数学实验讲义

上册

东南大学高等数学教研室

2019 年 9 月

实验一　　观察数列的极限

极限是高等数学中最基本的概念之一，初学者往往理解不够准确。本实验目的就是利用数学软件 Mathematica 加深对数列极限概念的理解 。

对于数列极限通俗的说法是：当n充分大时，a_n充分接近数 A，则$\lim\limits_{n\to\infty}a_n=A$。我们通过利用数学软件 Mathematica 来计算数列$\{a_n\}$足够多项的值，从而考察数列的极限。

例1　用数、形结合的方法观察极限$\lim\limits_{n\to\infty}n\sin\dfrac{1}{n}=1$。

解：通过逐渐增加点并画点图，来观察当 n 越来越大时$a_n=n\sin\dfrac{1}{n}$的变化趋势，我们输入 Mathematica 语句：

```
In[1]:= an = {Sin[1], 2 Sin[1 / 2], 3 Sin[1 / 3]};
        Do[an = Append[an, i * Sin[1 / i]];
         t = ListPlot[an, PlotRange → {0, 2},
           PlotStyle → PointSize[0.015]]; Print[t],
         {i, 4, 20}
        ]
```

运行后得到了 16 幅图，图 1-1 中列出了其中的 4 幅，从左至右图中点数逐渐增多，从图中可以看出所画出的点逐渐接近于直线$x=1$。

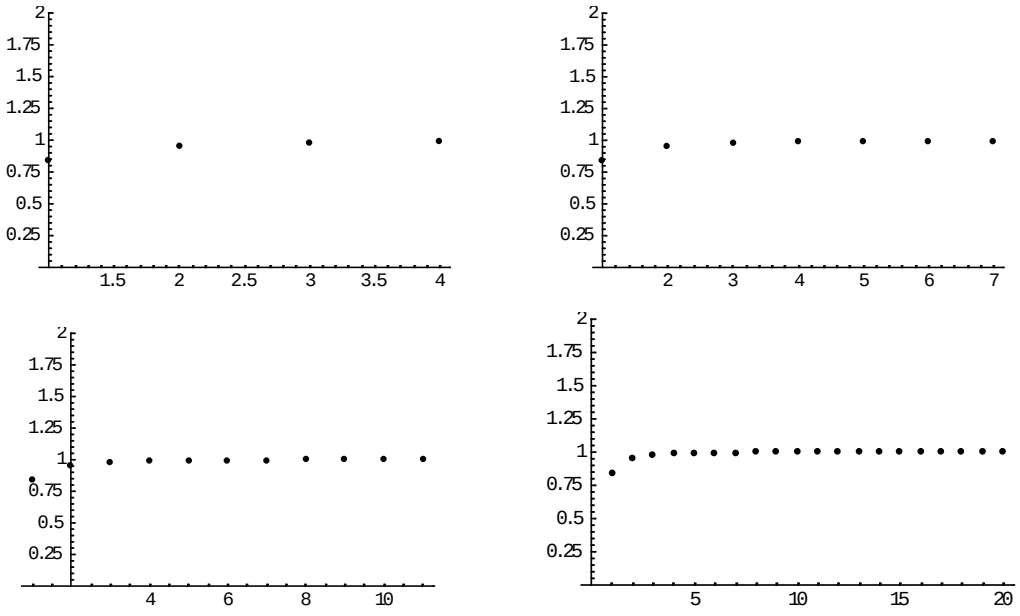


图 1-1

例 2. 为了引入极限的概念，介绍了非常著名的实例——割圆术，即当圆的内接正多边形的数目无限增加的时候，正多边形的面积会接近圆的面积。让同学们设计数学实验，来绘出圆

内接不同边数的正多边形，从图形中来理解圆的面积由多边形面积逼近的可行性。在 Mathematica 中输入循环语句：

```
In[3]:= Do[
          v1 = Graphics[{Thickness[0.01], RGBColor[0, 0, 0],
             Circle[{0, 0}, 1]}];
          v2 = Graphics[{Thickness[0.01], RGBColor[0, 0, 1],
             Line[Table[{Cos[2 * Pi * (i / n)], Sin[2 * Pi * (i / n)]},
               {i, 0, n}]]}];
          t = Show[v1, v2]; Print[t], {n, 3, 24, 6}]
```

运行后得到四幅图，分别为圆内接正三边形、正九边形、正十五边形和正十八边形（如图 1-2）。

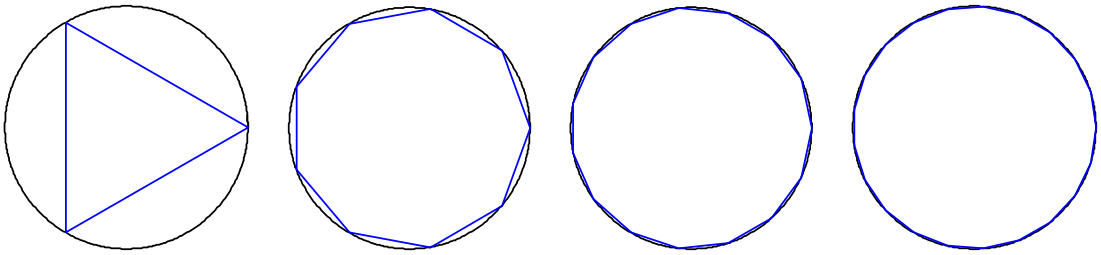


图　1-2

除了从图形上来观察逼近可行性以外，也可以通过软件计算单位圆的内接正多边形的面积，可以从数据上观察出当正多边形的边数越来越大时，其面积越来越接近圆周率 π。取单位圆，即半径为1的圆，首先考虑内接正三角形，然后每次将正多边形的边剖分为二，剖分 n 次之后，得到了圆的内接正 $3\cdot 2^{n-1}$ 边形，其面积为 $S_n = 3\cdot 2^{n-1}\cdot\frac{1}{2}\cdot\sin\frac{2\pi}{3\cdot 2^{n-1}}$。我们利用 Mathematica 编程如下：

```
In[4]:= R = 1;
        For[n = 1, n < 21, n++,
         sn = N[3 * 2 n - 2 * R^2 * Sin[2 * Pi / (3 * 2 n - 1)], 20];
         delta = N[Pi - sn, 20];
         Print["圆内接正", 3 * 2 n - 1, "边形的面积为", sn, "    ",
          "与圆周率的误差为", delta]]
```

运行后输出了 20 组数据，可以发现，当 n 越来越大时，正多边形的面积与圆周率误差越来越小。下面列举了最后一项输出的结果：

```
圆内接正119边形的面积为119.89444930573907526
    与圆周率的误差为-116.75285665214928202
```

例 3. 设数列$\{x_n\}$与$\{y_n\}$由下式确定：$\begin{cases} x_1=1,\ \ y_1=2 \\ x_{n+1}=\sqrt{x_n y_n} & n=1,2,\cdots \\ y_{n+1}=\dfrac{x_n+y_n}{2} & n=1,2,\cdots \end{cases}$，观察数列$\{x_n\}$与$\{y_n\}$的极限是否存在。

解：输入以下语句可进行观察，此程序的功能是输出$\{x_n\}$与$\{y_n\}$的前 10 项数值。大家可改变 For 循环中终结语句（$n\le 10$）来改变输出项的项数。

```
In[6]:= f[x_, y_] := Sqrt[x*y]; g[x_, y_] := (x+y)/2; xn = 1; yn = 2;
        For[n = 2, n ≤ 10, n++, xN = xn; yN = yn; xn = N[f[xN, yN]];
          yn = N[g[xN, yN]]; Print[xn, "        ", yn]];
        Print["x10=    ", xn, "    y10=", yn]
```

运行该程序可得：

```
1.41421      1.5
1.45648      1.45711
1.45679      1.45679
1.45679      1.45679
1.45679      1.45679
1.45679      1.45679
1.45679      1.45679
1.45679      1.45679
1.45679      1.45679
x10=   1.45679    y10=1.45679
```

可以由运行结果可观察到，$\{x_n\}$与$\{y_n\}$均有极限，且这两极限值是相等的。

实验习题 1

1、 根据上面的实验步骤，通过作图，观察重要极限：$\lim\limits_{n\to\infty}(1+\dfrac{1}{n})^n=e$。

2、 设数列$\{x_n\}$由下列递推关系式给出：$x_1=\dfrac{1}{2}, x_{n+1}=x_n{}^2+x_n\ \ (n=1,2,\cdots)$，观察数列

$\dfrac{1}{x_1+1}+\dfrac{1}{x_2+1}+\cdots+\dfrac{1}{x_n+1}$的极限。

实验二　方程的近似解

由连续函数的零点定理可知，若连续函数 $f(x)$ 在区间 $[a,b]$ 的两个端点处异号，则函数 $f(x)$ 在区间 $[a,b]$ 内必有零点，也即方程 $f(x)=0$ 必有根。在科学研究和工程技术问题中，常会遇到求解高次代数方程或其它类型的方程问题，由于求这类的方程精确解很困难，因此需要求方程的近似解。本实验的目的是介绍方程近似求根的方法，并利用 Mathematica 软件来实现算法。

1、 从函数图形观测根的大概位置

前面介绍了一元函数的图形的画法，我们知道方程 $f(x)=0$ 的实根在几何上表示曲线 $y=f(x)$ 与 x 轴交点的横坐标，因此可以从函数的图形上观测根的大概位置，我们称之为图形法。

采用 Mathematica 软件来实现图形法，则先利用作图命令，选定变量 x,y 的取值范围（y 的范围一般可由系统默认），第一次作图可把 x 的范围取得略为大点，当看出曲线 $y=f(x)$ 的图形与 x 轴的交点位于哪个子区间后，再在这个子区间内进行第二次作图，如此继续下去，每一次作图，便能获得更小的子区间，最后能将区间的端点作为根的近似值，它与方程的精确解的误差不超过最后那个子区间的长度。

例1　用图形法求方程 $4x^3-6x^2+3x-2=0$ 的近似解，要求误差不超过 10^{-4}。

解：（1）首先定义函数 $f(x)=4x^3-6x^2+3x-2$，由 $f(1)=-1, f(2)=12$ 知，在区间 $[-1,2]$ 内有根，因此先在区间 $[-1,2]$ 上作出函数的图形，输入命令：

```
In[1]:= f[x_] := 4 x^3 - 6 x^2 + 3 x - 2; Plot[f[x], {x, 1, 2}]
```

运行结果如图 2-1。从图形上可以看到函数在区间 $[-1,2]$ 内与 x 轴交于一点，交点在区间 $[1.2,1.4]$ 内，而且在这个区间的左半部，因此我们第二次画图选定区间为 $[1.2,1.3]$。并且从图形中可以知道，函数在区间 $[-1,2]$ 内只有唯一的根，把这样的区间称为**隔断区间**。

（2）在区间 $[1.2,1.3]$ 上再作函数的图形，并取因变量的显示区间为 $[-0.1,0.1]$，则键入

```
In[2]:= Plot[f[x], {x, 1.2, 1.3}, PlotRange → {-0.1, 0.1}]
```

运行结果如图 6-2。观察图后我们可以把第三次作图的隔断区间选定为 $[1.22,1.23]$。

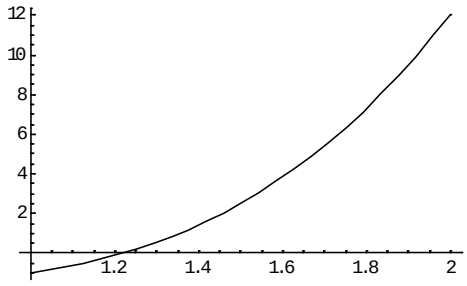


图 2-1

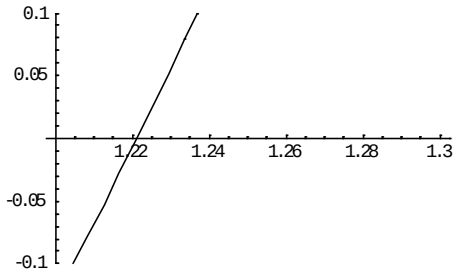


图 2-2

（3）在区间[1.22,1.23]上作函数的图形，并且以后依次选择隔断区间作图，输入命令如下：

```
In[3]:= Plot[f[x], {x, 1.22, 1.23}, PlotRange → {-0.01, 0.01}]
        Plot[f[x], {x, 1.221, 1.222}, PlotRange → {-0.001, 0.001}]
        Plot[f[x], {x, 1.2211, 1.2212}, PlotRange → {-0.0001, 0.0001}]
```

运行结果如图 2-3 所示。

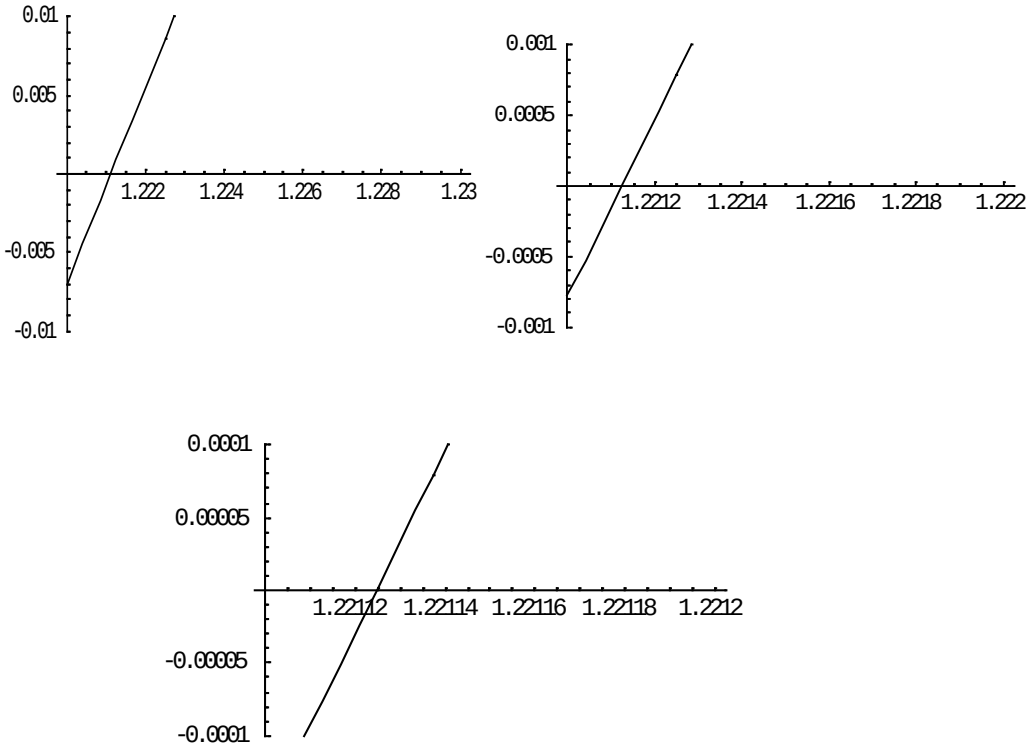


图 2-3

观察上述图可知，函数在区间[1.2211,1.2212]内有根，我们可以选择该区间的端点 1.2211 作为所求方程的近似解，且它与精确解的误差不会超过 0.0001。

2、用二分法求根

设函数 $f(x)$ 在区间$[a,b]$上连续，且在两个端点上异号，二分法是逐次将区间减半来

产生一个闭区间套$[a,b]\supset[a_1,b_1]\supset[a_2,b_2]\cdots\supset[a_n,b_n]\supset\cdots$，使得每个小区间都包含了$f(x)=0$的根。由于$|a_n-b_n|=\dfrac{|a-b|}{2^n}$，所以该区间套的端点数列$\{a_n\}$和$\{b_n\}$都收敛于根，则只要当$|a_n-b_n|$小于给定的正数$\delta$，就可以用$a_n$或$b_n$作为根的近似值，且误差的绝对值不会大于给定的正数$\delta$。

算法框图如图 2-4 所示，我们以实例来用 Mathematica 实现算法。

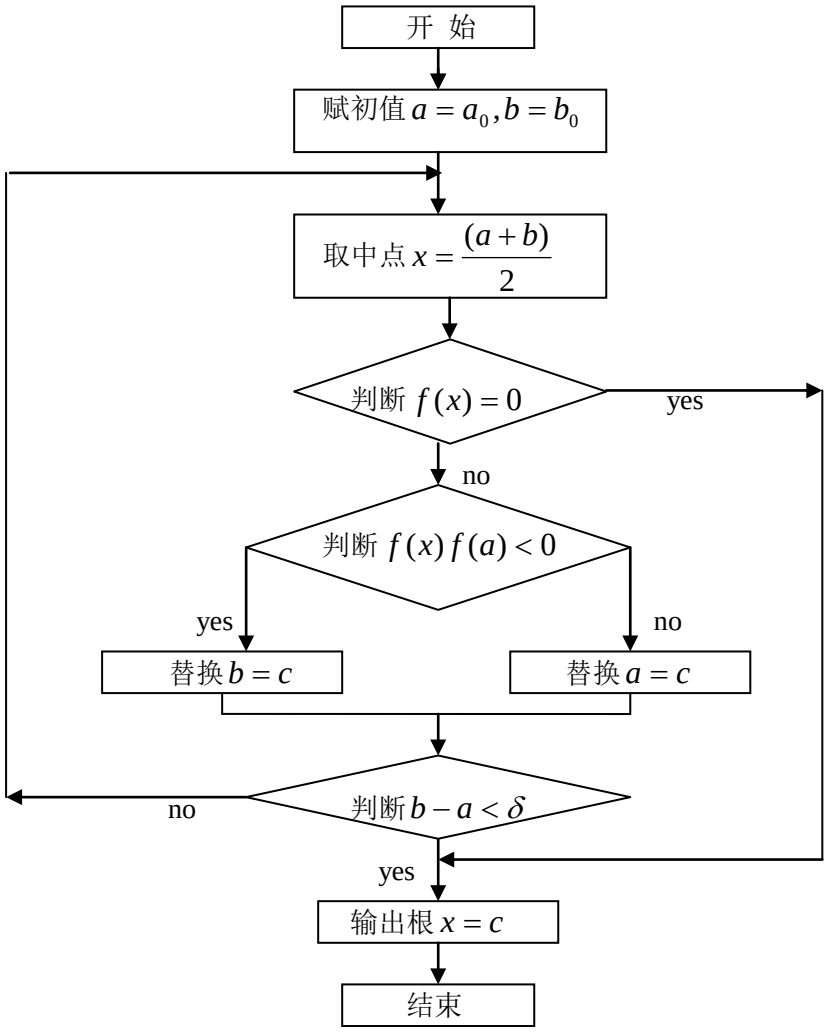


图 2-4　二分法的算法框图

例2　用二分法求方程$4x^3-6x^2+3x-2=0$的近似解，要求误差不超过10^{-6}。

解：根据二分法的算法，输入 Mathematica 程序如下，在程序中，`k0` 表示最大循环次数，如果循环次数已到达 `k0` 却还没有达到预定的误差要求，则显示“`fail`”。

```
In[6]:= f[x_] := 4 x^3 - 6 x^2 + 3 x - 2;
        a0 = 1; b0 = 2; dalta = 10^(-6); k0 = 25; a = a0; b = b0;
        Do[x = (a + b) / 2; Print[N[x, 17], "     n=", k];
         If[f[x] == 0, Break[], If[f[x] * f[a] < 0, b = x, a = x]];
         If[Abs[b - a] < dalta, Break[], If[k == k0, Print["fail"]]],
         {k, k0}]
```

从输出结果可知，迭代到第 20 次结束程序，得到近似解为 $x=$ 1.2211256027221680。

3、 用切线法（Newton 法）求根

如果函数 $f(x)$ 在区间 $[a,b]$ 上有二阶导数，且满足 $f(a)f(b)<0$ 且 $f'(x)$ 和 $f''(x)$ 都在区间 $[a,b]$ 上保持定号，则方程 $f(x)=0$ 在区间 $[a,b]$ 内有惟一的实根。

如果函数 $f(x)$ 在隔断区间 $[a,b]$ 内满足如上条件，则可采用切线迭代法（也称牛顿迭代法）来求方程 $f(x)=0$ 的近似解，其原理是用曲线弧一端点处的切线代替曲线弧，从而以切线与 x 轴的交点的横坐标作为方程实根的近似值。

假定 $f(a)f(b)<0$ 且 $f'(x)$ 和 $f''(x)$ 在区间 $[a,b]$ 上恒正，那么 $y=f(x)$ 的图形如图 6-5 所示，而且 $f(b)f''(b)>0$。图中点ξ 表示所求方程的精确解。

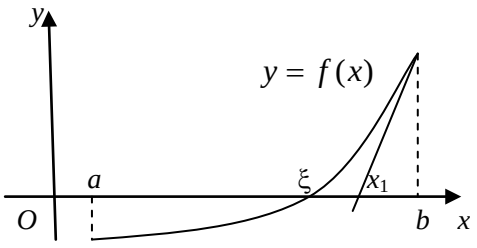


图 2-5

在点 $(b,f(b))$ 处作曲线的切线 $y=f(b)+f'(b)(x-b)$，该切线与 x 轴的交点为 $x_1=b-\dfrac{f(b)}{f'(b)}$，根据曲线的凹向可以看出 x_1 比 b 更接近精确解 ξ，用 x_1 作为新区间的右端点，重复上述过程。于是可产生一个单调趋向于 ξ 的点列（牛顿迭代公式）：

$$x_0=\begin{cases}a, & f(a)f''(a)>0\\ b, & f(b)f''(b)>0\end{cases},\qquad x_{n+1}=x_n-\frac{f(x_n)}{f'(x_n)}\qquad n=1,2,3,\cdots$$

下面来讨论迭代精度估计。由微分中值定理得，在 x_n 与 ξ 之间存在 η_n，使得

$$f(x_n)=f(x_n)-f(\xi)=f'(\eta_n)(x_n-\xi),$$

于是有$|x_n-\xi|=\frac{|f(x_n)|}{|f'(\eta_n)|}\le\frac{|f(x_n)|}{m}$，其中$m=\min\{f'(a),f'(b)\}$。因此，对于给定的误差$\delta$，当$|f(x_n)|<m\delta$时，近似解$x_n$与精确解$\xi$的误差$|x_n-\xi|<\delta$。

例3　用切线迭代法求方程$4x^3-6x^2+3x-2=0$的近似解，要求误差不超过10^{-6}。

解：令$f(x)=4x^3-6x^2+3x-2$，由$f(1)=-1,f(2)=12$知，在区间$[-1,2]$内有根。因为$f'(x)=12x^2-12x+3>0,f''(x)=24x-12>0,x\in[1,2]$，故隔断区间取为$[-1,2]$。

又因$f(2)f''(2)>0$，所以以$x_0=2$为迭代初始值，并由于$\min\{f'(1),f'(2)\}=3$，故当$|f(x_n)|<3\times10^{-6}$时，所求的x_n满足要求。

由上述分析建立切线迭代法的计算程序如下：

```
In[9]:= f[x_] := 4 x^3 - 6 x^2 + 3 x - 2;
        a = 1; b = 2; dalta = 10^(-6); k0 = 10; m = Min[f'[a], f'[b]];
        If[f[a] * f'[a] > 0, x0 = a, x0 = b];
        Do[x = x0 - f[x0] / f'[x0]; Print[N[x, 17], "     n=", k];
         If[Abs[f[x]] < m * dalta, Break[],
          If[k < k0, x0 = x, Print["fail"]]], {k, k0}]
```

运行后从输出结果看，迭代到第 5 次结束程序，近似解为 $x=1.2211248173005456$。

由例 2、例 3 可见，切线法比二分法收敛得要快，不过切线法要求的前提条件比较强，所以当难以判断是否满足条件时，应采用二分法。大家可以通过绘制图形知道在隔断区间上是否满足切线法的条件，这样可以免去精确的推导。

实验习题 2

1、用图形法和二分法求方程$\sin x+\cos x=0$在区间$[-1,4]$内的根，要求误差小于10^{-6}。

2、用切线迭代法求方程$x^2+\sqrt{x}-3=0$的近似解，要求误差不超过10^{-6}。

实验三　　一元函数图形及其性态

本实验的目的是让同学熟悉数学软件 Mathematica 所具有的良好的作图功能，并通过函数图形来认识函数，运用函数的图形来观察和分析函数的有关性态，建立数形结合的思想。

例1　给定函数（1） $f(x)=\dfrac{x^4+x^3+x^2+x}{x^2+x+1}$ （2） $\begin{cases} x(t)=\cos t\sin 2t \\ y(t)=\sin t\cos 3t \end{cases}$

（3） $g(x)=\begin{cases} \dfrac{1}{x}\sin x^2, & x\neq 0 \\ 0, & x=0 \end{cases}$

在同一坐标系下画出以上三个函数的图形。

解：输入命令如下：

```
In[1]:= f[x_] := (x^4 + x^3 + x^2 + x)/(x^2 + x + 1);
        t1 = Plot[f[x], {x, -2, 2}, PlotStyle → RGBColor[0, 1, 0]]
        t2 = ParametricPlot[{Cos[t] Sin[2 t], Sin[t] Cos[3 t]},
          {t, 0, 2 Pi}, PlotStyle → RGBColor[1, 0, 0]]
        g[x_] := If[x ≠ 0, 1/x Sin[x^2] , 0];
        t3 = Plot[g[x], {x, -2, 2}, PlotStyle → RGBColor[0, 0, 1]]
        Show[t1, t2, t3, PlotRange → {-1, 1}]
```

在上面的程序中，命令“Plot”的选项“PlotStyle→RGBColor[a,b,c]”是指选用颜色绘图，其中 a,b,c 为介于[0，1]之间的数，若 a,b,c 选择[1,0,0]、[0,1,0]、[0,0,1]，则分别表示的是三元色：红、绿、蓝。运行后输出结果如图 3-1 所示：

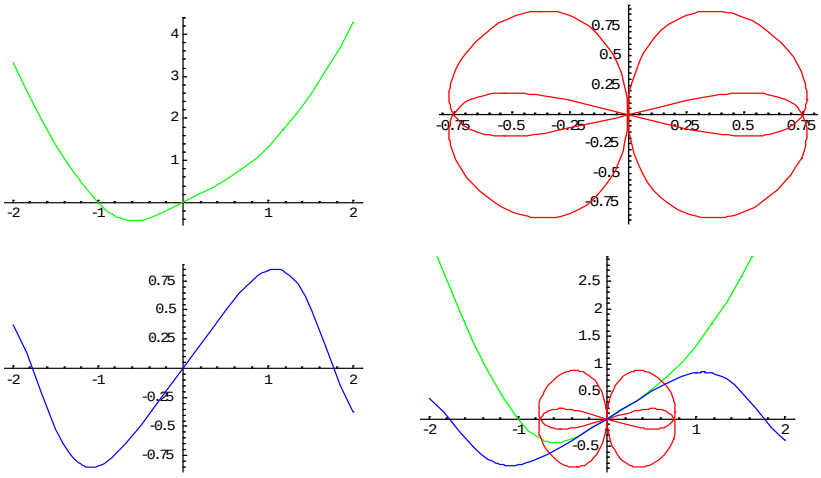


图 3-1

例 2　绘出函数 $f(x)=x^5+x^4-5x^3-x^2+8x-4$ 以及 $f'(x), f''(x)$ 的图形，并找出所有的驻点和拐点。

解：首先，我们不妨将 $f(x)$ 的自变量显示范围定为 $[-3,3]$，则输入如下命令：

```
In[7]:= f[x_] := -4 + 8 x - x^2 - 5 x^3 + x^4 + x^5
        Plot[f[x], {x, -3, 3},
         GridLines → Automatic,
         Frame → True,
         PlotStyle → RGBColor[1, 0, 0]]
```

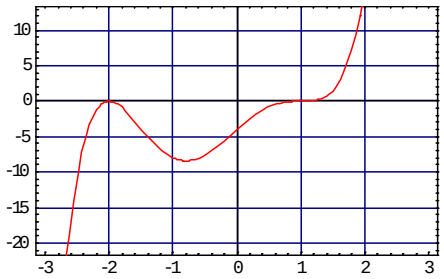


图 3-2

为了利于观察一些特殊点的位置，我们选择了选项“GirdLines→Automatic”使图形的坐标平面上出现了网格线，而且这时 Mathematica 将自动选择相应的 y 的显示范围为 $[-20,10]$（如图 3-2）。图中的曲线差不多是函数 $y=f(x)$ 图形的“全貌”。从图形中可以看出 $x=-2, x=1$ 为函数的零点，单调性在 $x=-2, x=-0.8, x=1$ 附近改变，而且在 $x=-1, x=0, x=1$ 附近曲线凸向似乎有所改变。总之，由函数的图形我们只能近似地判断出一些信息，那么这些印象是否属实呢？为了证实这些印象，我们利用下面的 Mathematica 语句来加以验证：

```
In[9]:= Plot[f'[x], {x, -3, 3}, GridLines → Automatic, Frame → True,
         PlotStyle → RGBColor[1, 0, 0],
         PlotLabel → "A Graph of f'[x]"]
        Plot[f''[x], {x, -3, 3}, GridLines → Automatic, Frame → True,
         PlotStyle → RGBColor[1, 0, 0],
         PlotLabel → "A Graph of f''[x]"]
```

运行后，绘出了 $f(x)$ 的一阶导函数和二阶导函数的图形（如图 3-3），从图中可以分别观察出，$f'(x)$ 有三个零点，且均为 $f(x)$ 的极值点；$f''(x)$ 有三个零点，且均为 $f(x)$ 的拐点。为了具体求出这些极值点和拐点，下面我们可以利用解方程的命令“Solve”来求解 $f'(x)$ 和 $f''(x)$ 的实根，输入命令为：

```
In[11]:= Solve[f'[x] == 0, x]
         Solve[f''[x] == 0, x]
```

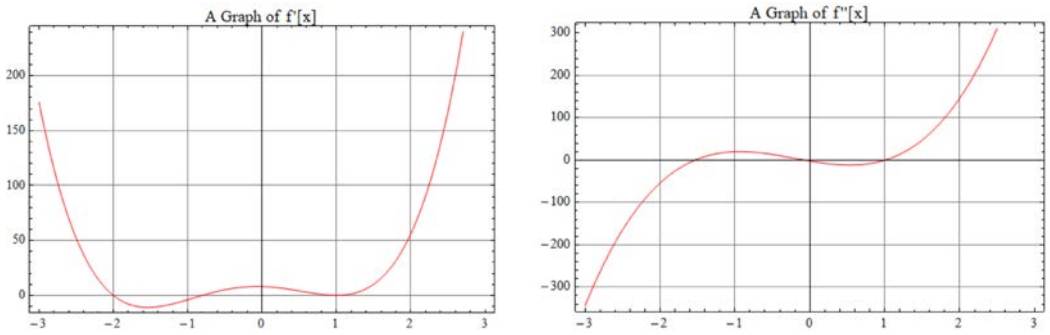


图 3-3

运行后可得到这两个方程的解为：

$$\left\{\{x \to -2\}, \left\{x \to -\frac{4}{5}\right\}, \{x \to 1\}, \{x \to 1\}\right\}$$

$$\left\{\{x \to 1\}, \left\{x \to \frac{1}{10}\left(-8 - 3\sqrt{6}\right)\right\}, \left\{x \to \frac{1}{10}\left(-8 + 3\sqrt{6}\right)\right\}\right\}$$

这样我们利用 Mathematica 并同时结合函数微分学的知识找出了一些关键点，从而对函数的图形就有了真实全面的了解。

实验习题 3

1、 制作函数 $y = \sin cx$ 的图形动画，并观察参数 c 对函数图形的影响。

2、 已知函数 $f(x) = \dfrac{1}{x^2 + 2x + c}(-5 \le x \le 4)$，作出并比较当 c 分别取-1,0,1,2,3 时的图形，并从图上观察极值点、驻点、单调区间、凹凸区间以及渐近线。

实验四　　泰勒公式与函数逼近

一个函数 $f(x)$ 若在点 x_0 的邻域内足够光滑，则在该邻域内有泰勒公式

$$f(x)=f(x_0)+\sum_{k=1}^{n}\frac{f^{(k)}(x_0)}{k!}(x-x_0)^k+o(|x-x_0|^n)，$$

当 $|x-x_0|$ 很小时，有 $f(x)\approx f(x_0)+\sum_{k=1}^{n}\frac{f^{(k)}(x_0)}{k!}(x-x_0)^k$，

其中，$T(x)=f(x_0)+\sum_{k=1}^{n}\frac{f^{(k)}(x_0)}{k!}(x-x_0)^k$ 称为 $f(x)$ 在点 x_0 处的 n 阶泰勒多项式；

$o(|x-x_0|^n)$ 为余项。下面我们利用 Mathematica 计算函数 $f(x)$ 的各阶泰勒多项式，并通过绘制曲线图形，来进一步掌握泰勒展开与函数逼近的思想。

例 1 （泰勒公式的误差）利用泰勒多项式近似计算 e^x。若 $|x|<1$，要求误差 $|R_n|<0.005$。

解：我们根据拉格朗日余项 $|R_n|=\left|\frac{e^x}{(n+1)!}x^{n+1}\right|<\frac{e}{(n+1)!}|x|^{n+1}<\frac{3}{(n+1)!}$ 可得，欲使 $|R_n|<0.005$，只要取 $n=5$ 即可。下面的 Mathematica 语句利用函数 e^x 的 5 阶泰勒多项式来近似计算 e^{d0} 的值，并判断误差：

```
In[1]:= d0 = -1;
        While[d0 ≤ 1,
         a = N[Normal[Series[Exp[x], {x, 0, 5}]]] /. x → d0;
         Print[d0, "      ", a, "      ", N[Exp[d0]],
          "          ", N[Exp[d0]] - a]; d0 += 0.4]
```

输出结果为：

```
-1      0.366667      0.367879          0.00121277
-0.6      0.548752      0.548812          0.0000596361
-0.2      0.818731      0.818731          8.64113×10^-8
0.2      1.2214      1.2214          9.14935×10^-8
0.6      1.82205      1.82212          0.0000708004
1.      2.71667      2.71828          0.00161516
```

输出结果每一行的最后一项表示误差，从结果中可以看出，当 $|x|<1$，其误差 $|R_n|<0.005$。

例 2 （观察阶数 n 对误差的影响）利用函数 e^x 的 n 阶多项式计算 e 的值，并求误差。

（n=5,6,7,8,9,10）

解：为此，我们输入 Mathematica 语言如下：

```
In[3]:= n = 5;
        While[n ≤ 10,
         a = N[Normal[Series[Exp[x], {x, 0, n}]] /. x → 1, 17];
         Print[n, "   ", a, "    ", N[Exp[1], 17], "    ",
          Exp[1] - a]; n += 1]
```

输出结果为：

```
5   2.7166666666666667    2.7182818284590452    0.0016151617923786
6   2.7180555555555556    2.7182818284590452    0.0002262729034897
7   2.7182539682539683    2.7182818284590452    0.0000278602050770
8   2.7182787698412698    2.7182818284590452    3.0586177754×10^-6
9   2.7182815255731922    2.7182818284590452    3.028858530×10^-7
10  2.7182818011463845    2.7182818284590452    2.73126608×10^-8
```

从结果中可知，阶数越高，误差越小。

例 3（根据图形观察泰勒展开的误差）观察 $f(x)=\sin x$ 的各阶泰勒展开的图形。

解：（1）固定 $x_0=0$，观察阶数 n 的影响。

因为 $f(x)=\sin x$ 在 $x_0=0$ 处的偶数阶导数为零，所以首先我们在同一坐标系内显示函数 $f(x)=\sin x$ 及它的 $n(n=1,3,5,\cdots,13)$ 阶泰勒多项式的图形。故输入命令如下：

```
In[5]:= t = Table[Normal[Series[Sin[x], {x, 0, i}]], {i, 1, 13, 2}];
        PrependTo[t, Sin[x]];
        Plot[Evaluate[t], {x, -Pi, Pi}]
```

上述语句中的函数“`PrependTo[t,Sin[x]]`”是表示把函数 $\sin x$ 添加到表 t 中。运行得：

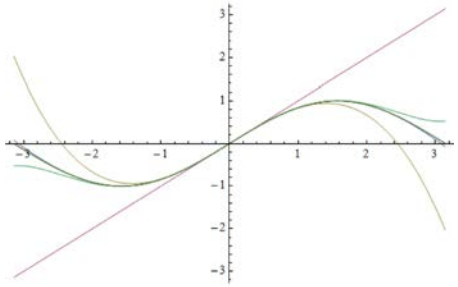


图 4-1

为了使图形比较更加生动，下面我们作出$\sin x$和它的某一阶泰勒多项式的同一坐标系下的比较图，并且在图中红色曲线表示函数$f(x)=\sin x$的图形，蓝色曲线表示泰勒多项式的图形。命令如下：

```
In[8]:= For[i = 1, i ≤ 11, a = Normal[Series[Sin[x], {x, 0, i}]];
        b = Plot[{a, Sin[x]}, {x, -2 Pi, 2 Pi},
          PlotStyle → {RGBColor[0, 0, 1], RGBColor[1, 0, 0]}];
        Print[b]; i = i + 2]
```

运行后得到了六幅图（图 4-2），从图表中可以观察到泰勒多项式与函数图形的重合与分离情况，显然在$[-\pi,\pi]$范围内，第五幅图中两个函数的图形已经基本上吻合了，也就是说，$\sin x$的 9 次多项式与函数几乎无差别。

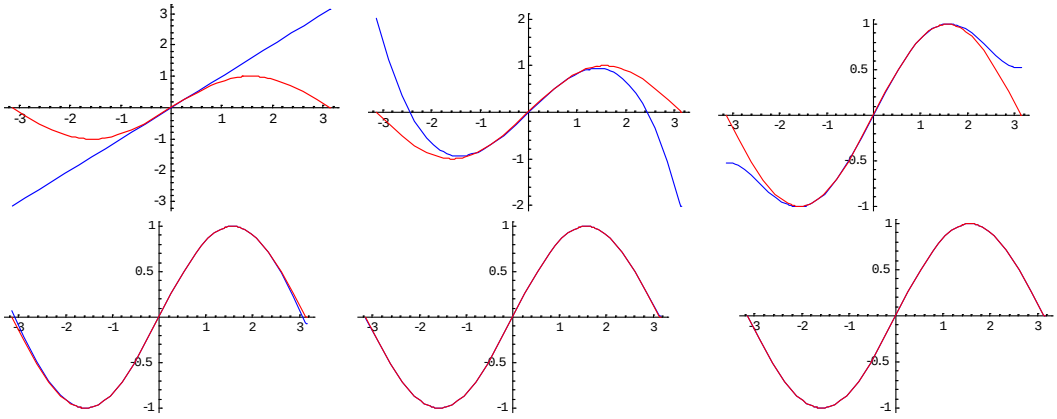


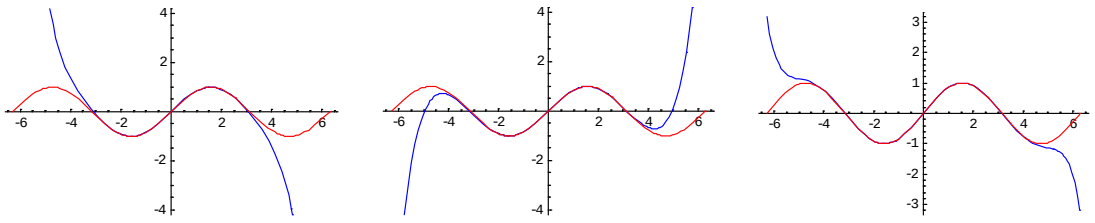
图 4-2

（2）扩大显示区间范围，以观察在偏离展开点x_0时泰勒多项式对函数的逼近情况。

显然，我们只要把上一个程序中的绘图命令中的x范围由$[-\pi,\pi]$分别改到$[-2\pi,2\pi]$，并相应增加阶数。故输入如下命令：

```
In[9]:= For[i = 7, i ≤ 17, a = Normal[Series[Sin[x], {x, 0, i}]];
        c = Plot[{a, Sin[x]}, {x, -2 Pi, 2 Pi},
          PlotStyle → {RGBColor[0, 0, 1], RGBColor[1, 0, 0]}],
        Print[c]; i = i + 2]
```

运行上面程序，绘出了从 7 阶直至 17 阶的泰勒多项式与$\sin x$的比较图（图 4-3），观察图表可得，在区间$[-2\pi,2\pi]$范围内，$\sin x$的 17 次多项式与函数吻合得很好了。



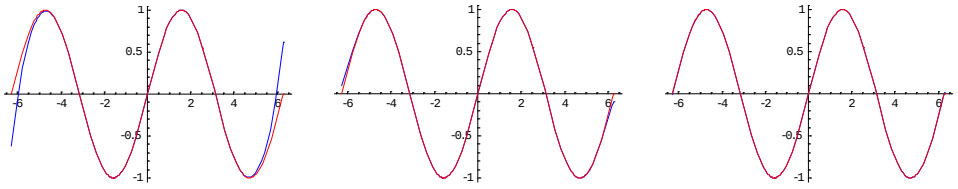


图 4-3

（3）固定 $n=6$，观察 x_0 对函数逼近的影响。

在下面的语句中，为了方便调用 $\sin x$ 的泰勒多项式，首先定义了 $\sin x$ 的泰勒展开函数 tt，然后用不同的颜色在同一坐标系中画出了 $\sin x$ 及 $\sin x$ 的分别在 $x_0=0, x_0=3, x_0=6$ 处的 6 阶泰勒多项式的图形：

```
In[10]:= tt[x0_, n_] := Normal[Series[Sin[x], {x, x0, n}]];
         gs0 = tt[0, 6]; gs3 = tt[3, 6]; gs6 = tt[6, 6];
         Plot[{Sin[x], gs0, gs3, gs6}, {x, -3 Pi, 3 Pi},
          PlotRange → {-2, 2},
          PlotStyle → {{RGBColor[0, 0, 1], Thickness[0.0074]},
            {RGBColor[1, 0, 1], Thickness[0.0074]},
            {RGBColor[1, 0, 0], Thickness[0.0074]},
            {RGBColor[0, 1, 0], Thickness[0.0074]}}]
```

输出的结果如图 4-4 所示。

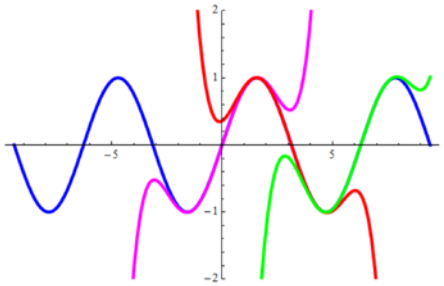


图 4-4

从本实验我们可以得到一些结论，函数的泰勒多项式对于函数的近似程度随着阶数的提高而提高，但是对于任一确定的次数的多项式，它只在展开点附近的一个局部范围内才有较好的近似精确度。

实验习题 4

1、 对 $f(x)=\cos x$ 重复上面的实验。

2、 作出函数 $y=\ln(\cos x^2+\sin x)$ $(-\frac{\pi}{4}\le x\le\frac{\pi}{4})$ 的函数图形和泰勒展开式（选取不同的 x_0 和 n 值）图形，并将图形进行比较。

实验五　定积分的近似计算

我们已经学习了定积分的基本概念和定积分的计算方法，那里所谓的计算方法，是基于原函数的牛顿-莱布尼兹公式。但在许多实际问题中遇到的定积分，被积函数往往不用算式给出，而通过图形或表格给出；或虽然可用一个算式给出，但是要计算它的原函数却很困难，甚至于原函数可能是非初等函数。本实验的目的，就是为了解决这些问题，介绍定积分的“数值积分”，即定积分的近似计算。

所谓定积分的近似计算，就是找到一个适当的计算公式，利用被积函数在积分区间上若干个点处的函数值，来计算定积分的近似值，并作出误差估计。我们知道，定积分 $\int_a^b f(x)dx$ 在几何上表示曲线 $y=f(x)$，直线 $x=a, x=b$ 及 x 轴所围成的曲边梯形的面积。定积分近似计算的思想，就是将积分区间分割成许多小区间，然后在小区间上近似计算小曲边梯形的面积，最后将小曲边梯形的面积求和，就得到了定积分的近似值。

1、 观察黎曼和式的收敛性

由定积分的定义知道，定积分就是黎曼和式 $\sum_{i=1}^{n} f(\xi_i)\Delta x_i$ 的极限，因此可以用黎曼和式来近似计算定积分。为计算方便，这里特殊的，将积分区间等分为 n 段，并以小区间中点处的函数值作近似，于是黎曼和式为：$\frac{b-a}{n}\sum_{k=1}^{n} f(a+((k-1)+0.5)\frac{b-a}{n})$，

因而　　$\int_a^b f(x)dx \approx \frac{b-a}{n}\sum_{k=1}^{n} f(a+((k-1)+0.5)\frac{b-a}{n})$。

例1　计算 $\int_2^3 \frac{1}{\ln x}dx$ 的黎曼和。

解：输入命令如下：

```
In[1]:= f[x_] := 1 / Log[x]; a = 2; b = 3; n = 200;
        s = NSum[f[a + ((k - 1) + 0.5) (b - a)/n] * (b - a)/n, {k, 1, n}]
```

上述命令是将区间[2, 3]等分为 200 段，运行求得黎曼和为：1.11842。

2、 梯形法

大家可以看出，用上述方法进行的近似计算，其实是对小曲边梯形的面积用矩形面积来近似，上面取的特殊的黎曼和又称为中点积分公式。如果不用矩形而改用梯形来近似，就可以得到定积分的一个较好的近似方法——梯形积分法。具体方法如下：

将区间$[a,b]$用$a=x_0,x_1,\cdots,x_n=b$等分为 n 个小区间，小区间的长度为$\frac{b-a}{n}$。设$y_i=f(x_i)=f(a+i\frac{b-a}{n})\quad(i=0,1,\cdots,n)$，则每个小梯形的面积为$\frac{y_i+y_{i+1}}{2}\cdot\frac{b-a}{n}$，从而得到梯形法的公式为：

$$\begin{aligned}\int_a^b f(x)dx &\approx [\frac{1}{2}(y_0+y_1)+\frac{1}{2}(y_1+y_2)+\cdots+\frac{1}{2}(y_{n-1}+y_n)]\frac{b-a}{n}\\ &=\frac{b-a}{n}[\frac{1}{2}(y_0+y_n)+y_1+y_2+\cdots+y_{n-1}]\\ &=\frac{b-a}{n}[\frac{f(a)+f(b)}{2}+\sum_{i=1}^{n}f(a+i\frac{b-a}{n})]\quad 。\end{aligned}$$

下面来估计梯形法的误差。第i个小曲边梯形的面积为$\Delta A_i=\int_{x_{i-1}}^{x_i}f(x)dx$，做变换$x=x_{i-1}+\frac{b-a}{n}t$，则$\Delta A_i=\int_{x_{i-1}}^{x_i}f(x)dx=\int_0^1 f(x_{i-1}+\frac{b-a}{n}t)\cdot\frac{b-a}{n}dt$，当$f''(x)$在区间$[a,b]$上连续时，利用分部积分法可以证明：

$$\Delta A_i=\frac{b-a}{2n}[f(x_{i-1})+f(x_i)]-\frac{(b-a)^3}{2n^3}\int_0^1 t(1-t)f''(x_{i-1}+\frac{b-a}{n}t)dt\text{。}$$

设M_2为$|f''(x)|$在区间$[a,b]$上的最大值，则第i个小曲边梯形与相应的梯形面积之差的绝对值估计如下：

$$\begin{aligned}&|\Delta A_i-\frac{b-a}{2n}[f(x_{i-1})+f(x_i)]|=\frac{(b-a)^3}{2n^3}|\int_0^1 t(1-t)f''(x_{i-1}+\frac{b-a}{n}t)dt|\\ &\le\frac{(b-a)^3}{2n^3}\int_0^1 t(1-t)|f''(x_{i-1}+\frac{b-a}{n}t)|dt\le\frac{(b-a)^3}{2n^3}\cdot M_2\int_0^1 t(1-t)|dt\\ &=\frac{(b-a)^3}{12n^3}M_2\end{aligned}$$

于是，梯形法的绝对误差为$\sum_{i=1}^{n}\frac{(b-a)^3}{12n^3}M_2=\frac{(b-a)^3}{12n^2}M_2$。

例2　用梯形法近似计算$\int_2^3\frac{1}{\ln x}dx$，要求误差不超过$10^{-5}$。

解：设$f(x)=\frac{1}{\ln x}$，则$f''(x)=\frac{2}{x^2(\ln x)^3}+\frac{1}{x^2(\ln x)^2}$，显然$f''(x)$在区间$[2,3]$上的最大值为$M_2=f''(2)$。下面我们根据梯形法利用 Mathematica 编程，在程序中，定义了n等

分时的梯形公式$t(n)$，并采用“Do”命令进行循环直到满足精度要求或达到预定的循环次数为止，每次循环要求输出n及$t(n)$。输入命令如下：

```
In[3]:= f[x_] := 1 / Log[x];
        a = 2; b = 3; m2 = N[f''[2]]; dalta = 10^(-5); n0 = 100;
        t[n_] := (b - a)/n * ((f[a] + f[b])/2 + Sum[f[a + i * (b - a)/n], {i, 1, n - 1}]);
        Do[Print[n, "      ", N[t[n], 8]];
         If[(b - a)^3/(12 n^2) * m2 < dalta, Break[], If[n == n0, Print["fail"]]],
         {n, n0}]
```

从运行结果看，循环到 100 次结束，最后输出“fail”，这表明没有达到精度要求，如把 n0 的值改为 200，再次运行，发现循环到 n=130 时结束，此时达到精度要求，积分的近似值为：1.1184286。

3、 抛物线法

梯形法的近似过程是在每个小区间中用直线段来近似被积函数段，即逐段地用线性函数来近似被积函数。为了进一步提高精确度，可以考虑在小范围内用二次函数来近似被积函数，这种方法称为抛物线法，也称为辛普森（Simpson）法。具体方法如下：

用分点$a = x_0, x_1, x_2, \cdots, x_n = b$，将积分区间$n$等分（这里要求$n$为偶数），各分点对应的函数值为$y_0, y_1, y_2, \cdots, y_n$，即$y_i = f(x_i) = f(a + i\dfrac{b-a}{n})$。我们知道平面上三点可以确定一条抛物线$y = px^2 + qx + r$，而相邻的两个小区间上经过曲线上的三个点，则由这三点做抛物线（因此抛物线法必须将区间等分为偶数个小区间），把这些抛物线构成的曲边梯形的面积相加，就得到了所求定积分的近似值。

下面计算在区间$[x_0, x_2]$上以抛物线为曲边的曲边梯形面积。为此，先计算区间$[-h, h]$上，以过$(-h, y_0), (0, y_1), (h, y_2)$三点的抛物线$y = px^2 + qx + r$为曲边的曲边梯形面积$S$：

$$S = \int_{-h}^{h} (px^2 + qx + r)dx = 2\int_{0}^{h} (px^2 + r)dx = \frac{2}{3}ph^3 + 2rh,$$

由$y_0 = ph^2 - qh + r,\ y_1 = r,\ y_2 = ph^2 + qh + r$得：$ph^2 = y_0 + y_2 - 2y_1$

故$S = \dfrac{1}{3}\left(2ph^3 + 6rh\right) = \dfrac{1}{3}h(2ph^2 + 6r) = \dfrac{1}{3}h(y_0 + 4y_1 + y_2)$。

取 $h=\dfrac{b-a}{n}$，则上面所求的 S 等于区间 $[x_0,x_2]$ 上以抛物线为曲边的曲边梯形的面积。同理可以得到区间 $[x_{i-1},x_{i+1}]$ 上以抛物线为曲边的曲边梯形的面积：

$$S_i=\frac{b-a}{3n}(y_{i-1}+4y_i+y_{i+1}),\ i=1,2,\cdots,n-1。$$

于是，将这 $\dfrac{n}{2}$ 个曲边梯形的面积加起来，得到定积分的近似值为（设 $n=2k$ ）：

$$S_n=\frac{b-a}{3n}[y_0+y_n+4(y_1+y_3+\cdots+y_{n-1})+2(y_2+y_4+\cdots+y_{n-2})]$$

$$=\frac{b-a}{6k}[f(a)+f(b)+4\sum_{i=1}^{k}f(x_{2i-1})+2\sum_{i=1}^{k-1}f(x_{2i})]。$$

上式称为辛普森公式或抛物线公式。用这个公式求定积分的近似值时，其绝对误差可以证明不超过 $\dfrac{(b-a)^5}{180n^4}M_4$，其中 M_4 是 $|f^{(4)}(x)|$ 在区间 $[a,b]$ 上的最大值。

例3　用抛物线法近似计算 $\int_2^3\frac{1}{\ln x}dx$，要求误差不超过 10^{-5}。

解：设 $f(x)=\dfrac{1}{\ln x}$，可由命令 **D[f[x],{x,4}]** 得到 $f(x)$ 的四阶导函数为：

$f^{(4)}(x)=\dfrac{24}{x^4(\ln x)^5}+\dfrac{36}{x^4(\ln x)^4}+\dfrac{22}{x^4(\ln x)^3}+\dfrac{6}{x^4(\ln x)^2}$，显然 $f^{(4)}(x)$ 在区间 $[2,3]$ 上的最大值为 $M_4=f^{(4)}(2)$。下面根据抛物线法的思想利用 Mathematica 编程，在程序中，与例 2 一样，定义了等分 $n=2k$ 时的抛物线公式 $p(k)$，并采用“Do”命令进行循环直到满足要求或达到预定的循环次数为止，每次循环要求输出 k 及 $p(k)$。输入命令如下：

```
In[7]:= f[x_] := 1 / Log[x];
        a = 2; b = 3; m4 = D[f[x], {x, 4}] /. x → 2; delta = 10^(-5);
        k0 = 100;
        p[k_] :=
          (b - a)/(6 k) * (f[a] + f[b] + 2 Sum[f[a + i * (b - a)/(2 k)], {i, 2, 2 k - 2, 2}] +
             4 Sum[f[a + i * (b - a)/(2 k)], {i, 1, 2 k - 1, 2}]);
        Do[Print[k, "     ", N[p[k]]];
         If[(b - a)^5/(180 * (2 k)^4) * m4 < delta, Break[], If[k == n0, Print["fail"]]],
         {k, k0}]
```

从运行结果看，循环到 $k=6$ 时因达到精度要求结束循环，并得到积分的近似值为：1.11843。从例 2、例 3 可以看出，抛物线法比梯形法收敛的要快，这与实际情况也是相符的。

最后，我们再说明一点，在 Mathematica 内部有一个数值积分的命令“`NIntegrate`”，例如要计算 $\int_2^3 \frac{1}{\ln x}dx$，我们可以调用命令：

```
In[12]:= N[Integrate[1 / Log[x], {x, 2, 3}]]
```

或者我们可以通过基本输入模板直接输入积分符号：

$$\mathbf{N}\left[\int_2^3 \mathbf{1 / Log[x]\ dx}\right]$$

字母“`N`”是表示输出的结果为实数的形式。运行后均得结果 1.11842。

虽然使用内部的命令计算数值积分非常方便，但是误差估计不明显，而且作为一个大学生，应该要知道隐藏在命令后面的原理。因此掌握本实验介绍的数值积分的原理、公式及编程方法也是很必要的。

实验习题 5

1、计算定积分 $\int_0^{\frac{\pi}{2}} \sin x^2 dx$ 的黎曼和。

2、分别用梯形法、抛物线法计算定积分 $\int_0^{\frac{\pi}{2}} \sin x^2 dx$ 的近似值（精确到 0.0001）。

实验七　常微分方程及追击问题

在实际问题中，需要研究一些变动的量以及它们之间的关系，由于这些量是时刻变化的，因此它们之间的关系不能用简单的代数方程，而要用微分方程来表达。在本实验中，我们要给出利用 Mathematica 求解一些简单常微分方程的方法，以及求微分方程的数值解的方法。

1、 求解常微分方程的精确解

在微分方程的学习中，对于一些特殊类型的微分方程可以求得其精确解，在 Mathematica 中对于这些类型，利用内部函数同样可以求得。

例 1 求解微分方程 $y' = e^{2x-y}$，并作出其积分曲线。

解：打开 Mathematica 并输入语句

```
In[1]:= DSolve[ y'[x] == Exp[ 2 x - y[x]], y[x], x]
```

运行后可求得方程的通解 $y = \ln(\frac{e^{2x}}{2} + C)$。

下面在同一坐标系中做出这个微分方程的六条积分曲线，输入如下命令：（其中分别对应 $C = 0,1,2,3,4,5$）。

```
In[2]:= t = Table[Log[e^(2 x)/2 + C], {C, 0, 5}];
        tu = Plot[Evaluate[t], {x, 1, 2}];
        Print[tu]
```

输出结果如图 7-1：

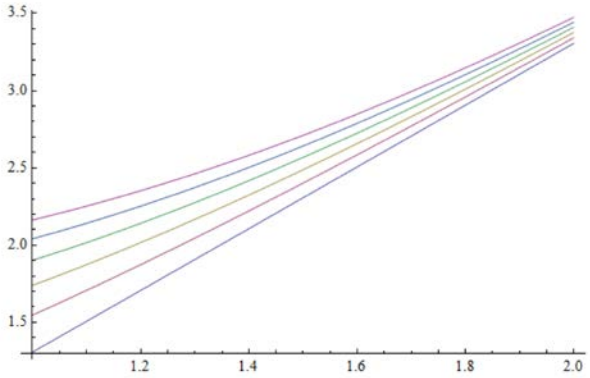


图 7-1

例 2 求微分方程 $y' = e^{2x-y}$ 满足初始条件 $y(0) = 0$ 的特解。

解：输入语句：

```
In[5]:= DSolve[ {y'[x] == Exp[ 2 x - y[x]], y[0] == 0}, y[x], x]
```

运行即可得到结果：

$$\text{Out[5]=}\ \left\{\left\{y[x] \to \text{Log}\left[\frac{1}{2}+\frac{e^{2x}}{2}\right]\right\}\right\}$$

。

对于一阶微分方程，除了可分离变量的微分方程，线性微分方程也可用 Mathematica 求出精确解，请同学们自己举例求解。除了这两类方程外，由实际操作可知，用 Mathematica 精确求解都比较困难，为此必须要考虑求微分方程的数值解问题。

2、 微分方程的数值解

例 3 求在区间$[0,10]$上初值问题$\begin{cases} y''+y'\sin^2 x+y=\cos^2 x \\ y(0)=1,\quad y'(0)=0 \end{cases}$的数值解，并作数值解的图形。

解：输入命令：

```
In[6]:= s1 = NDSolve[{y''[x] + Sin[x]^2 y'[x] + y[x] == Cos[x]^2,
          y[0] == 1, y'[0] == 0}, y[x], {x, 0, 10}]
```

运行后输出的结果为：

```
Out[6]= {{y[x] → InterpolatingFunction[{{0., 10.}}, <>][x]}}
```

它表示得到了插值函数 `InterpolatingFunction[domain, table]`类型的近似解，近似解的定义域 `domain` 为`[0,10]`。此时返回的解放在一个表中，函数 `y[x]`不能调用进行运算，要绘出数值解的图形，先要用“`Evaluate`”命令把它转化为可运算的。为此先把刚才得到的结果取代为 `y[x]`后定义为 `f[x]`，然后在区间`[0，10]`上作出 `f[x]`即近似解的图形。输入命令如下：

```
In[7]:= f[x_] := Evaluate[y[x] /. s1]; Plot[f[x], {x, 0, 10}]
```

运行后便得到近似解的图形如图 7-2。

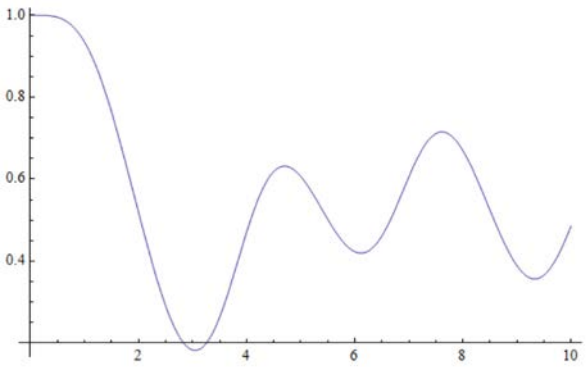
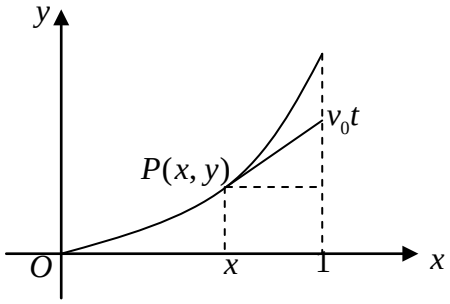


图 7-2

3、追击问题

下面看一个实际问题：我缉私艇雷达发现，正东 1 海里处一艘走私船正以常速 v_0 向北方向逃窜，缉私艇立即以 $2v_0$ 的速度追赶，借助于雷达，缉私艇航行的方向始终对准走私船。试求缉私艇的航行曲线方程，并问走私船航行多远时被我缉私艇追上。

（一）建立微分方程模型：首先如右图建立坐标系。



设缉私艇的航行曲线方程为 $y=f(x)$，在时刻 t 位于 $P(x,y)$，则有 $y'=\dfrac{v_0t-y}{1-x}$，又

$$\int_0^x\sqrt{1+y'^2}\,dx=2v_0t\,,$$

消去 t，得 $(1-x)y'+y=\dfrac{1}{2}\int_0^x\sqrt{1+y'^2}\,dx$。

两边对 x 求导，则实际问题化为求解微分方程：

$$\begin{cases}(1-x)y''=\dfrac{1}{2}\sqrt{1+y'^2}\\ y(0)=0,y'(0)=0\end{cases}$$

。此为可降阶的二阶微分方程，可解得：

缉私艇的航行曲线方程为 $y=-\sqrt{1-x}+\dfrac{1}{3}(1-x)^{\frac{3}{2}}+\dfrac{2}{3}$（$0\le x\le 1$）。

当 $x=1$ 时，$y(1)=\dfrac{2}{3}$，故走私船航行 $\dfrac{2}{3}$ 海里时被缉私艇追上。

或者用 mathematica 软件来求解上述微分方程，输入命令：

```
In[8]:= DSolve[{(1 - x) y''[x] == Sqrt[1 + (y'[x])^2] / 2, y'[0] == 0,
    y[0] == 0}, y[x], x]
```

运行后得到：

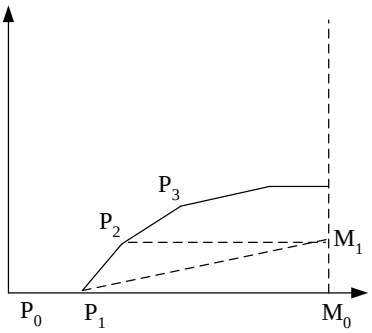
```
Out[8]= {{y[x] → 1/3 (2 - 2 Sqrt[1 - x] - Sqrt[1 - x] x)},
  {y[x] → 1/12 (-8 + 3 Sqrt[1 - x] + 15 Sqrt[(4 + 3 Sqrt[2])/(-4 + 3 Sqrt[2])] Sqrt[1 - x] -
     10 Sqrt[(2 (4 + 3 Sqrt[2]))/(-4 + 3 Sqrt[2])] Sqrt[1 - x] +
     6 Sqrt[1 - x] x - 6 Sqrt[(4 + 3 Sqrt[2])/(-4 + 3 Sqrt[2])] Sqrt[1 - x] x +
     4 Sqrt[(2 (4 + 3 Sqrt[2]))/(-4 + 3 Sqrt[2])] Sqrt[1 - x] x)}}
```

显然第一个解为所求的方程，为了求得当 $x=1$ 时函数得值，可由以下命令得到：

```
In[9]:= y1[x_] := 1/3 (2 - 2 Sqrt[1 - x] - Sqrt[1 - x] x); y1[1]
```

运行后也可得 $y(1)=\frac{2}{3}$，即走私船航行 $\frac{2}{3}$ 海里时被缉私艇追上。

（二）仿真方法：即模仿真实事件的行为和过程。在这个问题上，就是一步步地以时间间隔为 Δt 来模拟缉私艇追踪敌艇的实际过程。



如右图，当 $t=0$ 时，敌艇在 $M_0(1,0)$ 处，缉私艇在 $P_0(x_0=0,y_0=0)$ 处，方向指向敌艇，即 $\theta_0=0$。认为缉私挺的运动曲线是由水平方向和垂直方向组合而成。当 $t=\Delta t$ 时，敌艇运行到 $M_1(1,v_0\Delta t)$，而缉私艇运行到点 $P_1(x_1,y_1)$，则 $\begin{cases}x_1=x_0+2v_0\Delta t\cos\theta_0\\ y_1=y_0+2v_0\Delta t\sin\theta_0\end{cases}$，方向由 P_1 点指向 M_1 点，即 $\theta_1=\arctan\frac{v_0\Delta t-y_1}{1-x_1}$；当 $t=2\Delta t$ 时，敌艇运行到 $M_2(1,v_0 2\Delta t)$，缉私艇运行到点 $P_2(x_2,y_2)$，则 $\begin{cases}x_1=x_1+2v_0\Delta t\cos\theta_1\\ y_1=y_2+2v_0\Delta t\sin\theta_1\end{cases}$，方向由 P_2 点指向 M_2 点，即 $\theta_2=\arctan\frac{v_0 2\Delta t-y_2}{1-x_2}$；

于是，可以得到仿真的迭代格式为：

$$\begin{cases}x_{k+1}=x_k+2v_0\Delta t\cos\theta_k\\ y_{k+1}=y_k+2v_0\Delta t\sin\theta_k\end{cases}，\text{其中}\ \theta_k=\arctan\frac{kv_0\Delta t-y_k}{1-x_k}。$$

于是，利用 mathematica 可以对该实际问题进行模拟，不妨设 $v_0=5$ 海里/小时，则输入程序：

```
In[10]:= Clear[x, y, t, k]; Δt = 0.0005; x[0] = 0; y[0] = 0;
         t[0] = 0; v0 = 5;
         For[k = 0, k ≤ 1000, k++,
          θ[k] = ArcTan[(k v0 Δt - y[k]) / (1 - x[k])];
          x[k + 1] = x[k] + 2 v0 Δt Cos[θ[k]];
          y[k + 1] = y[k] + 2 v0 Δt Sin[θ[k]];
          t[k] = k Δt; Print[k, "    ", t[k], "    ", x[k], "    ",
           y[k]];
          If[x[k - 1] < 1 && x[k] >= 1, Break[]]]
```

在程序中，设定时间间隔为 $\Delta t=0.0005$ 小时，并且在 $x_{k-1}<1$ 且 $x_k\geq 1$ 迭代结束。

从运行结果可以看出，当 $k=267$ 时迭代结束，此时追赶的时间为 $t=0.1335$ 小时，$y_{267}=0.667497$ 说明敌艇航行了 0.667497 海里被缉私艇追上，这和前面的结果基本上一致。

另外，可以在程序中将 v_0 改变进行模拟，而从运行结果来看 v_0 不会改变敌艇被追上时的航行路程，改变的只是被追击上的时间。

实验习题 7

求在区间 $[2,5]$ 上初值问题 $\begin{cases} xy'-x^2y\sin x+1=0, \\ y(1)=1 \end{cases}$ 的数值解，并作出数值解的图形。