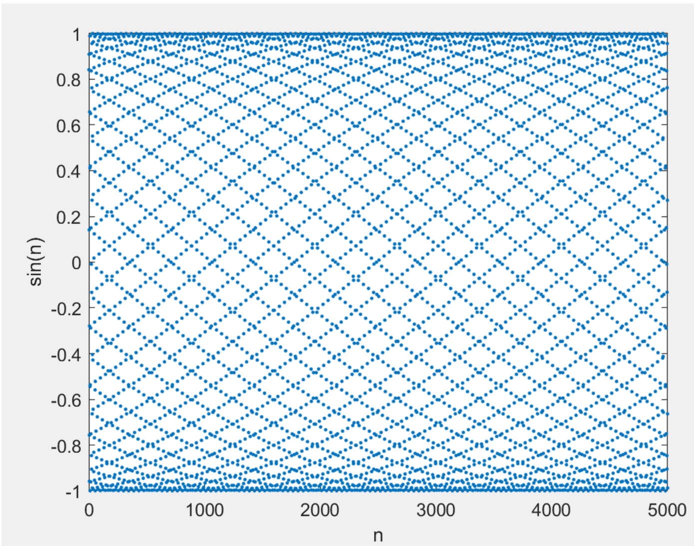
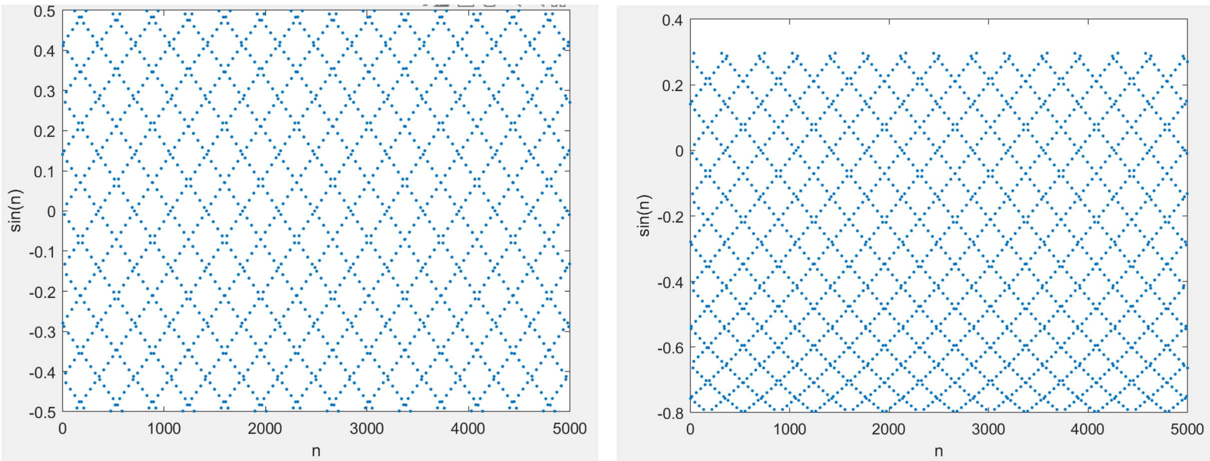
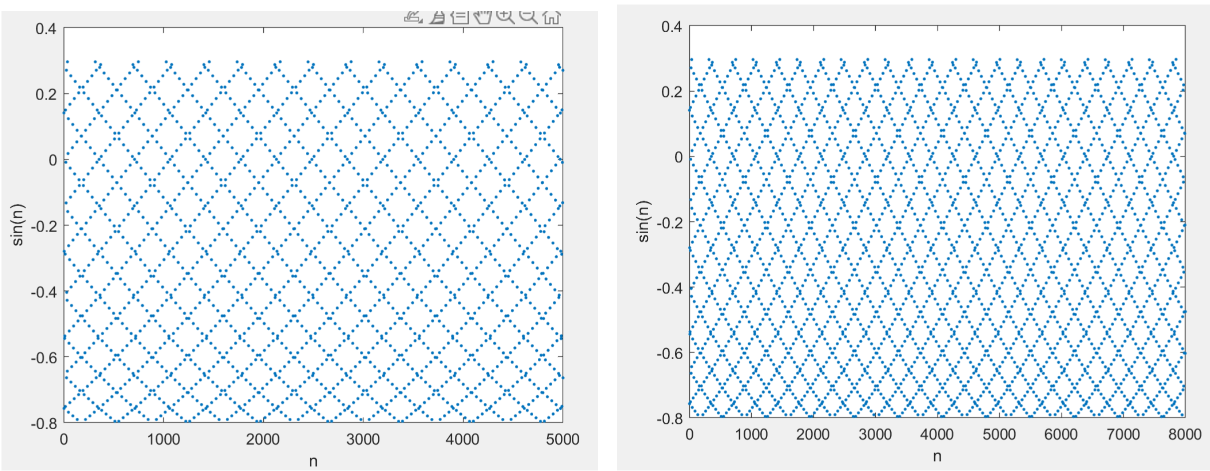



图 15: 数列 $a_n$ 作图结果




(a) a=-0.5,b=0.5　　(b) a=-0.8,b=0.3

图 16: 越靠近边缘的区域貌似越密，靠近 0 的区域貌似越稀疏




(a) $N$=5000　　(b) $N$=8000

图 17: $N$ 变化时图像的变化