第12章 无穷级数

高等数学的主线是微积分，微积分的一个最基本的思想就是“极限”。高等数学中所谓的“极限”，跟生活中的“极限”不大一样，准确地说是所能达到“极限”的尺度不大一样，高等数学中的“极限”要远比生活中所能达到“极限”还要“极限”得多。无论是微分还是积分，都蕴含着这种“无限逼近”的极限思想。只要透彻理解了高等数学中“极限”的概念，就不难理解微分、积分和无穷级数之间的关系，它们之间最简单的关系就是都蕴含了这种“极限”的思想。其实可以说无穷级数是微积分的另一种表现形式，就像一次函数和直线方程实质上是同一种东西。

无穷级数的重要作用是可以用有限的、比较简单的函数形式来逼近复杂的函数，从而实现对复杂函数的近似计算，这对于工程计算具有非常重要的意义，也体现了微积分在工程领域的应用方法。无穷级数是高等数学最后的内容，微积分由极限的思想引入，经历了微分、积分的学习，最后又回归到无穷级数这一体现极限思想的概念，所以说无穷级数也是对微积分的回归与总结。

12.1 本章目标

本章将首先介绍常数项级数的概念，介绍无穷级数的收敛与发散，并将尝试通过 MATLAB 来实现某些级数的审敛，而对于一些收敛的级数则确定它的和。随着知识的进一步深入，将了解一些特殊的级数，譬如正项级数等，进一步地也获得了更多的审敛法。

然后将了解函数项级数的概念，知道其中最常见的幂级数。有了阿贝尔定理的理论支持，可以编写对应的脚本很方便地求收敛半径与收敛域。将函数展开成幂级数的部分即为泰勒展开，很容易在 MATLAB 中实现，有着对应的 Taylor 函数，这部分内容在上册的第 3 章中已经讲述了相关知识，本章不作重复赘述了。另外，关于函数幂级数展开式应用部分将不做重点介绍，因为近似计算用 MATLAB 来做未免显得有些笨拙，其实已经有了其他强大的计算工具；用幂级数解微分方程较有技巧性，需

要分析，不适合用 MATLAB 编写脚本；欧拉公式只是引入，为之后的傅里叶级数做铺垫。

本章最后讲述了傅里叶级数的相关知识，将编写更为复杂的脚本来将函数展开为傅里叶级数。在了解一般周期函数的傅里叶级数以及傅里叶级数的复数形式之后，就能够简化脚本。学习傅里叶级数后，可以了解一些拓展知识。将参考 MATLAB 中的帮助文档了解快速傅里叶变换函数 fft 的用法与功能，并利用它进行傅里叶变换的一种常见应用——频谱分析。

12.2 相关命令

本章涉及的新的 MATLAB 命令。

(1) symsum：级数的和。用法如下：

- F＝symsum(f,k,a,b)：从下界 a 到上界 b 返回级数 f 对求和阶数 k 求和。如果不指定 k，symsum 将使用由 symvar 确定的变量作为求和阶数。如果 f 是常数，那么默认变量是 x。
- symsum(f,k,[a b])或 symsum(f,k,[a; b])等价于 symsum(f,k,a,b)。
- F＝symsum(f,k)：返回级数 f 关于求和阶数 k 的不定和(反差分)。f 定义了这个级数，使得不定和 f 满足 F(k＋1)－F(k)＝f(k)的关系。如果不指定 k，symsum 将使用由 symvar 确定的变量作为求和阶数。如果 f 是常数，那么默认变量是 x。

(2) strcat：水平串联字符串。用法如下：

- s＝strcat(s1,…,sN)：水平串联 s1,…,sN。每个输入参数都可以是字符数组、字符向量元胞数组或字符串数组。如果任一输入是字符串数组，则结果是字符串数组；如果任一输入是字符向量元胞数组，并且没有输入字符串数组，则结果是字符向量元胞数组；如果所有输入都是字符数组，则结果是字符数组。对于字符数组输入，strcat 会删除尾随的 ASCII 空白字符：空格、制表符、垂直制表符、换行符、回车和换页符。对于元胞数组和字符串数组输入，strcat 不删除尾随空白字符。

(3) fft：快速傅里叶变换。用法如下：

- Y＝fft(X)：用快速傅里叶变换(FFT)算法计算 X 的离散傅里叶变换(DFT)。
  如果 X 是向量，则 fft(X)返回该向量的傅里叶变换。
  如果 X 是矩阵，则 fft(X)将 X 的各列视为向量并返回每列的傅里叶变换。
  如果 X 是一个多维数组，则 fft(X)将沿大小不等于 1 的第一个数组维度的值视为向量并返回每个向量的傅里叶变换。
- Y＝fft(X,n)：返回 n 点 DFT。如果未指定任何值，则 Y 的大小与 X 相同。
  如果 X 是向量且 X 的长度小于 n，则为 X 补上尾零以达到长度 n。
  如果 X 是向量且 X 的长度大于 n，则对 X 进行截断以达到长度 n。

如果 X 是矩阵，则每列的处理与向量情况相同。

如果 X 为多维数组，则大小不等于 1 的第一个数组维度的处理与向量情况相同。

- Y＝fft(X,n,dim)：返回沿维度 dim 的傅里叶变换。例如，如果 X 是矩阵，则 fft(X,n,2)返回每行的 n 点傅里叶变换。

12.3 常数项级数的计算

本章最基础的部分是理解什么是级数，并会计算一些简单的级数的和。在 MATLAB 中，可以通过 symsum 函数实现这一点。

首先用 syms 函数定义 k 与 n，那么 symsum 函数则可以计算用 k 表示的通式在自变量遍历某一段区间的总和。

为了更好地理解 symsum 的用法，可以在 MATLAB 中先做几个小实验：

定义：syms k n

输入：	输出：
symsum(k)	k^2/2 - k/2
symsum(k,0,n-1)	(n*(n - 1))/2
symsum(k,0,n)	(n*(n + 1))/2
symsum(k^2,0,n)	(n*(2*n + 1)*(n + 1))/6
expand(symsum(k^2,0,n))	n^3/3 + n^2/2 + n/6
symsum(k^2,0,10)	385
symsum(k^2,[0,10])	385
symsum(k^2,11,10)	0
symsum(1/k^2)	-psi(k, 1)
symsum(1/k^2,1,Inf)	pi^2/6

其中，expand 表示将含括号的多项式拆开，化为单项式之和；psi 表示 Ψ 函数，也称为双 γ 函数，是 gamma 函数的对数导数；Inf 表示正无穷大。

下面以例题来展示如何运用 symsum 函数计算简单的级数的和。

例 12-1 计算等比级数的和。

解：

```
syms a q k
symsum(a*q^k,k,0,Inf)
```

运行程序，得到如下结果：

$$\text{ans} = \begin{cases} a\infty & \text{if } 1 \leqslant q \\ -\dfrac{a}{q-1} & \text{if } |q| < 1 \end{cases}$$

可见该函数很好地解决了等比级数求和问题，并且做出了分类：当 $q \geqslant 1$ 时，和为正无穷大乘以 a，与 a 的符号有关；当 $|q|<1$ 时，和为 $\frac{a}{1-q}$。同时可以理解的是，只要通过 symsum 函数能够算出某级数的和，且为有限量，那么它必然是收敛的；若该和与 Inf 有关或表示为 NaN，那么它就是一个发散级数。

例 12-2 证明级数 $1+2+3+\cdots+n+\cdots$ 是发散的。

解：

```
syms k
symsum(k,k,1,Inf)
```

运行程序，得到如下结果：

```
ans = ∞
```

说明该级数是发散的。

例 12-3 判定无穷级数 $\frac{1}{1\cdot 2}+\frac{1}{2\cdot 3}+\cdots+\frac{1}{n\cdot(n+1)}+\cdots$ 的收敛性。

解：

```
syms k
symsum(1/(k*(k+1)),k,1,Inf)
```

运行程序，得到如下结果：

```
ans = 1
```

也就是说该级数收敛且和为 1。可以看出面对简单的级数和与收敛性问题，只用一个 symsum 函数就完全可以解决。

12.4 常数项级数的审敛法

在上一节中，了解到了 symsum 函数可以计算常数级数和，在本节将继续用该函数来判断级数的审敛性。

例 12-4 讨论 p 级数的收敛性。

解：

```
syms p k
symsum(1/k^p,k,1,Inf)
```

运行程序，得到如下结果：

```
ans = {ξ(p) if 1 < real(p)}
```

尝试按之前的做法，发现答案比较奇怪。仅讨论 p 为实数时，知道当 p 大于 1 时级数收敛，而 zeta 函数在 MATLAB 里表示黎曼 zeta 函数，事实上就是当 p 为实数时的 p 级数。也就是说，symsum 函数只是起到了转换表达方式的作用，并没有起到计算的功能，也不能进一步地确切地说明 p 级数是否收敛。

不过，通过教材上的方法得到 p 级数的收敛性后，可以将其他级数与 p 级数作比较，利用极限形式比较审敛法：

设 $\sum_{n=1}^{+\infty} u_n$ 为正项级数，

(1) 如果 $\lim\limits_{n\to\infty} nu_n = l > 0$（或 $\lim\limits_{n\to\infty} nu_n = +\infty$），那么级数 $\sum_{n=1}^{+\infty} u_n$ 发散；

(2) 如果 $p > 1$，而 $\lim\limits_{n\to\infty} n^p u_n = l(0 \leqslant l < +\infty)$，那么级数 $\sum_{n=1}^{+\infty} u_n$ 收敛。

这样就可以判断级数的收敛情况。具体实现脚本为：

```
%% 输入通项
syms n f p;
f = f(x);
%% 审敛
if limit(n * f,n,Inf)> 0
    fprintf("级数发散")
else
    if limit(n^(1 + 1e - 10) * f,n,Inf)> = 0&&isfinite(limit(n^(1 + 1e - 10) * f,n,Inf)) == 1
    fprintf("级数收敛")
    else
    fprintf("无法用本方法判断该级数收敛性")
    end
end
```

把通项 f 用 n 的表达式来表示，就可以判断级数的收敛性了，其中 isfinite 函数能够判断某值是否为有限值，若有限则输出逻辑值 1；无限/NaN 则输出 0。1e－10 指 10^{-10}，表示非常小的数，这是因为通过(2)可以判断 p 越接近 1 越有可能满足条件。

例 12-5 证明级数 $\sum_{n=1}^{+\infty} \frac{1}{\sqrt{n(n+1)}}$ 是发散的。

解：

```
% 输入通项
syms n f p;
f = 1/sqrt(n * (n + 1));
```

```
% 审敛
if limit(n*f,n,Inf)>0
    fprintf("级数发散")
else
    if limit(n^(1+1e-10)*f,n,Inf)>=0&&isfinite(limit(n^(1+1e-10)*f,n,Inf))==1
    fprintf("级数收敛")
    else
    fprintf("无法用本方法判断该级数收敛性")
    end
end
```

运行程序,得到如下结果:

```
级数发散
```

利用之前编写的脚本,将 f 表示为 1/sqrt(n*(n+1)),运行脚本,得到该级数是发散的。同样地,可以编写使用比值审敛法与根值审敛法判断级数收敛性的脚本,这里以比值审敛法为例。读者可以自己尝试编写根值审敛法的脚本。

例 12-6 判断级数 $\sum_{n=1}^{+\infty}\frac{1}{(n-1)!}$ 的收敛性。

解:

```
% 输入通项
syms n f1 f2;
f1=1/factorial(n-1);
f2=subs(f1,n,n+1);
% 审敛
l=limit(f2/f1,n,Inf);
if l>1
    fprintf("级数发散")
else
    if l<1
        fprintf("级数收敛")
    else
        fprintf("无法用本方法判断该级数收敛性")
    end
end
```

运行程序,得到如下结果:

```
级数收敛
```

其中 subs 起到复制的作用,把 f2 作为 f1 的副本的同时,将 n 替换为 n+1。

根据绝对收敛必定收敛的定理,便可以通过判断正项级数收敛性的脚本解决绝大多数问题,为此只需对之前的脚本做一些改进。

12.5 幂级数

将常数项级数的概念扩展至函数项级数后，便引入了一个新的简单而常见的概念——幂级数。通过教材知识的学习，可以知道幂级数有收敛域，在收敛域内它会收敛于某一和函数。通过 MATLAB 可以很容易地求出幂级数的收敛半径进而判断出收敛域。

例 12-7 求幂级数 $x-\frac{x^2}{2}+\frac{x^3}{3}-\cdots+(-1)^{n-1}\frac{x^n}{n}+\cdots$ 的收敛半径与收敛域。

解：

```
% 输入通项
syms n x an an1;
an = (-1)^(n-1)^n/n;
an1 = subs(an,n,n+1);
% 判断收敛半径
l = limit(abs(an1/an),n,Inf);
R = 1/l
% 判断收敛域
if isfinite(symsum(an*R^n,n,1,Inf)) == 1
    Right = 1;
else
    Right = 0;
end
if isfinite(symsum(an*(-R)^n,n,1,Inf)) == 1
    Left = 1;
else
    Left = 0;
end
% 输出收敛域
if Left == 1
    s = '[';
else
    s = '(';
end
s = strcat(s,num2str(double(-R),'%d,'),num2str(double(R),'%d'));
if Right == 1
    s = strcat(s,']');
else
    s = strcat(s,')');
end
fprintf(s);
```

运行程序，得到如下结果：

```
R = 1
[-1,1)
```

事实上，对于简单的幂级数，完全可以仅通过 symsum 函数判断它的性质，既可以了解它的收敛域，又能够知晓它在收敛域内的和函数。

例 12-8 求幂级数 $\sum_{n=0}^{\infty}\frac{x^n}{n+1}$ 的和函数。

解：

```
syms x n
symsum(x^n/(n+1),n,0,Inf)
```

运行程序，得到如下结果：

$$\text{ans} = \begin{cases} \infty & 1 \leqslant x \\ -\dfrac{\log(1-x)}{x} & |x| \leqslant 1 \wedge x \neq 1 \end{cases}$$

由此可以看到，该级数的收敛域为$[-1,1)$，并且它的和函数为$-\frac{\ln(1-x)}{x}$。特别地，由和函数的连续性可以得到其在 $x=0$ 处取 1。

12.6 傅里叶级数

这节将学习如何通过 MATLAB 将函数展开为傅里叶级数。根据教材中的操作步骤，以 $f(x)=x$ 为例将其展开，实现代码为：

```
%% 参数定义
syms x n;
f = x;
n2 = 5;
A = zeros(2,n2);
f1 = int(f*cos(n*x),-pi,pi);
f2 = int(f*sin(n*x),-pi,pi);
f3 = 0;
%% 计算傅里叶系数
a0 = int(f,-pi,pi)/pi;
for idx = 1:n2
    A(1,idx) = int(f*cos(idx*x)/pi,-pi,pi);
    A(2,idx) = int(f*sin(idx*x)/pi,-pi,pi);
end
%% 打印结果
s = 'f(x) = ';
```

```
if (a0～ = 0)
    s = strcat(s,num2str(a0/2,'%g + '));
else
    a0 = 0;
end
if A(1,1)～ = 0
    s = strcat(s,num2str(A(1,1),'%g * cos(x) + '));
else
    A(1,1) = 0;
end
if A(2,1)～ = 0
    s = strcat(s,num2str(A(2,1),'%g * sin(x) + '));
else
    A(2,1) = 0;
end
for idx = 2:n2
    if A(1,idx)～ = 0
        s = strcat(s,num2str(A(1,idx),'%g * '),num2str(idx,'cos(%gx) + '));
    else
        A(1,idx) = 0;
    end
    if A(2,idx)～ = 0
        s = strcat(s,num2str(A(2,idx),'%g * '),num2str(idx,'sin(%gx) + '));
    else
        A(2,idx) = 0;
    end
end
s = strcat(s,'...');
fprintf(s);
```

运行程序，得到如下结果：

```
f(x) = 2 * sin(x) - 1 * sin(2x) + 0.666667 * sin(3x) - 0.5 * sin(4x) + 0.4 * sin(5x) + …
```

```
%% 画图检验
fplot(f,[-pi,pi]);
for idx = 1:n2
    f3 = f3 + A(1,idx) * cos(idx * x) + A(2,idx) * sin(idx * x);
end
hold on;
fplot(f3 + a0/2,[-pi,pi],'--r');
xlabel('x'), ylabel('f');
title(['n 为',num2str(n2)])
legend('原函数', '傅里叶级数')
hold off
```

运行程序，得到如图 12-1 所示的检验图（对照图）。

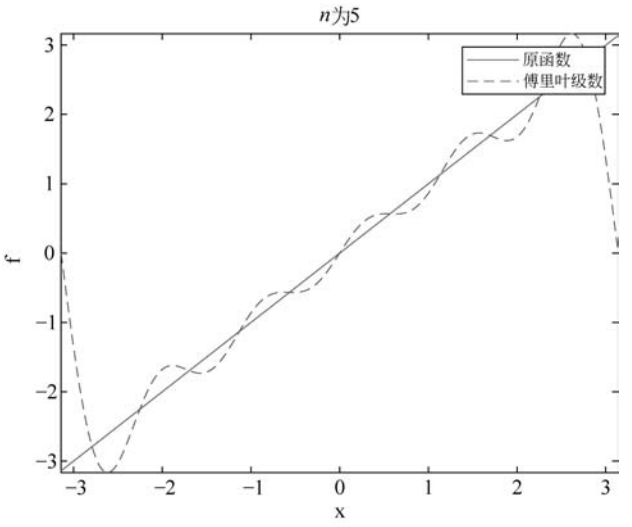

图 12-1　检验图（$n=5$）

进一步地，考查将 n 提高后的结果（只打印图片），如图 12-2 所示。

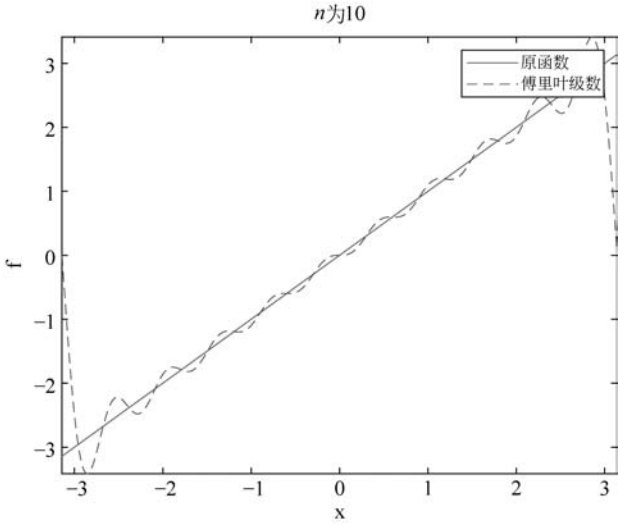

图 12-2　对照图（$n=10$）

可以看出通过教材中的方法能够实现将函数展开为傅里叶级数，且精度随 n 的增大而提高。不过善于思考的读者可以发现，对于某些分段函数（如方波，是一种非正弦曲线的波形）无法用统一的表达式表达 f，无法输入该脚本也无法求定积分。特定的分段函数可以使用 polyfit 与 polyval 函数拟合为多项式函数，但方波显然是难以较好地拟合为多项式函数的。为此需要使用数值积分函数 integral（会导致精确度降低）。

例 12-9 设 $f(x)$是周期为 2π 的周期函数，它在$[-\pi,\pi)$上的表达式为 $f(x)=\begin{cases}-1, & -\pi\leqslant x<0\\ 1, & 0\leqslant x<\pi\end{cases}$，将 $f(x)$展开为傅里叶级数。

解：

```
% 参数定义
syms n x;
f = @(x)square(x);
n2 = 10;
A = zeros(2, n2);
f0 = 0;
x0 = -pi:0.01:pi;
% 计算傅里叶系数
a0 = integral(@(x)f(x)./pi, -pi,pi);
for idx = 1:n2
    A(1,idx) = integral(@(x)f(x).*cos(idx.*x)./pi, -pi,pi);
    A(2, idx) = integral(@(x)f(x).*sin(idx.*x)./pi, -pi,pi);
end
% 打印结果
s = 'f(x) = ';
if abs(a0)>=1e-10
    s = strcat(s,num2str(a0/2,'%g+'));
else
    a0 = 0;
end
if abs(A(1,1))>=1e-10
    s = strcat(s,num2str(A(1,1),'%g*cos(x)+'));
else
    A(1,1) = 0;
end
if abs(A(2,1))>=1e-10
    s = strcat(s,num2str(A(2,1),'%g*sin(x)+'));
else
    A(2,1) = 0;
end
for idx = 2:n2
    if abs(A(1,idx))>=1e-10
        s = strcat(s,num2str(A(1,idx),'%g*'),num2str(idx,'cos(%gx)+'));
    else
        A(1,idx) = 0;
    end
    if abs(A(2,idx))>=1e-10
        s = strcat(s,num2str(A(2,idx),'%g*'),num2str(idx,'sin(%gx)+'));
    else
```

```
            A(2,idx) = 0;
        end
    end
    s = strcat(s,'...');
    fprintf(s);
    % 画图检验
    plot(x0,f(x0));
    for idx = 1:n2
        f0 = f0 + A(1,idx) * cos(idx * x) + A(2,idx) * sin(idx * x);
    end
    hold on
    fplot(f0 + a0/2,[ - pi,pi]);
    xlabel('x'), ylabel('f');
    hold off
```

运行程序,得到如下结果:

```
f(x) = 1.27324 * sin(x) + 0.424413 * sin(3x) + 0.254648 * sin(5x) + 0.181891 * sin(7x) +
0.141471 * sin(9x) + …
```

得到的检验图(对照图)如图 12-3 所示。

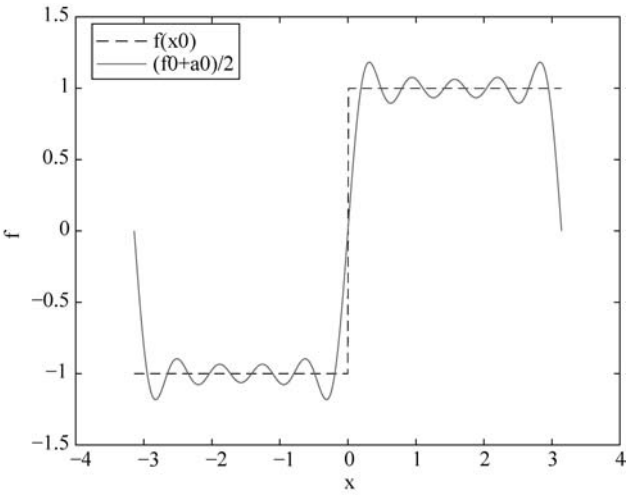

图 12-3 方波傅里叶级数与原函数的对照图

其中@是 MATLAB 中函数句柄的符号,在 integral 函数中用到。可以在函数名称前添加一个@符号来为函数创建句柄。

由于 integral 函数并不精确,会有极小的偏差,导致有些值为零的数值变为极小但非零(1e−17 等),为此需要修改一些判断条件,在脚本中有体现。

本题利用了 square 函数,而对于一般的分段函数,如 $f(x)=\begin{cases}x, & -\pi\leqslant x<0\\ 0, & 0\leqslant x<\pi\end{cases}$,可以通

过 f=@(x)(x<0). * x;实现，其中利用了在条件为 True 时逻辑值为 1，为 False 时逻辑值为 0 的特点。同时，使用逻辑值进行乘、除、幂运算时符号前要加“.”。但在之前的脚本中，如果不使用@符号，是不能按这种方法实现的，感兴趣的读者可以尝试一下。

12.7 一般周期函数的傅里叶级数

一般周期函数的周期不一定是 2π，不过通过教材上的定理，很容易发现只需修改一小部分脚本就能够实现将周期不为 2π 的函数展开为傅里叶级数。

例 12-10 设 $f(x)$ 是周期为 4 的周期函数，它在[-2，2）上的表达式为 $f(x)=\begin{cases}0, & -2\leqslant x<0\\ 1, & 0\leqslant x<2\end{cases}$，将 $f(x)$ 展开成傅里叶级数。

解：

```
% 参数定义
syms n x;
f = @(x)(x>=0);
n2 = 10;
l = 2;
A = zeros(2,n2);
f0 = 0;
x0 = -l:0.01:l;
% 计算傅里叶系数
a0 = integral(@(x)f(x)./l,-l,l);
for idx = 1:n2
    A(1,idx) = integral(@(x)f(x).*cos(idx.*x.*pi/l)./l,-l,l);
    A(2,idx) = integral(@(x)f(x).*sin(idx.*x.*pi/l)./l,-l,l);
end
% 打印结果
s = 'f(x) = ';
if abs(a0)>=1e-10
    s = strcat(s,num2str(a0/2,'%g + '));
else
    a0 = 0;
end
for idx = 1:n2
    if abs(A(1,idx))>=1e-10
        s = strcat(s,num2str(A(1,idx),'%g*'),num2str(idx*pi/l,'cos(%gx) + '));
    else
        A(1,idx) = 0;
    end
    if abs(A(2,idx))>=1e-10
        s = strcat(s,num2str(A(2,idx),'%g*'),num2str(idx*pi/l,'sin(%gx) + '));
```

```
    else
        A(2,idx) = 0;
    end
end
s = strcat(s,'...');
fprintf(s);
% 画图检验
plot(x0,f(x0));
for idx = 1:n2
    f0 = f0 + A(1,idx) * cos(idx * x * pi/l) + A(2,idx) * sin(idx * x * pi/l);
end
hold on
fplot(f0 + a0/2,[ - l,l]);
xlabel('x'), ylabel('f');
hold off
```

运行程序,得到如下结果:

```
f(x) = 0.5 + 0.63662 * sin(1.5708x) + 0.212207 * sin(4.71239x) + 0.127324 * sin(7.85398x) +
0.0909457 * sin(10.9956x) + 0.0707355 * sin(14.1372x) + …
```

得到的检验图(对照图)如图 12-4 所示。

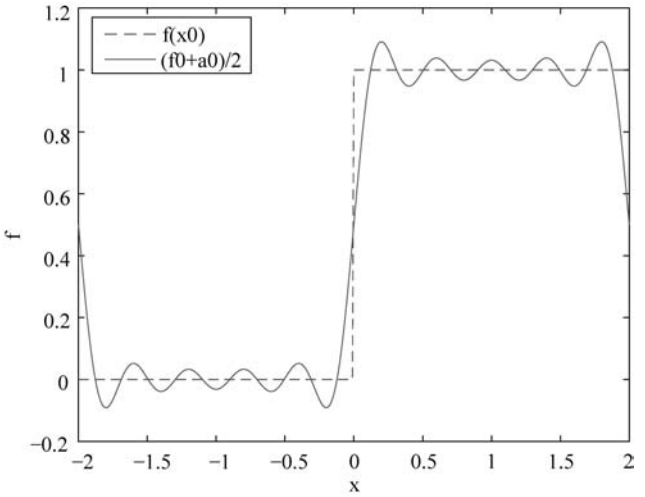

图 12-4　函数 $f(x)$展开所得傅里叶级数与原函数的对照图

由此得到了一般周期函数傅里叶级数展开的方法,进一步地,还可以将其展开为傅里叶级数的复数形式,而且也很容易用 MATLAB 实现,只需要做一些小小的改动就可以了,感兴趣的读者可以尝试一下。

12.8 拓展内容：傅里叶变换的应用——频谱分析

将周期函数展开为傅里叶级数，是将其展开为一个个频率离散的三角函数之和，而傅里叶变换则能够将一个一般函数展开为频率连续的三角函数之和。MATLAB 内置的函数 fft(x)能够实现用快速傅里叶变换计算 X 的离散傅里叶变换。如果 X 是向量，则 fft(X)返回该向量的傅里叶变换。如果 X 是矩阵，则 fft(X)将 X 的各列视为向量并返回每列的傅里叶变换。如果 X 是一个多维数组，则 fft(X)将沿大小不等于 1 的第一个数组维度的值视为向量并返回每个向量的傅里叶变换。

快速傅里叶变换最常见的一个应用便是频谱分析，以 MATLAB 自带的一个例子“使用傅里叶变换求噪声中隐藏信号的频率分量”来考查其用法与功能。

首先需要定义信号的参数，采样频率为 1kHz，信号持续时间为 1.5 秒：

```
Fs = 1000;              %采样频率
T = 1/Fs;               %采样周期
L = 1500;               %信号长度
t = (0:L-1)*T;          %持续时间
```

接下来构造一个信号，包含幅值为 0.7 的 50Hz 正弦量和幅值为 1 的 120Hz 正弦量：

```
S = 0.7*sin(2*pi*50*t) + sin(2*pi*120*t);
```

再用均值为零、方差为 4 的白噪声干扰该信号：

```
X = S + 2*randn(size(t));
```

接下来绘制该噪声信号：

```
plot(1000*t(1:50),X(1:50))
title('被零均值随机噪声污染的信号')
xlabel('时间 t/毫秒')
ylabel('X(t)')
```

运行程序，得到如图 12-5 所示的噪声信号。

通过查看信号 $X(t)$很难确定原信号 $S(t)$的频率分量，而通过 fft 函数进行傅里叶变换后，将能够把该信号分解为无穷多个不同频率的三角函数和。可以猜想由于白噪声分解后的频率应是均匀随机的，而原信号分解后的频率应以其频率分量为主，这样就能够通过分解该信号得到原信号的频率分量。为此，可以尝试对信号进行傅里叶变换：

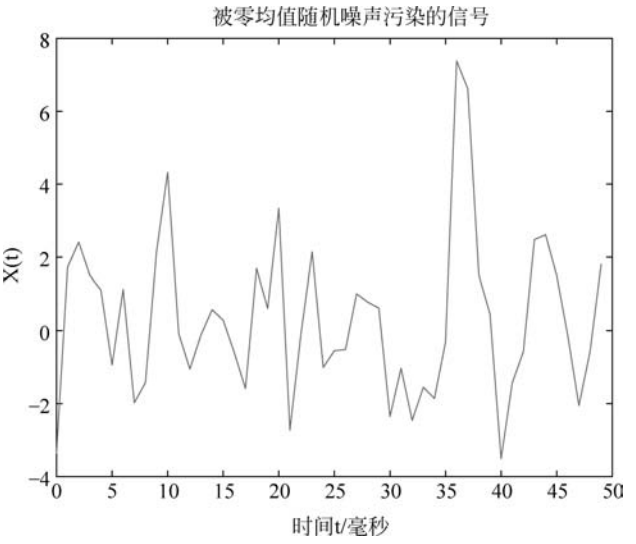

图 12-5　被噪声污染的信号变化图

```
Y = fft(X);
```

接下来计算双侧频谱 $P2$，然后基于 $P2$ 和偶数信号长度 L 计算单侧频谱 $P1$：

```
P2 = abs(Y/L);
P1 = P2(1:L/2+1);
P1(2:end-1) = 2*P1(2:end-1);
```

$P2$ 处要将 Y/L 的原因可以参考 fft 函数的帮助文档中的详细信息：

Y＝fft(X)和 X＝ifft(Y)分别实现傅里叶变换和逆傅里叶变换，对于长度 n 的 X 和 Y，这些变换定义如下：

$$Y(k)=\sum_{j=1}^{n}X(j)W_n^{(j-1)(k-1)}$$

$$X(j)=\frac{1}{n}\sum_{k=1}^{n}Y(k)W_n^{-(j-1)(k-1)}$$

其中，$W_n=e^{-\frac{2\pi i}{n}}$ 为 n 次单位根之一。

可见，fft 函数直接计算出的 $Y(k)$ 并不是振幅谱，观察 $X(j)$ 的公式，对应的幅值为 $Y(k)/N$，因此可以得出需要将 fft 得到的结果除以信号长度 L。

由于经过傅里叶变换后得到的频谱是对称的(一半是信号的负频率)，只需要考查单侧频谱，$P1$ 只需取 $P2$ 的前半部分。进一步地，由于对称的频率幅值相同，取好单侧频谱后需要对幅值进行翻倍。

接下来，定义频域 f 并绘制单侧振幅谱 $P1$：

```
f = Fs*(0:(L/2))/L;
plot(f,P1)
title('X(t)单侧振幅谱')
xlabel('f (Hz)')
ylabel('|P1(f)|')
```

运行程序，得到如图 12-6 所示的频域 f 的单侧振幅谱。

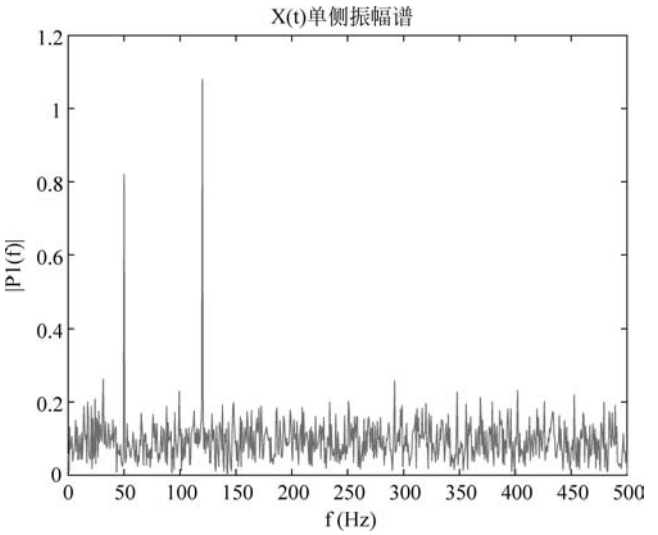

图 12-6　频域 f 的单侧振幅谱

基本与猜想相符，不过由于增加了噪声，幅值并不精确等于 0.7 和 1。一般情况下，较长的信号会产生更好的频率近似值。

接下来尝试将原信号进行傅里叶变换，检索精确幅值 0.7 和 1.0，与之前步骤基本相同：

```
Y = fft(S);
P2 = abs(Y/L);
P1 = P2(1:L/2+1);
P1(2:end-1) = 2*P1(2:end-1);
plot(f,P1)
title('S(t)单侧振幅谱')
xlabel('f (Hz)')
ylabel('|P1(f)|')
```

运行程序，得到如图 12-7 所示的原信号傅里叶变换后的单侧振幅谱。

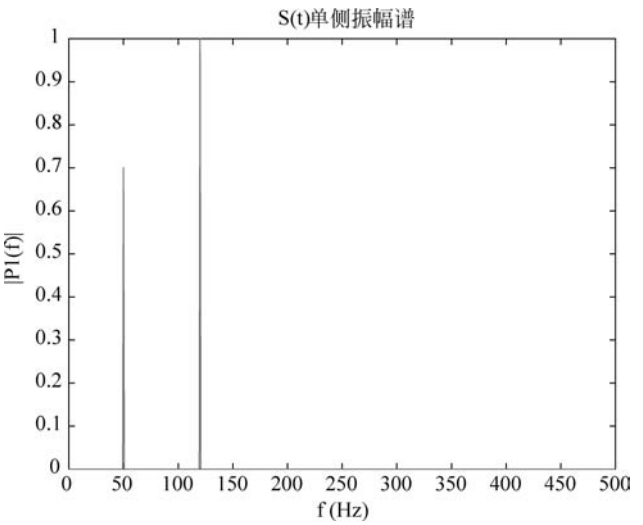

图 12-7　对原信号进行傅里叶变换所得的单侧振幅谱

可以看到，通过 fft 函数对噪声信号进行傅里叶变换，能够较好地得到原噪声中隐藏的信号的频率分量。频谱分析是该函数的一个常见应用，其他的应用有兴趣的读者可以在参考文档中进一步了解。

12.9　上机实践

1. 输入命令判断调和级数的收敛性。

2. 通过 $\gamma=\lim\limits_{n\to+\infty}1+\frac{1}{2}+\cdots+\frac{1}{n}-\ln n$ 了解欧拉常数在 MATLAB 中的表示方法。

3. 判断 $1+\frac{1}{2}-\frac{1}{3}+\frac{1}{4}+\frac{1}{5}-\frac{1}{6}+\cdots$ 的收敛性。

4. 编写利用根值审敛法判断级数收敛性的脚本 $\left(\text{以}\sum\limits_{n=1}^{+\infty}\frac{2+(-1)^n}{2^n}\text{ 为例}\right)$。

5. 求 $\sum\limits_{n=1}^{+\infty}\frac{x^{4n+1}}{4n+1}$ 的和函数。

6. 将函数 $f(x)=\begin{cases}e^x, & -\pi\leqslant x<0\\ 1, & 0\leqslant x<\pi\end{cases}$ 展开成傅里叶级数。

7. 将函数 $f(x)=\begin{cases}-\frac{\pi}{2}, & -\pi\leqslant x<-\frac{\pi}{2}\\ x, & -\frac{\pi}{2}\leqslant x<\frac{\pi}{2}\\ \frac{\pi}{2}, & \frac{\pi}{2}\leqslant x<\pi\end{cases}$ 展开成傅里叶级数。

8. 将函数 $f(x)=\begin{cases}\cos x, & 0\leqslant x<\dfrac{\pi}{2}\\ 0, & \dfrac{\pi}{2}\leqslant x\leqslant\pi\end{cases}$ 分别展开成正弦级数和余弦级数。

9. 更改并简化将一般周期函数展开为傅里叶级数的脚本，使得输出为复数形式，以

$$f(x)=\begin{cases}0, & -3\leqslant x<-1\\ 1, & -1\leqslant x\leqslant 1\\ 0, & 1<x\leqslant 3\end{cases}$$

为例。

10. 将 $f(x)=1-x^2\left(-\dfrac{1}{2}\leqslant x<\dfrac{1}{2}\right)$ 展开为傅里叶级数。