第 10 次课 数列和差分方程

Python 科学计算

周吕文

宁波大学，机械工程与力学学院

2024 年 9 月 1 日

Notes

数列

数列是数学中的核心主题之一

$$x_0, x_1, x_2, \cdots, x_n, \cdots$$

实例

- $1,\ 3,\ 5,\ 7,\ 9, \cdots$ $\quad x_0 = 1,\ x_n = 2n+1 = x_{n-1}+2$
- $1,\ 4,\ 9, 16, 25, \cdots$ $\quad x_0 = 1,\ x_n = (n+1)^2 = x_{n-1}+2n+1$
- $1,\ 1,\ 2,\ 6, 24, \cdots$ $\quad x_0 = 1,\ x_n = n! = n \times x_{n-1}$
- $1,\ \frac{1}{2}, \frac{1}{3}, \frac{1}{4}, \frac{1}{5}, \cdots$ $\quad x_0 = 1,\ x_n = \frac{1}{n+1}$

有限与无限数列

- 无限数列：项数无限多的数列（$n \to \infty$）
- 有限数列：实际应用中通常是有限的（$n \le N$）

Notes

差分方程

定义

- 当数列用于描述现实世界现象时，通常没有直接的公式表达。
- 可以建立描述数列规律的一个或多个方程，这些方程称为“差分方程”。
- 差分方程是一种递推地定义一个数列的方程式，数列的每一项被定义为前若干项的函数。

求解

- 差分方程可以通过计算机轻松解决，生成数列。
- 仅用纸笔解出数列通常较难或不太可能。
- 差分方程的 python 求解主要依赖于循环和数组。

Notes

提要

Notes

例子：利息计算

初始金额 x_0，在年利率为 p 的银行，n 年后将增长到多少？

$$x_n = x_{n-1} + px_{n-1}$$

growth_years.py

```python
import matplotlib.pyplot as plt, numpy as np
x0 = 100       # 初始金额
p = 5/100      # 年利率
N = 4          # 年数
x = np.zeros(N+1); x[0] = x0
t = np.arange(N+1)
for n in t[1:]:
    x[n] = x[n-1] + p*x[n-1]
    print('%2d, %5.2f'% (n, x[n]))
plt.plot(t, x, 'ro')
plt.show()
```

```
>>>
 1, 105.00
 2, 110.25
 3, 115.76
 4, 121.55
```

Notes

例子：利息计算

日利率模型：若 p 为年利率，D 为一年的天数，每日利率为

$$r = \frac{p}{D}$$

如何计算两个日期之间的天数

```python
>>> import datetime
>>> date1 = datetime.date(1994, 7, 29)
>>> date2 = datetime.date(2019, 3, 24)
>>> diff = date2 - date1
>>> diff.days
9004
```

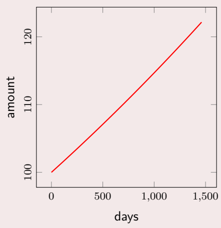

```python
>>> # 前女友朋友圈今天发了一张照片，你能算算孩子是哪天出生吗？
```


Notes

例子：利息计算

growth_days.py

```python
import datetime, matplotlib.pyplot as plt, numpy as np
x0 = 100       # 初始金额
p = 5/100      # 年利率
r = p/365      # 日利率
date1 = datetime.date(2007, 8, 3)
date2 = datetime.date(2011, 8, 3)
diff = date2 - date1
N = diff.days  # 天数
x = np.zeros(N+1); x[0] = x0
t = np.arange(N+1)
for n in t[1:]:
    x[n] = x[n-1] + r*x[n-1]
plt.plot(t, x, '-r')
plt.xlabel('days'); plt.xlabel('amount')
plt.show()
```



Notes

课堂练习

1 线性下降的利息计算　　文件名：growth_days_timedep.py

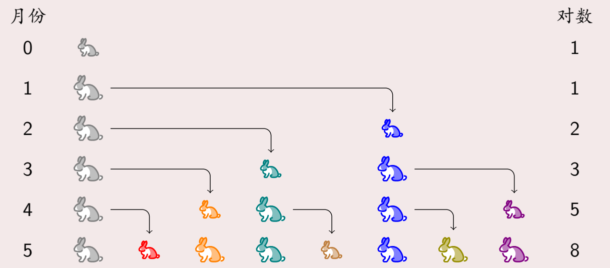
- 假设在 2022 年 2 月 22 日至 2024 年 11 月 14 日期间，某银行的利率 p 从 3% 线性下降至 1.5%。
- 如果在 2022 年 2 月 22 日存入 100 元本金，那么到 2024 年 11 月 14 日时，本息总额将是多少？

Notes

例子：兔子繁殖过程

问题：一对初生的小兔子一年之内可以繁殖多少对兔子？

- 如果一对兔子每个月都可以生一对小兔子
- 并且小兔子在出生两月后就具有生育能力



Notes

例子：兔子繁殖过程 - 模型

差分方程：$n_0(k)$ 和 $n_1(k)$ 分别表示第 k 月的幼兔和成兔数量（对）

$$n_0(k+1) = n_1(k), \qquad n_1(k+1) = n_0(k) + n_1(k)$$

推算：$n_0(0) = 1$, $n_1(0) = 0$

k	0	1	2	3	4	5	6	7	8	9	10	11
$n_0(k)$	1	0	1	1	2	3	5	8	13	21	34	55
$n_1(k)$	0	1	1	2	3	5	8	13	21	34	55	89
$n(k) = \sum_i n_i(k)$	1	1	2	3	5	8	13	21	34	55	89	144

兔子的繁殖过程是斐波那契数列

$$\left.\begin{aligned} n_1(k+1) &= n_0(k) + n_1(k) = n(k) \\ n_0(k+1) &= n_1(k) = n(k-1) \end{aligned}\right\} \Longrightarrow n(k) = n(k-1) + n(k-2)$$

Notes

例子：兔子繁殖过程 - 程序实现

fibonacci.py

```python
import numpy as np
N = 12
n0, n1 = np.ones(N+1), np.zeros(N+1)
for k in range(1,N+1):  # 差分方程
    n0[k] = n1[k-1]
    n1[k] = n0[k-1] + n1[k-1]
x1 = n0 + n1

x2 = np.ones(N+1)
for k in range(2, N+1): # 斐波那契数列
    x2[k] = x2[k-1] + x2[k-2]

print('%2s %3s %3s' % ('k', 'x1', 'x2'))
for k in range(N+1):
    print('%2d %3d %3d' % (k, x1[k], x2[k]))
```

```
>>>
 k  x1  x2
 0   1   1
 1   1   1
 2   2   2
 3   3   3
 4   5   5
 5   8   8
 6  13  13
 7  21  21
 8  34  34
 9  55  55
10  89  89
11 144 144
12 233 233
```

Notes

课堂练习

2 兔子繁殖过程 II 　　　　文件名：rabit.py

- 假设一对兔子每月可以生一对小兔，小兔在出生两月后具有生育能力。
- 兔子存活三个月，完成生育任务后就离开这个群体（出售）。
- 问题：一对初生小兔子一年内繁衍并保存下来的群体有多大？

提示：可将兔子分为幼兔 n_0、两月龄成兔 n_1、三月龄且完成生育的老兔 n_2，则有以下状态转移差分方程：

- 幼兔：上个月成兔和老兔的繁殖结果 $n_0(k+1) = n_1(k) + n_2(k)$
- 成兔：上个月幼兔的发育结果 $n_1(k+1) = n_0(k)$
- 老兔：上个月成兔的发育结果 $n_2(k+1) = n_1(k)$

Notes

例子：种群增长

银行货币增长模型 - 指数增长

$$x_n = x_{n-1} + rx_{n-1} = x_{n-1}(1+r) = x_0 C^n = x_0 e^{n \ln C}$$

- 银行货币的增长是时间（n）的指数函数。
- 人口、动植物种群数量在无限资源的情况下也呈现这种增长。
- 现实情况下，存在环境限制，种群数量往往存在上限 M。

如何模拟存在环境限制的种群数量的增长？ - Logistic 增长模型

- 当种群数量数量较小时（$x \ll M$），增长率 $r = r_0 > 0$。
- 当种群数量接近较大时（$x \approx M$），增长率 $r \to 0$。
- 假设 $r(x)$ 是种群数量的线性减函数，且 $r(M) = 0$:

$$r(x) = r_0 \left(1 - \frac{x}{M}\right)$$

Notes

例子：种群增长

培养皿内酵母菌株，k 到 $k+1$ h 平均增长率：$\dfrac{x(k+1)-x(k)}{x(k)}$

时间/h	0	1	2	3	4	5	6	7	8	9
菌株数	9.6	18	29	47	71	119	175	257	351	441
增长率	-	0.906	0.585	0.628	0.506	0.675	0.466	0.474	0.363	0.257
时间/h	10	11	12	13	14	15	16	17	18	
菌株数	513	560	595	629	641	651	656	660	662	
增长率	0.164	0.090	0.063	0.058	0.018	0.016	0.007	0.006	0.003	

每小时增长率和最大种群数量

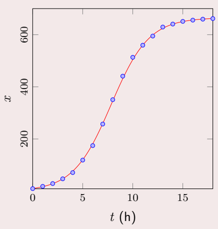
$$r_0 = 0.54, \qquad M = 665$$

Notes

例子：种群增长

logistic_growth.py

```python
import numpy as np, matplotlib.pyplot as plt
t = [  0,   1,   2,   3,   4,   5,   6,   7,   8,\
       9,  10,  11,  12,  13,  14,  15,  16,  17,  18]
x = [9.6,  18,  29,  47,  71, 119, 175, 257, 351,\
     441, 513, 560, 595, 629, 641, 651, 656, 660, 662]
r0 = 0.54/60   # 每分钟增长率
M = 665        # 最大种群数量
ti = np.arange(0, 18, 1/60)
xi = np.zeros_like(ti); xi[0] = x[0]
for n in range(1,len(ti)):
    r = r0*(1-xi[n-1]/M)
    xi[n] = xi[n-1] + r*xi[n-1]
plt.plot(t, x, 'o', ti, xi, '-')
plt.xlabel('t (h)'); plt.ylabel('x')
plt.show()
```



Notes

提要

Notes

泰勒级数

泰勒级数 ⟶ 麦克劳林级数（$a = 0$ 的泰勒级数）

$$f(x) = \sum_{n=0}^{\infty} \frac{f^{(n)}(a)}{n!}(x-a)^n \xrightarrow{a=0} f(x) = \sum_{n=0}^{\infty} \frac{f^{(n)}(0)}{n!}x^n$$

几种常用函数的麦克劳林级数

- $\mathrm{e}^x = \sum_{n=0}^{\infty} \frac{x^n}{n!} = 1 + x + \frac{x^2}{2!} + \frac{x^3}{3!} + \cdots$
- $\sin x = \sum_{n=0}^{\infty} \frac{(-1)^n}{(2n+1)!}x^{2n+1} = x - \frac{x^3}{3!} + \frac{x^5}{5!} - \cdots$
- $\ln(1+x) = \sum_{n=1}^{\infty} \frac{(-1)^{n+1}}{n}x^n = x - \frac{x^2}{2} + \frac{x^3}{3} - \cdots + \frac{(-1)^{n+1}}{n} + \cdots$

Notes

泰勒级数截断近似

实际应用中，泰勒级数只取有限项

$$f(x) \approx \sum_{n=0}^{N} \frac{f^{(n)}(0)}{n!}x^n, \quad \text{例如：} \mathrm{e}^x = \sum_{n=0}^{\infty} \frac{x^n}{n!} \approx 1 + x + \frac{1}{2}x^2 + \frac{1}{6}x^3$$

exp_Taylor_series_diffeq.py

```python
def exp_sum(x, N):
    e = 0
    for n in range(N):
        e += x**n/math.factorial(n)
    return e
```

```python
>>> math.exp(2)
7.38905609893065
>>> [exp_sum(2,N) for N in [2, 4, 16]]
[3.0, 6.333333333333333, 7.389056095384136]
```

Notes

用差分方程计算泰勒级数

用差分方程表示 $\mathrm{e}^x = \sum_{n=0}^{\infty} \frac{x^n}{n!}$，定义 $e_n = \sum_{k=0}^{n-1} \frac{x^k}{k!}$ 和 $a_n = \frac{x^n}{n!}$

$$e_n = e_{n-1} + \frac{x^{n-1}}{(n-1)!} = e_{n-1} + a_{n-1}, \quad e_0 = 0,\ a_0 = 1$$

$$a_n = \frac{x}{n} \cdot \frac{x^{n-1}}{(n-1)!} = \frac{x}{n}a_{n-1}$$

exp_Taylor_series_diffeq.py

```python
def exp_diffeq(x, N):
    an_prev, en_prev = 1, 0
    for n in range(1, N+1):
        en = en_prev + an_prev
        an = x/n*an_prev
        en_prev, an_prev = en, an
    return en
```

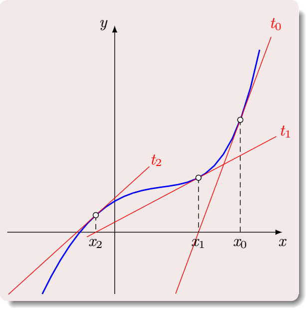
```python
>>> exp_diffeq(2, 2)
3.0
>>> exp_diffeq(2, 4)
6.333333333333333
>>> exp_diffeq(2, 16)
7.389056095384136
```

Notes

提要

Notes

牛顿迭代法求根



过点 $(x_n, f(x_n))$ 的切线

$$t_n(x) = f'(x_n)\cdot(x - x_n) + f(x_n)$$

切线与 x 轴的交点 $t_n(x_{n+1}) = 0$

$$x_{n+1} = x_n - \frac{f(x_n)}{f'(x_n)}$$

求 $f(x) = 0$ 的根 x

$$\lim_{n\to\infty} x_n \to x$$

Notes

牛顿迭代法求根：快速实现

Newton0.py

```python
def Newton(f, x, dfdx, epsilon=1E-7, max_n=100):
    n = 0
    while abs(f(x)) > epsilon and n <= max_n:
        x = x - f(x)/dfdx(x); n += 1
    return x, n, f(x)

import math
g  = lambda x: math.sin(x) - 0.5
dg = lambda x: math.cos(x)
print('x = %g\nn = %d\nf = %g' % Newton(g, 1, dg))
```

```
>>>
x = 0.523599
n = 4
f = -1.59595e-11
```

问题

- 在函数 Newton 的每次循环中，f(x) 被重复计算两次。
- 除数 dfdx(x) 有可能等于 0。
- 无法保存每次迭代的 x 和 f(x) 的值（用于绘图或查看收敛情况）。

Notes

牛顿迭代法求根：改进版本

Newton.py

```python
def Newton(f, x, dfdx, epsilon=1.0E-7, N=100, store=False):
    f_value = f(x)
    n = 0
    if store: info = [(x, f_value)]
    while abs(f_value) > epsilon and n <= N:
        dfdx_value = dfdx(x)
        if abs(dfdx_value) < 1E-14:
            raise ValueError("f'(%g)=%g" % (x, dfdx_value))
        x = x - f_value/dfdx_value
        n += 1
        f_value = f(x)
        if store: info.append((x, f_value))

    return (x, info) if store else (x, n, f_value)
```


Notes

牛顿迭代法求根：改进版本

$$\mathrm{e}^{-0.1x^2}\cdot\sin\left(\frac{\pi}{2}x\right) = 0, \quad 根：x = 0, \pm2, \pm4, \pm6, \cdots$$

Newton.py：初值 x0 = 1.7，预期结果 2，计算结果符合预期

```python
from numpy import sin, cos, exp, linspace, pi
g  = lambda x: exp(-0.1*x**2)*sin(pi/2*x)
dg = lambda x: -2*0.1*x*exp(-0.1*x**2)*sin(pi/2*x) + \
                pi/2*exp(-0.1*x**2)*cos(pi/2*x)

x0 = 1.7
x, info = Newton(g, x0, dg, \
                 store=True)
print('root: %.16g' % x)
for i in range(len(info)):
    print('i=%d: f(%g)=%g' % \
    (i, *info[i]))
```

```
>>>
root: 1.999999999768449
i=0: f(1.7)=0.340044
i=1: f(1.99215)=0.00828786
i=2: f(1.99998)=2.53347e-05
i=3: f(2)=2.43808e-10
```

Notes

牛顿迭代法求根：改进版本

$$e^{-0.1x^2}\cdot\sin\left(\frac{\pi}{2}x\right)=0,\quad 根：x=0,\pm2,\pm4,\pm6,\cdots$$

Newton.py: 初值 x0 = 3.0，预期结果 2 或 4，计算结果不符合预期

```
x0 = 3.0
x, info = Newton(g, x0, dg, \
                 store=True)
print('root: %.16g' % x)
for i in range(len(info)):
    print('i=%d: f(%g)=%g' % \
    (i, *info[i]))
```

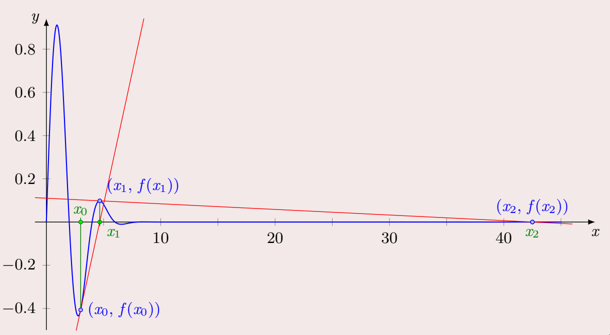
```
>>>
root: 42.49723316011362
i=0: f(3)=-0.40657
i=1: f(4.66667)=0.0981146
i=2: f(42.4972)=-2.59037e-79
```

- 牛顿迭代法可能效果良好，但也可能给出错误的结果！
- 理解牛顿迭代法的原理是正确理解和解释结果的关键！

Notes

牛顿迭代法求根：解释结果

Newton_demo.py



Notes

课堂练习

3 割线法解方程　　文件名：secant.py

求解方程 $f(x)=0$ 的牛顿迭代法需要求解函数 $f(x)$ 的导数：

$$x_n=x_{n-1}-\frac{f(x_{n-1})}{f'(x_{n-1})}\tag{1}$$

求导有时困难，函数导数可用最后两个近似根 x_{n-1} 和 x_{n-2} 处的割线斜率近似：

$$f'(x_{n-1})\approx\frac{f(x_{n-1})-f(x_{n-2})}{x_{n-1}-x_{n-2}}\tag{2}$$

将上式代入式 (1) 可得

$$x_n=x_{n-1}-\frac{f(x_{n-1})(x_{n-1}-x_{n-2})}{f(x_{n-1})-f(x_{n-2})},\quad x_0 \text{ 和 } x_1 \text{ 已知}\tag{3}$$

这就是割线法解方程。编写程序，使用割线法求解方程 $x^5=\sin x$。

Notes

提要

Notes

随机服务系统模拟

随机服务系统：日常生活、工业生产、科学技术、军事领域经常遇到

- 研究银行、海关通道、高速路收费口等服务人员个数的设置和排队规则
- 研究计算机网络网关、移动网络的调度规则，等等

随机服务系统模拟的三个要素

- 输入过程：比如，银行的顾客到来的规律。
- 排队规则：比如，银行有多个柜员时，顾客选最短一队，还是随机选。
- 服务机构：有多少个柜员，服务时间的分布等。

模拟原理

- 按时间顺序记录发生的事件，如顾客到来、接受服务、结束服务等
- 这样的系统的模拟也叫做离散事件模拟（Discrete Event Simulation）

Notes

例子：厕所排队问题

问题：计算排队如厕时间

- 男女平均到达间隔时间为每 10 秒一位（每分钟 6 位）
- 假设男性上厕所的平均时间为 1 分钟，女性平均为 1.5 分钟
- 女厕有 10 个厕位，男厕所有 12 个厕位

模型：$\lambda = 6\ \mathrm{min}^{-1}$，$\mu_{\mathrm{m}} = 1\ \mathrm{min}^{-1}$，$\mu_{\mathrm{w}} = 1/1.5\ \mathrm{min}^{-1}$

- 生成到达厕所的时刻：$A_i = A_{i-1} + X_i$，$X_i \sim \mathrm{Exp}(\lambda)$
- 找出最早空闲的厕位：$T_k = \min(T_j)$，$j = 1, 2, \cdots, c$
- 计算进入厕位的时刻：$S_i = \max(A_i,\ T_k)$
- 计算离开厕所的时刻：$D_i = S_i + Y_i$，$Y_i \sim \mathrm{Exp}(\mu)$
- 更新厕位空闲的时刻：$T_k = D_i$
- 计算排队等待的时间：$W_i = S_i - A_i$

Notes

例子：厕所排队问题 - 程序实现

toilet_queue.py

```python
import numpy as np
def wcqueue(mu, n, lmd, c):
    X = np.random.exponential(1/lmd, n) # 到达时间间隔
    Y = np.random.exponential(1/mu,  n) # 服务时长
    A, S, D = np.zeros(n), np.zeros(n), np.zeros(n)
    T = np.zeros(c)               # 记录各厕位空闲时刻
    for i in range(n):
        A[i] = A[i-1] + X[i] if i>0 else X[i] # 生成到时刻
        k = np.argmin(T)          # 找出最早空闲的厕位
        S[i] = max(A[i], T[k])    # 计算进入厕位的时刻
        D[i] = S[i] + Y[i]        # 计算离开厕所的时刻
        T[k] = D[i]               # 更新厕位空闲的时刻
    W = S - A                     # 计算排队等待的时间
    return W, A, D
```


Notes

例子：厕所排队问题 - 程序实现

toilet_queue.py

```python
lmd = 6                   # [/min] 顾客到达率：单位时间到达人数
n = lmd*8*60*100          # 模拟人数
muw, mum = 1/1.5, 1/1.0 # [/min] 厕位服务率：平均如厕时间倒数
cw, cm = 10, 12           # 厕位数

Ww, Aw, Dw = wcqueue(muw, n, lmd, cw)
Wm, Am, Dm = wcqueue(mum, n, lmd, cm)

print('女性平均排队等待时长 %6.4f min' % np.mean(Ww))
print('男性平均排队等待时长 %6.4f min' % np.mean(Wm))
```

```
>>>
女性平均排队等待时长 0.9519 min
男性平均排队等待时长 0.0039 min
```


Notes

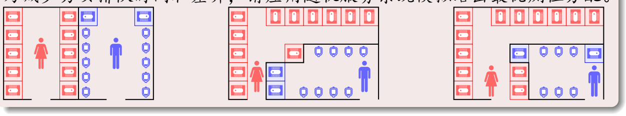
课堂练习（思政案例）

4 最佳厕位分配 文件名：toilet_optimal.py

在一个可建造 20 个隔间的区域，如何合理分配男女厕位？假定男性小便池和隔间所占面积比为 3/4。设女厕位数为 c_{w}（全为隔间），男厕位数为 c_{m}（2 个隔间及 $c_{\mathrm{m}}-2$ 个小便池）。为了最大化空间利用率，有以下约束条件：

$$19 \le (c_{\mathrm{w}}+2)+\frac{3}{4}\cdot(c_{\mathrm{m}}-2) \le 20$$

为减少男女排队时间和差异，请应用随机服务系统模拟给出最优厕位分配。



思政点：男女平等是我国的一项基本国策

- 所谓的男女平等，应考虑女性作为弱势群体和男女客观存在的生理差异
- 我国《公共厕所规划和设计标准》要求女男厕位的比例不应小于 1.5:1

Notes

The End!

Notes

Notes

Notes