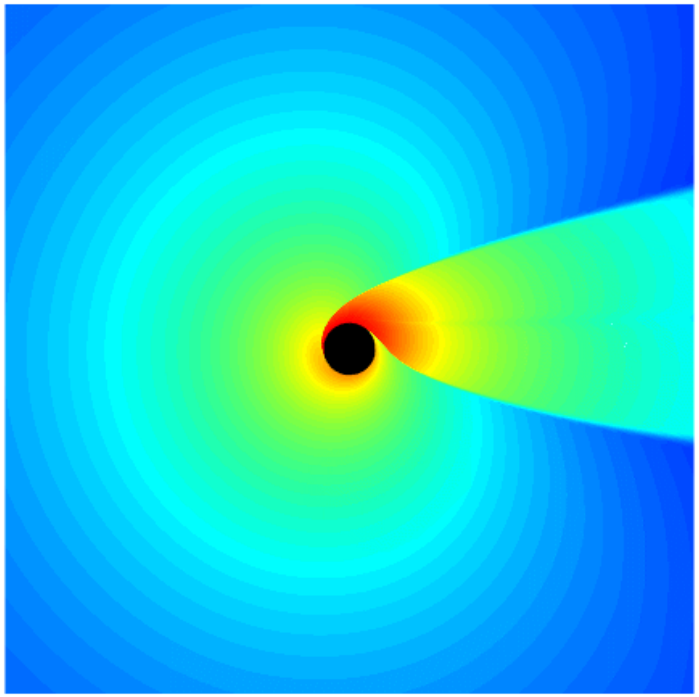
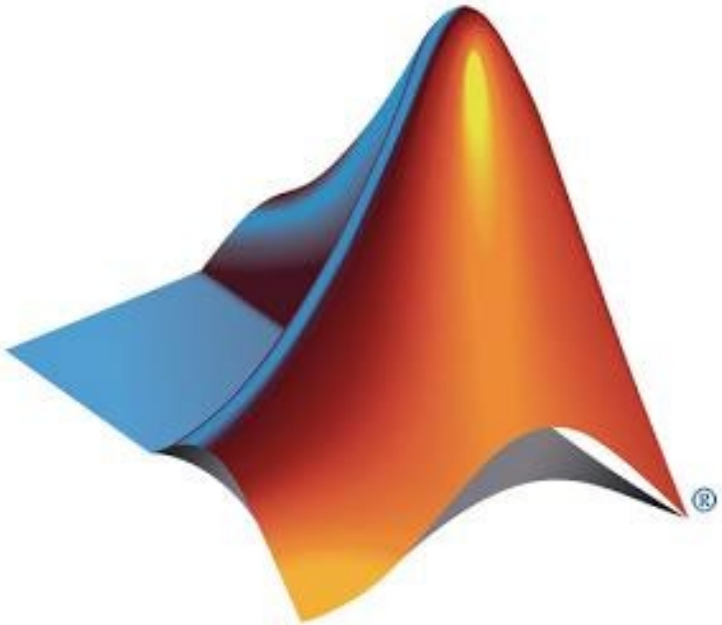
数学实验
Mathematical Experiments





实验七：
代数方程求根实验
Solutions to algebraic equations

实验背景

- 在科学研究和工程实践中，常常会遇到非线性方程或非线性方程组的问题

例 如

1. 在光的衍射理论（the theory of diffraction of light)中,需要求x-tanx=0的根
2. 在行星轨道（planetary orbits）的计算中,对任意的a和b，需要求x-asinx=b的根
3. 在数学中，需要求n次多项式$x^n+a_1x^{n-1}+\ldots+a_{n-1}x+a_n=0$的根

实验背景

- 若干个世纪以来，工程师和数学家花了大量时间用于探索求解方程（组），研究各种各样的方程求解方法。对于方程

$$f(x) = 0 \tag{1}$$

当f(x)为线性函数时，称式(2.1)为线性方程；当f(x)为非线性函数时，称式(2.1)为非线性方程。对于线性方程（组）的求解，理论与数值求解方法的研究成果较为丰富，我们已在线性代数实验部分介绍求解方法；对于非线性方程求解，由于f(x)的多样性，尚无一般的解析解法。例如，当f(x)为n(n≥2)次代数多项式时，式(1)称为n次代数方程，即

$$a_n x^n + a_{n-1} x^{n-1} + \cdots + a_1 x + a_0 = 0 \tag{2}$$

实验背景

- 寻找代数方程的解一直是个重要的数学问题。
- 由代数基本定理知，一元n次代数方程一定有n个根(含复根)。
- **计算上：**
  - 一元一次方程和二次方程很早就找到了公式解。经过努力，16世纪时，意大利数学家塔塔利亚和卡当等人，发现了一元三次方程的求根公式，费拉里找到了四次方程的求根公式。当时数学家们非常乐观，以为马上就可以写出五次方程、六次方程，甚至更高次方程的求根公式了。然而，时光流逝了几百年，谁也找不出这样的求根公式。
  - 大约三百年之后，在1825年，挪威学者阿贝尔（Abel）终于证明了：一般的一个代数方程，如果方程的次数n≥5 ，那么此方程不存在统一的根式解（即由方程的系数通过有限次的四则运算及根号组合而成的公式解）。这就是著名的阿贝尔定理。

实验背景

- 寻找代数方程的解一直是个重要的数学问题。

- 而当f(x)包含三角函数、指数函数、对数函数等类型时，式(1)称为超越方程，例如

$$\ln x - x \sin \frac{\pi}{2} x + 1 = 0 \qquad (3)$$

此类方程不仅很难求得解析解，有时连解的存在性、解的个数也难以判断。

- 前面我们学习了：符号计算解析地求根（只解决少部分问题）

S=solve(eqn1,eqn2,...,eqnM,var1,var2,...,varN)

本次实验我们学习如何用Matlab求代数方程（组）的数值解。

实验目的

1. 理解迭代法原理，并利用迭代法求非线性方程的根。

2. 熟悉MATLAB软件中非线性方程（组）的求解命令及用法。

3. 使用MATLAB解决一些非线性方程问题。

实验的理论基础

1. 迭代格式

将方程f(x)=0转化为等价的方程$x = \varphi(x)$, 并据此构造迭代格式:

$$x_{n+1} = \varphi(x_n)$$

对某个选定的初值x_0进行迭代，得到一个迭代数列$\{x_n\}$。如果数列$\{x_n\}$存在极限，即$\lim\limits_{n\to\infty} x_n = x^*$

称迭代收敛，x^*就是$x = \varphi(x)$的根(不动点)，亦是f(x)=0的根。

2. 零点定理

将若函数f(x)在闭区间[a, b]上连续，且f(a)f(b)<0, 则至少存在一点$c \in (a, b)$, 使得f(c)=0。

实验的理论基础

3. 压缩映射原理

设定义在[a, b]上的函数f(x)满足：对任意x $\in [a,b]$, 有$f(x) \in [a,b]$，且存在一个常数L>0, 使得

$$|f(x)-f(y)| \le L|x-y|, x, y \in [a,b]$$

成立。则当L<1时，称f为[a, b]上的一个压缩映射，且函数f(x)在[a, b]上有唯一的不动点。

实验1：迭代法之二分法

1、**基本架设：**设f(x)在闭区间[a,b]上连续、有单根，且f(a)f(b)<0.

2、**二分法原理：**将区域二分，根据零点定理，判断根在某个分段内，再进行二分，依次推，重复进行，直到满足精度为止。算法流程如下：

Step 1：赋初值a,b及$\mathrm{k}=0, a_k=a, b_k=b$。

Step 2：令$x_k=\frac{a_k+b_k}{2}$，计算$f(x_k)$。

Step 3：若$f(x_k)=0$，则x_k是$f(x_k)=0$的根，停止计算，输出结果$x=x_k$；若$f(a_k)f(x_k)<0$,则令$a_{k+1}=a_k, b_{k+1}=x_k$;若$f(a_k)f(x_k)>0$,则令$a_{k+1}=x_k, b_{k+1}=b_k$。

实验1：迭代法之二分法

Step 4：令k=k+1，$x_k=\frac{a_k+b_k}{2}$，若$|f(x_k)|\le\varepsilon$（ε为精度要求），退出计算，输出结果x_k；若$|f(x_k)|>\varepsilon$，则转入Step 3。

3、误差估计：若执行以上过程，可得到每次缩小1/2的区间序列$\{[a_k,b_k]\}$，在(a_k,b_k)中含有方程的根x^*。当区间长度b_k-a_k很小时，取其中点$x_k=\frac{a_k+b_k}{2}$为根的近似值。因此：

$$|x_k-x^*|\le\frac{1}{2}(b_k-a_k)=\frac{1}{2}\times\frac{1}{2}\times(b_{k-1}-a_{k-1})=\cdots$$

$$=\frac{1}{2^{k+1}}(b-a)$$

实际问题计算时，根据 ε 的数值，利用上式可估计二分次数k。

实验1：迭代法之二分法

小实验：

编写MATLAB程序，使用二分法求方程$x^3+x-1=0$在$[0,1]$内的根的近似值(误差$<10^{-5}$).

实验2：迭代法之不动点迭代

基本思想

将方程f(x)=0转化为等价的方程$x = g(x)$, 并据此构造迭代格式：

$$x_{n+1} = g(x_n)$$

对某个选定的初值x_0进行迭代，得到一个迭代数列$\{x_n\}$。如果数列$\{x_n\}$存在极限，即$\lim_{n\to\infty} x_n = x^*$

称迭代收敛，x^*就是$x = g(x)$的根(不动点)，亦是f(x)=0的根

实验2：迭代法之不动点迭代

一个简单的例子

设计不同迭代算法求方程$x^2 = 2$的正根。

方法1：$x^2 = 2 \Leftrightarrow x = \frac{2}{x}$，从而设置迭代格式$x_{n+1} = \frac{2}{x_n}$。

方法2：$x^2 = 2 \Leftrightarrow x = \frac{2}{x} \Leftrightarrow x + x = x + \frac{2}{x} \Leftrightarrow x = \frac{1}{2}(x + \frac{2}{x})$, 从而设置迭代格式$x_{n+1} = \frac{1}{2}(x_n + \frac{2}{x_n})$。

方法3：$x^2 = 2 \Leftrightarrow x^3 = 2x$，所以$x^3 = 2x - x^2 + 2$, 故$x = \sqrt[3]{2x - x^2 + 2}$，从而设置迭代格式$x_{n+1} = \sqrt[3]{2x_n - {x_n}^2 + 2}$。

方法4：$x^2 = 2 \Leftrightarrow x^2 + x = 2 + x$，所以$x = \frac{2+x}{1+x}$, 故$x = 1 + \frac{1}{1+x}$，从而设置迭代格式$x_{n+1} = 1 + \frac{1}{1+x_n}$。

实验2：迭代法之不动点迭代

一个简单的例子

设计不同迭代算法求方程$x^2 = 2$的正根。

选用初值x=1，同时按以上四种格式进行迭代计算。

编程实现

实验2：迭代法之不动点迭代

一个简单的例子

序号	1	2	3	4
方法1	1.000000000000000	2.000000000000000	1.000000000000000	2.000000000000000
方法2	1.000000000000000	1.500000000000000	1.416666666666667	1.414215686274510
方法3	1.000000000000000	1.442249570307408	1.410200215949785	1.414764790503810
方法4	1.000000000000000	1.500000000000000	1.400000000000000	1.416666666666667
序号	5	6	7	8
方法1	1.000000000000000	2.000000000000000	1.000000000000000	2.000000000000000
方法2	1.414213562374690	1.414213562373095	1.414213562373095	1.414213562373095
方法3	1.414137398906688	1.414224077308350	1.414212110543581	1.414213762828541
方法4	1.413793103448276	1.414285714285714	1.414201183431953	1.414215686274510
序号	9	10	11	...
方法1	1.000000000000000	2.000000000000000	1.000000000000000	...
方法2	1.414213562373095	1.414213562373095	1.414213562373095	...
方法3	1.414213534695966	1.414213566194509	1.414213561845468	...
方法4	1.414213197969543	1.414213624894870	1.414213551646055	...

实验2：迭代法之不动点迭代

如何设计收敛的不动点迭代法？

迭代收敛判别

如果在区间[a, b]上，g为[a, b]上的一个压缩映射，则迭代格式$x_{n+1} = g(x_n)$收敛。

特殊地，如果在区间[a, b]上，$g(x)$连续可导，且满足$|g'(x)| \leq q \leq 1$，则迭代格式$x_{n+1} = g(x_n)$收敛。

实验2：迭代法之不动点迭代

如何设计收敛的不动点迭代法？

根据上述四种迭代格式，也可以从理论分析的方式判别收敛性。

对于方法1，令$\varphi_1(x)=\frac{2}{x}$, 则$\varphi_1'(x)=-\frac{2}{x}$;

对于方法2，令$\varphi_2(x)=\frac{1}{2}(x+\frac{2}{x})$， $\varphi_2'(x)=\frac{1}{2}(1-\frac{2}{x^2})$;

对于方法3，令$(x)=\varphi_3'\sqrt[3]{2x-x^2+2}$，$\varphi_3'(x)=\frac{1}{3}\left(2x-x^2+2\right)^{-\frac{2}{3}}(2-2x)$;

对于方法4，令$\varphi_4(x)=1+\frac{1}{1+x},\varphi_4'(x)=1-\frac{1}{(1+x)^2}$。当$1\le x\le 2$时，$|\varphi_i'(x)|<1(i=2,3,4)$成立，即满足迭代收敛条件，而对于方法1中情形不满足要求。

实验2：迭代法之不动点迭代

用蛛网图观察不动点迭代

实验问题：用求不动点的方法求方程$f(x)=0$的根，这里方程为$f(x)=x^2+x-4=0$. 对这一方程，构造如下三个函数：

$$g_1(x)=4-x^2, g_2(x)=\frac{4}{1+x}, g_3(x)=x-\frac{x^2+x-4}{2x+1}, \quad (3)$$

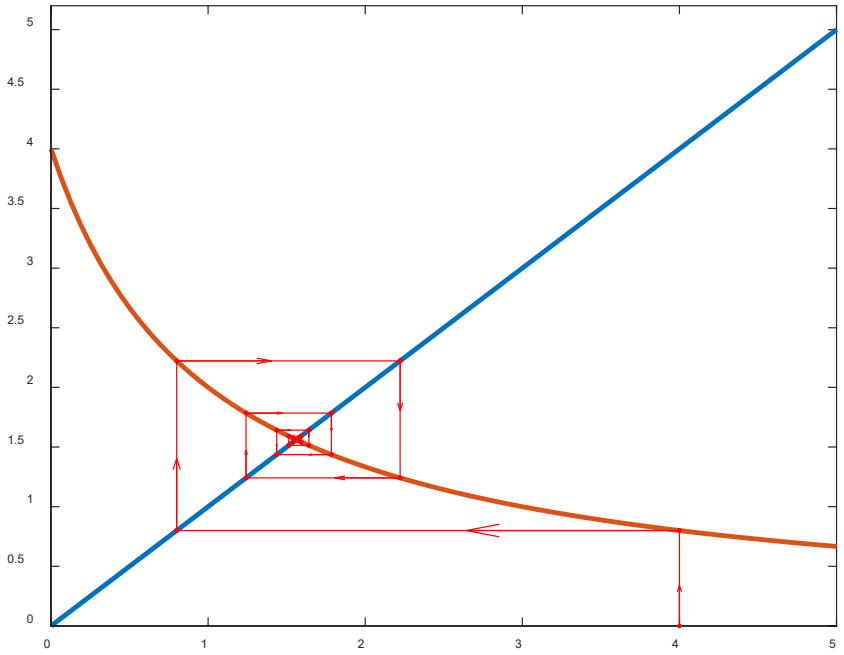
容易验证$f(x)=0 \Leftrightarrow g_i(x)=x(i=1,2,3)$，于是求根转化为求不动点问题，试用如下给出的迭代法观察它们的收敛性和收敛快慢.

实验2：迭代法之不动点迭代

用蛛网图观察不动点迭代

观察：运行观察程序Exp7_3.m，在程序中选择参数s=2,观察函数$g_2(x)=4/(1+x)$的迭代蛛网图

理解蛛网图的原理



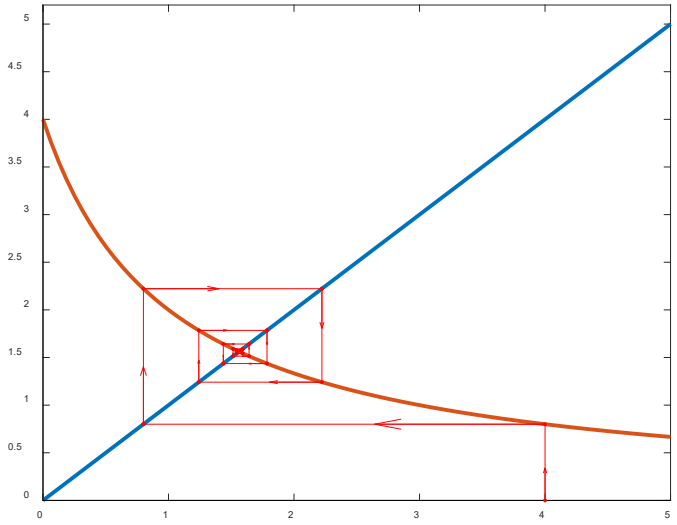
实验2：迭代法之不动点迭代

用蛛网图观察不动点迭代

如图所示，不动点$x^* = g(x^*)$是直线$y = x$和曲线$y = g(x)$交点的横坐标. 选定初始点$x_0 = 4$后，迭代$x_1 = f(x_0)$可以看成如下两个步骤的合成：

(1) 计算$y_1 = g(x_0) = 0.8$，在图中以一条自$x_0 = 4$出发，并与y轴平行的有向线段表示这一过程，箭头指向曲线$y = g(x)$，线段的终点位于点$(x_0, g(x_0)) = (4, 0.8)$；

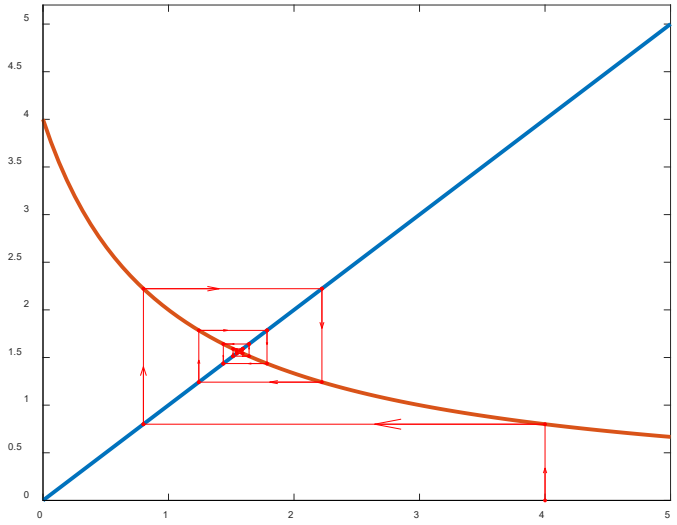
(2) 由$(x_0, g(x_0)) = (4, 0.8)$出发，作平行于x轴的有向线段（如图所示），其终点位于直线$y = x$之上，这时由于终点的横坐标与纵坐标相等，即得$x_1 = y_1 = g(x_1) = 0.8$，也就完成了第一次迭代，在横轴上用箭头指示了$x_1 = 0.8$的位置.



实验2：迭代法之不动点迭代

用蛛网图观察不动点迭代

于是可以如图那样用两条垂直的有向线段表示一次迭代过程. 当迭代继续时，这种表示也随之跟进，最终形成了一个类似于蛛网的迭代过程图，这个用来形象表示迭代过程的图形叫做**蛛网图**.

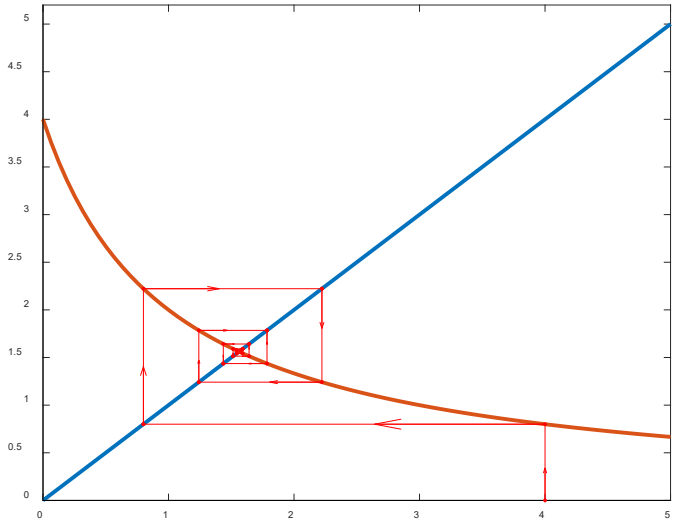


实验2：迭代法之不动点迭代

用蛛网图观察不动点迭代

- 在程序中令参数s=1，即对函数$g_1(x) = 4 - x^2$进行迭代(初始点为$x_0 = 4$)，观察结果，这时迭代发散到负无穷.
- 再选取控制参数s=3, 对函数$g_3(x) = x - (x^2 + x - 4)/(2x + 1)$进行选代，仍取初始点为$x_0 = 4$，并控制坐标系范围和右图一致，以便与$f_2(x) = 4/(1 + x)$的迭代过程进行比较.

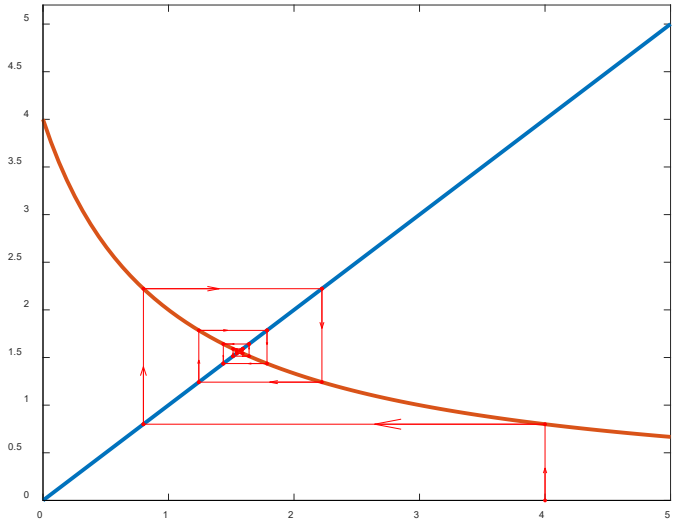
比较，在两种情况下，蛛网图的箭头一直指向点(x^*, x^*)，这里$y^* = x^*$，迭代均收敛到不动点x^*.



实验2：迭代法之不动点迭代

用蛛网图观察不动点迭代

- **思考：**虽然两者均收敛到不动点，但它们收敛速度有快慢之分，从相同的初始点出发，图?的收敛速度显然要比图?的快. 前者在两次迭代后便十分接近不动点了，而后者经过了6次迭代才到达前者的精度. 是什么因素决定了收敛速度的快慢？仔细观察图形，你能够洞察到其中的原因吗？



实验2：迭代法之不动点迭代

简单和复杂：二次函数的迭代和混沌

二次函数是非常简单的，如果将它进行迭代，它还会简单吗？让我们进行如下的观察：观察：对二次函数进行迭代：

$$x_{n+1} = f(x_n) = rx_n(1 - x_n), n = 0,1,2, \ldots \quad (5)$$

其中$r \in (0,1)$是一个可变的参数，观察不同的r值对迭代点列的影响.

实验2：迭代法之不动点迭代

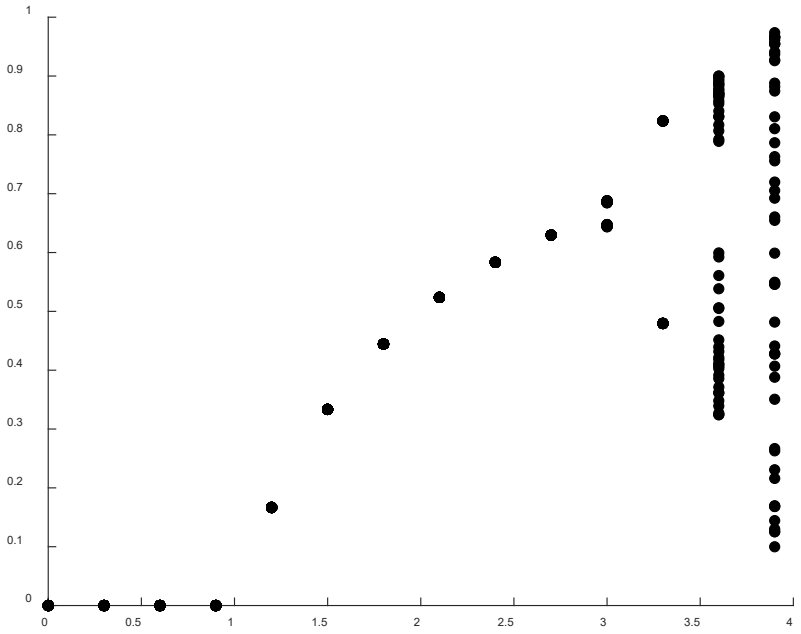
简单和复杂：二次函数的迭代和混沌

步骤1. 固定若干个不同的r的值，观察迭代序列(5)的极限；

$$x_{n+1} = f(x_n) = rx_n(1 - x_n), n = 0,1,2, \ldots \qquad (5)$$

可采用下面的方法：如果对固定的r，极限$x^* = \lim_{n\to\infty} x_n$存在，则n充分大时有$x_n \approx x^*$. 所以按(5)迭代N次，略去前n个迭代值，并将后N-n个迭代值画在以r为横坐标，x为纵坐标的图形窗口中，即可观察迭代序列(5)的极限性态.

取r=[0 0.3 0.6 0.9 1.2 1.5 1.8 2.1 2.4 2.7 3.0 3.3 3.6 3.9]，运行观察程序Exp7_4a.m，结果显示在图中.

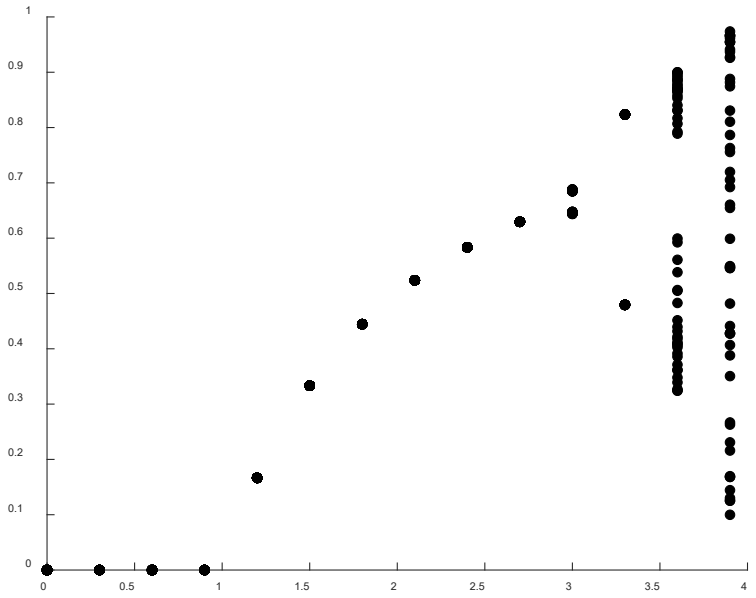


实验2：迭代法之不动点迭代

简单和复杂：二次函数的迭代和混沌

步骤2. 用蛛网图观察三种不同类型的迭代。

上面的运算取了14个r值，每一个r值所对应迭代序列的“极限”如图所示，注意在本次计算中，对每个r值做了N=150次代，仅画出了最后的50个代值. 由图可见：
(1) 当r分别等于0，0.3，0.6，0.9，1.2，1.5，1.8，2.1，2.4时，每个r对应的这50个迭代值均重合为一占，这说明对这些r值极限存在；
(2) 当r分别等于3.0和3.3时，迭代值对应着两个点，这说明极限不存在；
(3) 当r等于3.6和3.9时，出来更多的点，这种情况显然有别于情形(1)和(2).
分别选取代表这三种类型的r值用蛛网图进行观察.



实验2：迭代法之不动点迭代

简单和复杂：二次函数的迭代和混沌

步骤2. 用蛛网图观察三种不同类型的迭代。

在图(a)中，迭代序列收敛到一个不动点$x = f(x)$，我们用$x \to x$来表示这件事情；

实验2：迭代法之不动点迭代

简单和复杂：二次函数的迭代和混沌

步骤2. 用蛛网图观察三种不同类型的迭代。

在图(b)的情况，迭代序列呈现出周期性：
$x_{2k+1} \to x_1^*, x_{2k} \to x_2^*$；
如果迭代点具有性质：$f(x_1) = x_2, f(x_2) = x_3, \cdots, f(x_{k-1}) = x_k, f(x_k) = x_1$，则$x_1, x_2, \cdots, x_k$形成了一个**k循环**. 我们用$x_1 \to x \to \cdots \to x_k$来表示这件事，并说$x_1$是一个k周期点，$x_1, x_2, \cdots, x_k$为一个周期轨道. 在图中，迭代点列的前18个值为

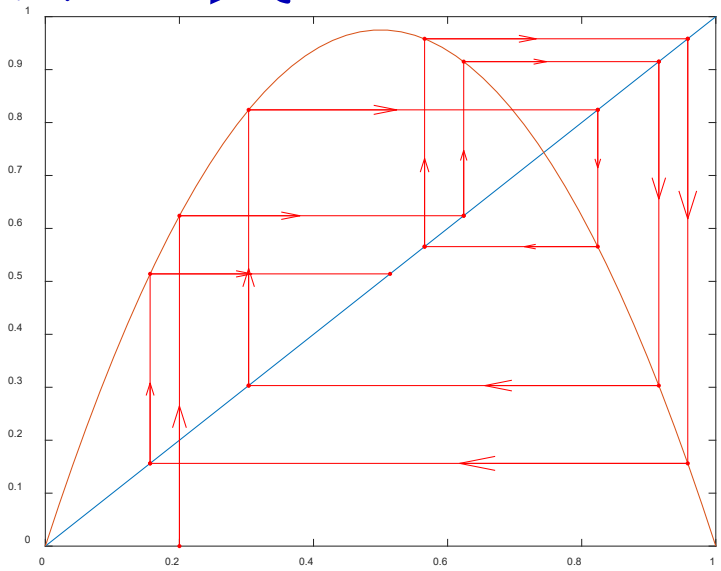
0.5120 0.7995 0.5129 0.7995 0.5130
0.7995 0.5130 0.7995 0.5130 0.7995
0.5130 0.7995 0.5130 0.7995 0.5130
0.7995 0.5130 0.7995

显然0.5130→0.7995→0.5130构成了一个2循环. 所以迭代序列形成了一个2周期轨道.

实验2：迭代法之不动点迭代

简单和复杂：二次函数的迭代和混沌

步骤2. 用蛛网图观察三种不同类型的迭代。

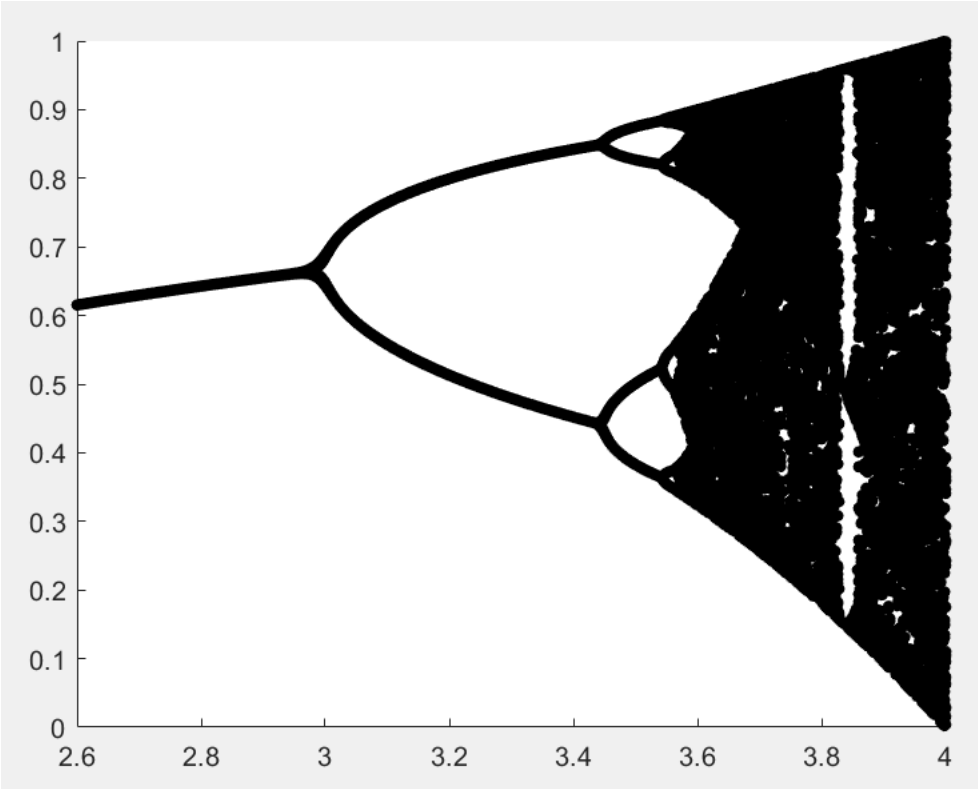


在图(c)和(d)的情况，迭代序列既不收敛也不呈现出周期性，而是反映出一种十分混乱的混沌现象：轨道不再是趋向任何稳定的周期轨道，而似乎随机的在(0,1)区间(或在其中的某些子区间)中跳来跳去，但这一轨道在这些区间内的任何一个子区间(a,b)内都会出现无数次，这就是混沌的遍历性.

实验2：迭代法之不动点迭代

简单和复杂：二次函数的迭代和混沌

步骤3. 加密r的取值，得到Feigenbaum（费根鲍姆）图，在Feigenbaum图中你看到了什么？上面说的三种情况是怎样表现在这一图中的？适当修改观察程序Exp7_4.m中的参数和某些绘图属性值，对r=2.6:0.001:4重新计算，运行结果显示在图中，这个图形叫做**Feigenbaum图**.



实验2：迭代法之不动点迭代

简单和复杂：二次函数的迭代和混沌

我们看到，随着r的增加，图中从开始的一条曲线(不动点)分裂为两条曲线(2周期循环)、4条曲线(4周期循环)和8条曲线(8周期循环)，这种现象叫**倍周期现象**.

在8条曲线曲线之后是否是16条曲线→32条曲线等等，由于尺度很小的原因已经很难看清了.

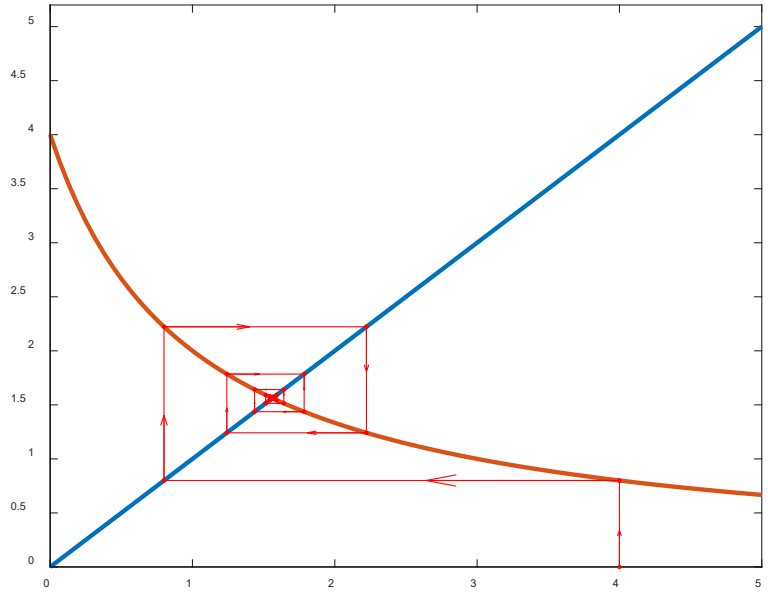
如果编制一个横向放大镜程序做进一步观察.例如设定一个适当的横向放大比例c,使得可以在一个很小的r的取值区间段观察Feigenbaum图.观察程序Exp7_4.m

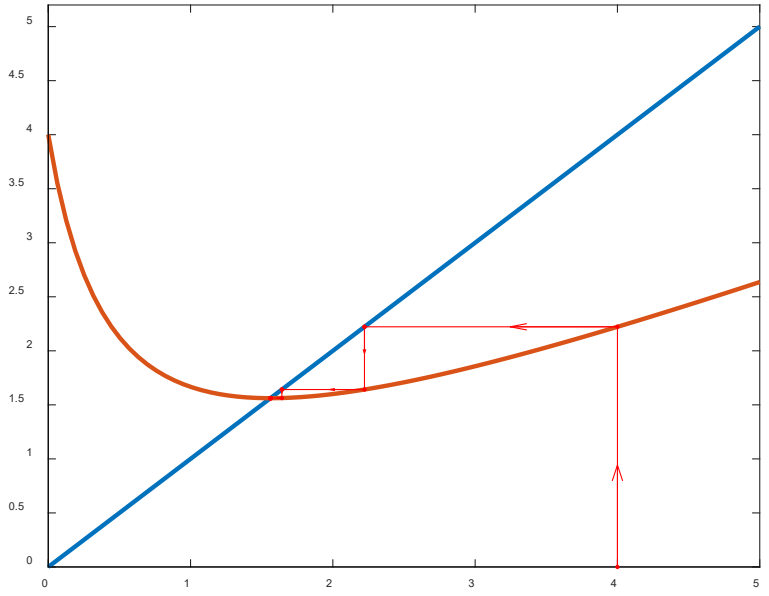
实验2：迭代法之不动点迭代

如何直观的理解收敛快慢？

函数越平坦，迭代越快吗？

通过对实例观察和分析，找到问题的规律，是数学实验的重要方法之一. 在上面的实验观察中，对比图g2(b)和图g3(c)，我们看到在不动点附近，后者的曲线图形要比前者平坦. 这是否是影响迭代收敛速度的内在原因？你能否构造一些更为平坦的函数做进一步的观察呢？如果你想从理论上进行分析，首先需要考虑的问题是不是如何描述一个函数在某点附近的平坦性？





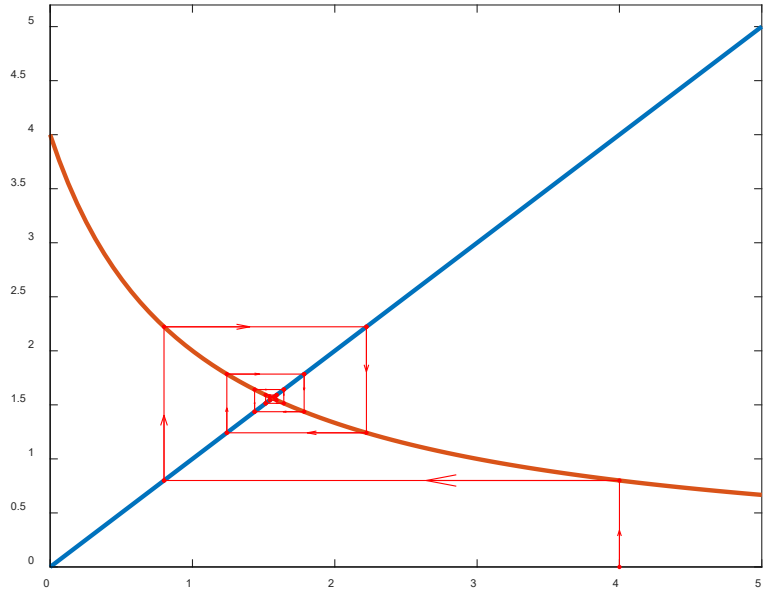
实验2：迭代法之不动点迭代

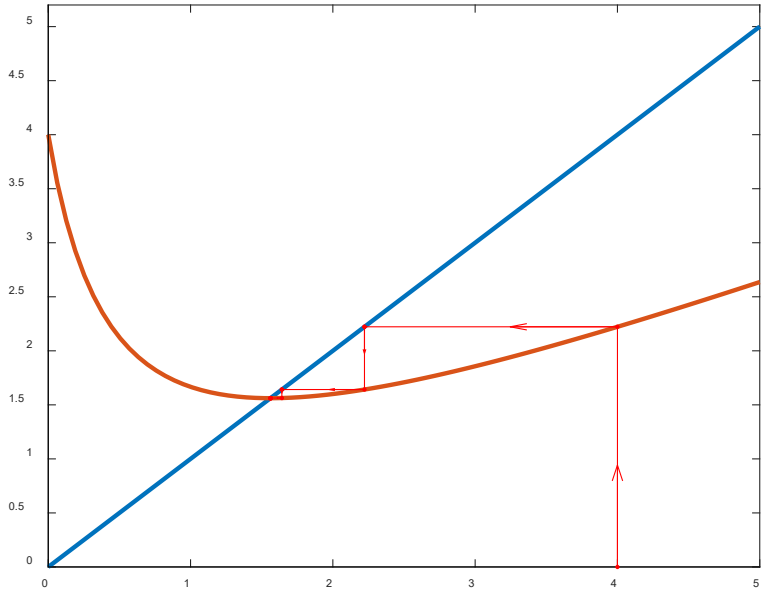
如何直观的理解收敛快慢？

函数越平坦，迭代越快吗？

切线的斜率也就是导数可以用来描述曲线的平坦程度，如果$f'(x^*)=0$，则函数f在x^*处有水平切线，也就是最平坦了.根据上面的观察，可以猜想到如下结论：一般，若函数f具有连续导数，则$|f'(x^*)|$越小，函数f在不动点x^*附近就越平坦，从而收敛速度也就越快.

这与我们的理论分析相一致。





实验2：迭代法之不动点迭代

如何构造迭代函数使之具有较快的收敛速度？

通把方程$f(x)=0$的求根转化为求不动点问题来处理，可以从不动点方程$g(x)=x$自然地引出迭代式$x_{n+1}=g(x_n)$. 同时通过上面的讨论我们知道，函数f的构造是十分重要的，不同构造方法将导致不同的收敛性(收敛或发散)和收敛速度. 例如, 对g_1的迭代是发散的；而对g_2、g_3的迭代是收敛的，但两者的收敛速度又不同. 当初始点靠近不动点时，如果迭代函数f在不动点附近的导数绝对值较小，收敛的可能性就较大，收敛的速度也就较快.

实验2：迭代法之不动点迭代

如何构造迭代函数使之具有较快的收敛速度？

从前面的实验观察中知道，使得迭代序列收敛并尽快收敛到方程f(x)=0的某一解的条件是迭代函数g(x)在解的附近的导数的绝对值尽量小. 这启发我们将迭代方程修改成$x = h(x) = \lambda g(x) + (1 - \lambda)x.$ (4)

我们需要选取λ使得$|h'(x)|$在解的附近尽量小. 为此，我们可以令
$h'(x) = \lambda g'(x) + 1 - \lambda = 0,$

得$\lambda = \frac{1}{1-g'(x)}.$

于是$h(x) = x - \frac{g(x)-x}{g'(x)-1}.$

特别地，如果取$g(x) = f(x) + x$, 则我们得到迭代公式

$x_{n+1} = x_n - \frac{f(x_n)}{f'(x_n)}, n = 0,1,\cdots,$ (5)

实验3：迭代法之牛顿迭代

牛顿迭代公式

$$x_{n+1} = x_n - \frac{f(x_n)}{f'(x_n)}, n = 0,1,\cdots, \qquad (5)$$

能否给出上述改进公式的几何解释？使用改进的迭代公式求方程的根.将它的收敛速度与你得到的其他的迭代公式相比较，哪个更快？

实验3：迭代法之牛顿迭代

牛顿迭代公式

$$x_{n+1} = x_n - \frac{f(x_n)}{f'(x_n)}, n = 0,1,\cdots, \qquad (5)$$

能否给出上述改进公式的几何解释？使用改进的迭代公式求方程的根. 将它的收敛速度与你得到的其他的迭代公式相比较，哪个更快？

设x_0为$f(x)$上一点，则过x_0的切线方程为

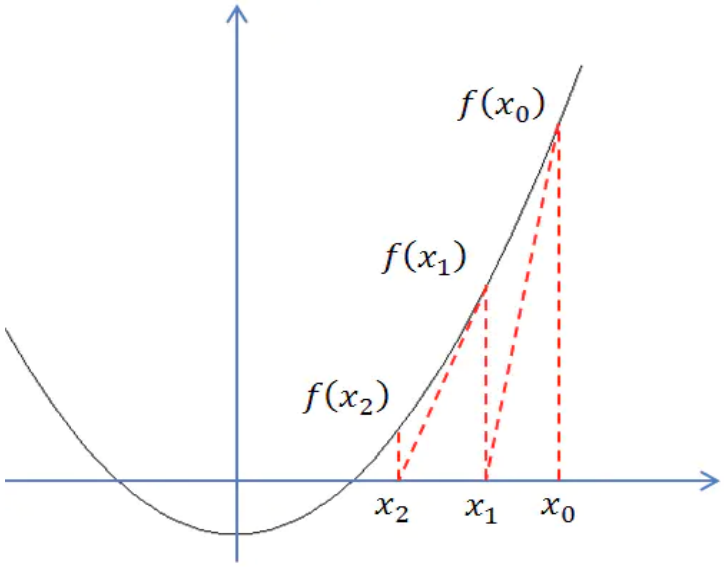
$$y = f(x_0) + f'(x_0)(x - x_0)$$

式中：令y=0，得到切线与x轴交点为

$$x = x_0 - \frac{f(x_0)}{f'(x_0)}$$

根据式(2.9)构造迭代关系式

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



收敛的理论条件？

实验3：迭代法之牛顿迭代

牛顿迭代公式

$$x_{n+1} = x_n - \frac{f(x_n)}{f'(x_n)}, n = 0,1,\cdots, \qquad (5)$$

令$\varphi(x) = x - \frac{f(x)}{f'(x)}$，则

$$\varphi'(x) = 1 - \frac{\left(f'(x)\right)^2 - f(x)f''(x)}{\left(f'(x)\right)^2} = \frac{f(x)f''(x)}{(f'(x))^2}$$

在$x = x^*$处，有

$$\varphi'(x^*) = \frac{f(x^*)f''(x^*)}{(f'(x^*))^2} = 0$$

因此在$x = x^*$充分小的邻域内有$|\varphi'(x)| \le q \le 1$，则牛顿迭代法收敛。

实验3：迭代法之牛顿迭代

对初值敏感性

f(x) = x^3 - 2*x^2 - 11*x + 12 = 0

2.35287527 converges to 4;

2.35284172 converges to -3;

2.35283735 converges to 4;

2.352836327 converges to -3;

2.352836323 converges to 1.

实验3：迭代法之牛顿迭代

吸引域（basin of attraction）：

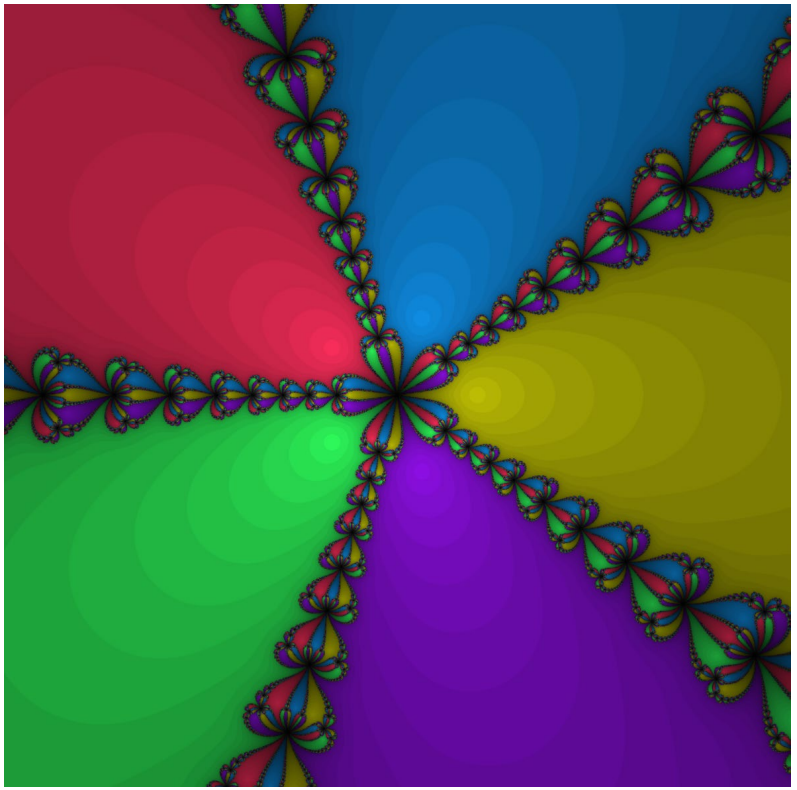
When dealing with complex functions, Newton's method can be directly applied to find their zeroes. Each zero has a basin of attraction in the complex plane, the set of all starting values that cause the method to converge to that particular zero.

Basins of attraction for

x^5 - 1 = 0;

darker means more iterations

to converge.



实验3：迭代法之牛顿迭代

示例 使用牛顿迭代求方程$x^3+x-1=0$在$[0,1]$内的近似根。

写出格式

程序Exp7_6.m

实验3：迭代法之牛顿迭代

讨论：比较优缺点

- 牛顿法
- 基本不动点迭代
- 二方法

实验4：迭代法之弦截法

牛顿迭代法的每一步不但要计算函数值，而且要计算导数值，在使用过程中有一定的局限性。特别是遇到以下情况：①导数计算比较困难；②在根的领域内导数值非常小；③方程有多重根，牛顿迭代法可能失效。对

$$x_{n+1} = x_n - \frac{f(x_n)}{f'(x_n)}, n = 0,1,\cdots,$$

使用割线代替切线，若取$f'(x_n) = \frac{f(x_n)-f(x_0)}{x_n-x_0}$

可得单点弦截法： $x_{n+1} = x_n - \frac{f(x_n)(x_n-x_0)}{f(x_n)-f(x_0)}$

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

可得两点弦截法： $x_{n+1} = x_n - \frac{f(x_n)(x_n-x_{n-1})}{f(x_n)-f(x_{n-1})}$

实验4：迭代法之弦截法

实验练习：

使用两点弦截法求方程$x^3-3x-1=0$在$x_0=2$内的近似根（误差<10^{-5}，取$x_0=2$，$x_1=1.9$）。

分析，写出格式

程序Exp7_7.m

命令窗口执行:

```
>>format long
>>[result,n]=Exp7_7(1e-5,100,2,1.9)
result=
2.000000000000000 1.900000000000000 1.881093935790725
1.879411060169918 1.879385274283925 1.879385241572444
```

上述迭代过程表明：使用两点弦截法，迭代n-2=4步，即可达到误差限内近似解x=1.879385241572444。

实验5：求解方程组的迭代法

不动点迭代：

$$\boldsymbol{F(x)=0} \quad \text{恰当地变形} \rightarrow \quad \boldsymbol{x=G(x)}$$

基本格式：$\boldsymbol{x_{n+1}=G(x_n)}$

以线性方程组 $\boldsymbol{Ax=b}$ 为例：

- Jacobi method:

Then A can be decomposed into a diagonal component D, a lower triangular part L and an upper triangular part U:

$$A = D + L + U \qquad \text{where} \qquad D = \begin{bmatrix} a_{11} & 0 & \cdots & 0 \\ 0 & a_{22} & \cdots & 0 \\ \vdots & \vdots & \ddots & \vdots \\ 0 & 0 & \cdots & a_{nn} \end{bmatrix} \text{ and } L + U = \begin{bmatrix} 0 & a_{12} & \cdots & a_{1n} \\ a_{21} & 0 & \cdots & a_{2n} \\ \vdots & \vdots & \ddots & \vdots \\ a_{n1} & a_{n2} & \cdots & 0 \end{bmatrix}.$$

The solution is then obtained iteratively via

$$\mathbf{x}^{(k+1)} = D^{-1}(\mathbf{b} - (L + U)\mathbf{x}^{(k)}),$$

where $\mathbf{x}^{(k)}$ is the kth approximation or iteration of $\mathbf{x}$ and $\mathbf{x}^{(k+1)}$ is the next or $k + 1$ iteration of $\mathbf{x}$. The element-based formula is thus:

$$x_i^{(k+1)} = \frac{1}{a_{ii}} \left(b_i - \sum_{j \neq i} a_{ij} x_j^{(k)} \right), \quad i = 1, 2, \ldots, n.$$

实验5：求解方程组的迭代法

不动点迭代：

$$\boldsymbol{F(x)=0} \quad \text{恰当地变形} \rightarrow \quad \boldsymbol{x=G(x)}$$

基本格式：$\boldsymbol{x_{n+1}=G(x_n)}$

以线性方程组 $\boldsymbol{Ax=b}$ 为例：

- Gauss-Seidel method:

Then the decomposition of A into its lower triangular component and its strictly upper triangular component is given by:

$$A = L_* + U \qquad \text{where} \qquad L_* = \begin{bmatrix} a_{11} & 0 & \cdots & 0 \\ a_{21} & a_{22} & \cdots & 0 \\ \vdots & \vdots & \ddots & \vdots \\ a_{n1} & a_{n2} & \cdots & a_{nn} \end{bmatrix}, \quad U = \begin{bmatrix} 0 & a_{12} & \cdots & a_{1n} \\ 0 & 0 & \cdots & a_{2n} \\ \vdots & \vdots & \ddots & \vdots \\ 0 & 0 & \cdots & 0 \end{bmatrix}.$$

The system of linear equations may be rewritten as:

$$L_*\mathbf{x} = \mathbf{b} - U\mathbf{x}$$

The Gauss–Seidel method now solves the left hand side of this expression for **x**, using previous value for **x** on the right hand side. Analytically, this may be written as:

$$\mathbf{x}^{(k+1)} = L_*^{-1}\left(\mathbf{b} - U\mathbf{x}^{(k)}\right).$$

However, by taking advantage of the triangular form of L_*, the elements of $\mathbf{x}^{(k+1)}$ can be computed sequentially using forward substitution:

$$x_i^{(k+1)} = \frac{1}{a_{ii}}\left(b_i - \sum_{j=1}^{i-1} a_{ij}x_j^{(k+1)} - \sum_{j=i+1}^{n} a_{ij}x_j^{(k)}\right), \quad i = 1, 2, \ldots, n.$$ [5]

实验5：求解方程组的迭代法

Newton’s method for Systems of equations

$$\mathbf{x}_{n+1} = \mathbf{x}_n - J_F(\mathbf{x}_n)^{-1} F(\mathbf{x}_n)$$

Rather than actually computing the inverse of the Jacobian matrix, one may save time and increase numerical stability by solving the system of linear equations

$$J_F(\mathbf{x}_n)(\mathbf{x}_{n+1} - \mathbf{x}_n) = -F(\mathbf{x}_n)$$

实验6：求根有关的Matlab内置函数

在MATLAB中，主要有solve、fzero、fsolve、roots等函数用于方程（组）求解，下面介绍其详细用法。

1.solve函数solve函数可用来求解代数方程（组）与非线性方程（组），具体使用格式：solve（'F'，'var'）：用于求解单个方程情形，F表示求解方程，var表示求解变量，当求解变量省略时，表示对默认变量求解，若方程为符号方程，求解变量为符号变量时，上述格式中的单引号可省略；
[x1,x2,…,xn]=solve('F1','F2',…,'Fn','var11','var2',…,'varn'):用于求解n个方程组成的方程组问题，F1、F2、…、Fn表示各个方程，var1，var2，…，varn表示各个求解变量，[x1，x2，…，xn]表示求解结果。

实验6：求根有关的Matlab内置函数

例 求一元二次方程$ax^2+bx+c=0$的根。

```
>>syms x a b c
>>ff=a*x^2+b*x+c;
>>solve(ff)
ans=
-1/2*(b-(b^2-4*a*c)^(1/2))/a
-1/2*(b+(b^2-4*a*c)^(1/2))/a
```

实验6：求根有关的Matlab内置函数

2.fzero函数

fzero函数用于求非线性方程的最优解，使用格式：fzeno(F,[a,b]):F表示求解的方程,一般通过子程序建立F,[a,b]表示求解区间，该格式寻求F在[a,b]内的根；

fzero(‘F‘,x0):F表示解的方是建立方式同上，x0表示迭代初值。

实验6：求根有关的Matlab内置函数

例 求方程$x+2\sin xe^x-1=0$的根。

先在文件编辑窗口编写如下M文件，并保存在当前工作目录下。

```
function f=exam2_10(x)
f=x+2*sin(x)*exp(x)-1;
```

然后在命令窗口中执行:

```
>>fzero(@exam2_10,[0,1])
ans=0.2774
>>fzero(@exam2_10,0.8)
ans=
0.2774
```

实验6：求根有关的Matlab内置函数

例 求方程$x+2\sin xe^x-1=0$的根。

对于方程的建立，也可以通过@方式也可以建立函数，执行如下:

```
>>test=@(x)[x+2*sin(x)*exp(x)-1];
>>fzero(test,0.8)
ans=
0.2774
```

实验6：求根有关的Matlab内置函数

3.fsolve函数

fsolve函数用于求非线性方程组的最优解，使用格式：
fsolve('F ‘,x0):F表示求解的方程组，一般通过子程序建立F，x0表示迭代初值。

实验6：求根有关的Matlab内置函数

例 求方程组$\begin{cases} x_1 - \sin x_1 - x_2 = 0 \\ 2x_1 + x_2 - \cos x_2 = 0 \end{cases}$的解。

先在文件编辑窗口编写如下M文件，并保存在当前工作目录下。

```
function F=exam2_11(x)
F(1)=x(1)-sin(x(1))-5*x(2);
F(2)=2*x(1)+x(2)-cos(x(2));
```

然后在命令窗口中执行如下语句

```
>>fsolve(@exam2_11,[0,0])
Optimization terminated:first-order optimality is less than options.TolFun.
ans=
0.4980  0.0041
```

实验6：求根有关的Matlab内置函数

4. roots函数

roots函数用于求多项式方程的根，使用格式：

roots（A)：表示在复数范围内求多项式方程$a_n x^n + a_{n-1} x^{n-1} + \cdots a_1 x + a_0 = 0$的所有根，其中A=[$a_n, a_{n-1}, \cdots a_1, a_0$]。

实验6：求根有关的Matlab内置函数

例 求方程$x^5-6x^4+5x^3-3x+8=0$的根。

```
>>a=[1,-6,0,5,-3,8];
>>roots(a)
ans=
5.8626
-1.2901
1.2681
0.0797+0.9098i
0.0797-0.9098i
```

实验7：求根的可视化方法

图形放大法

在科学研究与工程计算过程中，有些问题只需要大致确定方程根的取值范围，并不需要计算精确解；有些问题需要求精确解，但无解析表达式，而设计的迭代方法对初值具有敏感性。对上述两个问题，可以通过图形放大的方法，确定根的取值范围或找到根值附近的初始迭代点，具体步骤如下：

(1)绘制函数曲线。
(2)图形放大，确定曲线与x轴的交点或两曲线的交点。

实验7：求根的可视化方法

例1 使用图形放大法确定方程$x^5-6x^4+5x^3-3x+8=0$各根的取值范围。

```
>>x=-2:0.01:6;
>>y=x.^5-6*x.^4+5*x.^2-3*x+8;
>>plot(x,y)
>>grid on
```

由图可知，三个实根大致取值范围为

(-1.4,-1.2),(1.25,1.3),(5.8,5.9)。

实验7：求根的可视化方法

例2 使用图形放大法确定方程组$\begin{cases} 3x^2 - y^3 = 1 \\ e^{-x} - y = -2 \end{cases}$解的取值范围。

```
>>ezplot('3*x^2-y3-1',[-7,7,-7,7])
>>hold on
>>ezplot('exp(-x)-y+2',[-7,7,-7,7])
>>grid on
```

由图可知，方程组的解的大致范围为1.8<x<2,2.1<y<2.2。

11月8日实验课实验题

随机模拟实验
上交截止日期:
2023年11月8日23:00

- 实验课实验题1：

用所给迭代法求方程 $x^3-\sin x-12x+1=0$ 全部根。

(a) $x=\varphi(x)=\sqrt[3]{12x+\sin x-1},\quad -4\le x\le 3.$

(b) $x=\varphi(x)=\dfrac{1}{12}(x^3-\sin x+1),\quad 0\le x\le 0.1.$

(c) $x=\varphi(x)=\sqrt[3]{12x+\sin x-1},\quad 3\le x\le 4.$

并绘制迭代过程可视化的蛛网图。

- **实验课实验题2：**

设有方程组 $Ax=b, A\in R^{20\times 20}$，其中A是一个稀疏的五对角矩阵：

$$A=\begin{pmatrix} 3 & 1 & 1 & & \\ 1 & 3 & 1 & \ddots & \\ 1 & 1 & \ddots & \ddots & 1 \\ & \ddots & \ddots & \ddots & 1 \\ & & 1 & 1 & 3 \end{pmatrix},$$

随机取一个非零的右端向量b。分别采用Jacobi迭代和Gauss-Seidel迭代两种方法求解上述线性方程组，画出误差的收敛历史图，观察和分析计算结果。

实验课实验题3：

考虑以下问题并编程实现

（1）构造三种基本迭代法求 $f(x)=3x^2-e^x=0$ 的根。

（2）用恰当的方法求方程 $x^9-522+e^x=0$ 在 1.9 附近的根。

（3）设多项式 $p(x)=(x-1)(x-2)\cdots(x-20)$，取多个非常小的 ε，解方程 $p(x)+\varepsilon x^{19}=0$，并对结果进行分析。

注：

对于第(2)问，要求计算出的根精确到小数点后13位，并给出理论依据。

对于第(3)问，假如不使用roots函数，你还有其它方法求出所有的根吗？（鼓励提出其它的方法）