第 5 章

分数阶微积分算子与系统的近似

动态系统是描述很多物理现象的数学模型基础。从系统分析与描述角度看，通常可以将系统分为线性系统和非线性系统。从本章开始将引入系统的概念。5.1 节首先介绍基于 MATLAB 控制系统工具箱的整数阶线性系统建模与分析工具，为以后将介绍的系统行为的比较奠定基础。

在前面章节介绍分数阶微分 $\mathscr{D}^{\alpha}y(t)$ 数值计算的时候，一直假设函数 $y(t)$ 或其采样值是已知的，这样先构造出 $\boldsymbol{y}$ 的采样值向量，然后才能计算出其分数阶的导数与积分。如果分数阶算子 $\mathscr{D}^{\alpha}$ 被一个事先未知的信号驱动，就像在控制系统中分数阶受控对象被控制器驱动那样，则前面讨论的方法是不能计算这样的分数阶动作的。这时，通常需要建立起一个能模拟分数阶行为的装置完成这样的任务。有了这样的装置，就可以将信号馈入该装置，而该装置的输出就是所需的 α 阶微分信号。当然，这样的思想同样适用于分数阶积分信号的获得。

若想要设计这样一个装置，比较有效且常用的思路是设计一个整数阶的线性连续滤波器。如线性连续的传递函数模型 $G(s)$，使得其频域响应尽可能好地逼近原始的分数阶算子模型。

由理论分析可知，分数阶微分环节 s^{α} 的 Bode 幅频特性就是斜率为 20α dB/dec 的斜线，而相频特性就是值为 $\alpha\pi/2$ 的水平直线，而整数阶连续传递函数模型的幅频特性的渐近线为斜率为 $20k$ dB/dec 的斜线，且 k 只能为整数。所以不可能存在一个整数阶传递函数模型，使得在整个频率范围内都能对分数阶行为进行完全一致的逼近，所以只能退而求其次，得出在某个频率段内能逼近分数阶行为的滤波器。

在早期整数阶传递函数逼近的研究中，人们主要采用连分式类近似方法，包括低频与高频部分的连分式近似、Carlson 近似与 Matsuda–Fujii 近似等，5.2 节将给出这类近似与滤波器设计的一般介绍。以现在的观点来看，连分式类近似方法效果很差，不太适合实用，因为没有办法指定感兴趣的频率段，而频率段的选择在分数阶系统的数值仿真中是至关重要的。

法国学者Oustaloup教授及其同事提出的分数阶算子的滤波器近似开启了复杂分数阶系统仿真的新时代[1]。这类滤波器允许用户自行选择感兴趣的频率段与阶次，利用整数阶传递函数模型逼近分数阶微积分算子。这类滤波器的设计将在5.3节给出详细介绍，该节还介绍这类滤波器的改进形式。如果分数阶系统或传递函数中每个分数阶算子都被这类滤波器替代，则可以将整个分数阶系统用高阶整数阶传递函数逼近，也可以用低阶最优模型降阶技术对其逼近。5.4节将给出这些近似的具体实现方法。

对于那些不能由分数阶传递函数标准形式描述的无理系统，例如，如果系统中含有$p^\gamma(s)$项，其中$p(s)$为整数阶传递函数模型。5.5节将介绍三类近似方法：第一类是$1/(\tau s+1)^\nu$隐式模型的时域近似；第二类是用频率响应拟合的方法设计滤波器；第三类是Charef滤波器设计技术。该节还将介绍最优Charef滤波器设计算法及其实现。

除了前面介绍的连续滤波器近似之外，5.6节还介绍用离散滤波器或离散化的滤波器对分数阶算子进行近似[2,3]。不过，从频域响应拟合角度看，拟合模型在高频时常伴有极高的增益，从而使这类滤波器对噪声信号放大效果严重，不一定适合实际应用。

本书将各种通过频域响应拟合得出的高阶整数阶传递函数统称为滤波器设计，因为其作用是让某个函数通过拟合模型后得出所期望的分数阶运算。

5.1　线性整数阶模型的表示与分析

MATLAB的控制系统工具箱对整数阶线性时不变系统的模型输入与分析提供了强大的支持。作为本章的底层支持，这里简要介绍在MATLAB环境下线性整数阶系统的建模与分析工具。

5.1.1　数学模型输入与处理

整数阶线性时不变（linear time invariant，LTI）连续系统一般由传递函数和状态方程表示。本节先介绍这两种模型的数学形式，再介绍利用MATLAB输入这些模型的方法。

定义 5-1 ▶ 整数阶传递函数

整数阶线性时不变系统的传递函数模型一般形式为

$$G(s)=\frac{b_1s^m+b_2s^{m-1}+\cdots+b_ms+b_{m+1}}{a_1s^n+a_2s^{n-1}+\cdots+a_ns+a_{n+1}}\tag{5-1-1}$$

其中，若$m\leqslant n$。若$m>n$，则称为物理不可实现系统，本书不考虑这样的系统。

定义 5-2 ▶ 整数阶状态方程

整数阶线性时不变系统的状态方程模型一般形式为

$$\begin{aligned}\boldsymbol{x}'(t) &= \boldsymbol{A}\boldsymbol{x}(t) + \boldsymbol{u}(t) \\ \boldsymbol{y}(t) &= \boldsymbol{C}\boldsymbol{x}(t) + \boldsymbol{D}\boldsymbol{u}(t)\end{aligned} \tag{5-1-2}$$

其中，$\boldsymbol{x}(t)$、$\boldsymbol{u}(t)$ 和 $\boldsymbol{y}(t)$ 分别称为系统的状态变量、输入与输出向量。$\boldsymbol{A}$ 为 $n\times n$ 矩阵，$\boldsymbol{B}$、$\boldsymbol{C}$ 和 $\boldsymbol{D}$ 分别为 $n\times p$、$q\times n$ 和 $q\times p$ 矩阵；n 为状态变量的个数，又称为系统的阶次；p 和 q 分别为系统的输入、输出信号路数。

系统的传递函数模型有两种输入方法：

（1）先用 `num=[`b_1`,`b_2`,`$\cdots$`,`b_{m+1}`]`、`den=[`a_1`,`a_2`,`$\cdots$`,`a_{n+1}`]` 命令输入传递函数的分子、分母多项式，然后用 G`=tf(num,den)` 命令构造传递函数模型。

（2）由 s`=tf('s')` 命令定义 Laplace 算子 s，然后在 MATLAB 下用表达式输入传递函数模型。

若已知状态方程模型，即已知 $\boldsymbol{A}$、$\boldsymbol{B}$、$\boldsymbol{C}$ 和 $\boldsymbol{D}$ 矩阵，则由 G`=ss(`$\boldsymbol{A}$`,`$\boldsymbol{B}$`,`$\boldsymbol{C}$`,`$\boldsymbol{D}$`)` 命令可以输入系统的状态方程模型。

如果已知 $G_1(s)$ 和 $G_2(s)$ 两个模块串联连接，且这两个模块分别用 G_1、G_2 两个变量表示，则串联系统总的模型可以由 G`=`G_2`*`G_1 直接求出；如果这两个模块并联连接，则可以由 G`=`G_1`+`G_2 直接求出总模型；如果这两个模块负反馈连接，其中前向通路为 G_1，反向通路为 G_2，则总系统可以由 G`=feedback(`G_1`,`G_2`)` 命令直接求出，而求正反馈总模型应该用 G`=feedback(`G_1`,`G_2`,1)` 命令。

5.1.2　时域与频域响应

若已知系统的模型变量 G，则不论是传递函数还是状态方程模型，均可以对该系统进行分析。例如，由 `bode(`G`)`、`nyquist(`G`)`、`nichols(`G`)` 函数可以分别绘制系统的 Bode 图、Nyquist 图和 Nichols 图；由 `step(`G`)`、`impulse(`G`)` 语句可以绘制系统的阶跃或冲激响应；若已知时间向量 $\boldsymbol{t}$ 和激励信号的采样点训练 $\boldsymbol{u}$，则可以调用 `lsim(`G`,`$\boldsymbol{u}$`,`$\boldsymbol{t}$`)` 直接绘制时域响应曲线，或由 $\boldsymbol{y}$`=lsim(`G`,`$\boldsymbol{u}$`,`$\boldsymbol{t}$`)` 获得时域响应数据向量 $\boldsymbol{y}$。

5.1.3　分数阶线性系统的建模与分析

作者的 FOTF 工具箱是在仿照上面的建模与分析语句调用格式的基础上开发的，系统分析函数与调用格式尽量与上面介绍的函数保持一致，可以直接调用。后面还将专门介绍 FOTF 工具箱的使用方法。

5.2 基于连分式的几种近似方法

在早期的研究中，人们经常采用连分式展开的形式设计滤波器，而连分式技术也经常被认为是逼近某些非线性函数的有效工具。

定义 5-3 ▶ 连分式

给定函数 $f(x)$ 的连分式展开的一般形式为

$$f(x)=b_1+\cfrac{(x-a)^{c_1}}{b_2+\cfrac{(x-a)^{c_2}}{b_3+\cfrac{(x-a)^{c_3}}{b_4+\cfrac{(x-a)^{c_4}}{b_5+\cfrac{(x-a)^{c_5}}{\cdots}}}}} \tag{5-2-1}$$

式中，b_i 为常数；c_i 为有理数；a 为连分式展开的参考点。对整数阶连分式逼近而言，通常令 $c_i=1$。

5.2.1 连分式近似

从理论上看是不可能直接对分数阶算子 $s^{-\alpha}$ 进行连分式近似的，所以，研究者分别在高频和低频段对 $G_1(s)=1/(1+Ts)^\alpha$ 和 $G_2(s)=(1+1/s)^\alpha$ 进行连分式近似。其中，当 $\omega T\gg 1$ 时用 $G_1(s)$ 逼近 $1/s^\alpha$；而 $\omega\ll 1$ 时，用 $G_2(s)$ 逼近该积分算子。

MATLAB 的符号运算引擎 MuPAD 提供了底层函数 `contfrac()`，作者编写了该函数的一个接口函数，对给定函数进行连分式近似。

```
function [F,r]=contfrac(f,s,n,a)
   arguments, f(1,1), s(1,1), n(1,1), a(1,1), end
   F=feval(symengine,'contfrac',f,[inputname(2) '=' num2str(a)],n);
   if nargout==2
      r=feval(symengine,'contfrac::rational',F);
end, end
```

该函数的调用格式为 `[`f_1`,`r`]=contfrac(`f`,`s`,`n`,`a`)`，其中，f 是被近似函数的符号表达式，s 为自变量，n 为连分式的级次，a 为参考点。返回的 f_1 为连分式的符号表达式，r 为有理式。

遗憾的是，由于近期版本 MuPAD 底层的 `contfrac()` 函数似乎不再支持求取函数的连分式表达式，这里给出的接口函数不能正常工作。如果想得出连分式可以尝试早期的版本，如 MATLAB 2018b，或改用 Padé 近似技术。

例 5-1 试分别在高频与低频段用连分式近似分数阶积分算子 $1/\sqrt{s}$，并评价其逼近效果。

解　选择参考点 $a=2$，连分式级次为 $n=9$，可以得出 4 阶高阶近似模型。

```
>> syms s; T1=1/(1+s)^0.5; [c1,G2]=contfrac(T1,s,9,2)
```

得出的连分式模型为

$$c_1(s)\approx\frac{\sqrt{3}}{3}+\cfrac{s-2}{-6\sqrt{3}+\cfrac{s-2}{-\frac{2\sqrt{3}}{9}+\cfrac{s-2}{-54\sqrt{3}+\cfrac{s-2}{-\frac{2\sqrt{3}}{45}+\cfrac{s-2}{-150\sqrt{3}+\cfrac{s-2}{-\frac{2\sqrt{3}}{105}+\cdots}}}}}}$$

对应的有理函数近似模型为

$$G_2(s)=\frac{s^4+112s^3+1464s^2+4864s+4240}{9\sqrt{3}s^4+288\sqrt{3}s^3+1944\sqrt{3}s^2+4032\sqrt{3}s+2448\sqrt{3}}$$

可以看出，这里得出的近似模型与下面显示的文献 [4] 中的模型不一致：

$$G_{\mathrm{h}}(s)=\frac{0.3513s^4+1.405s^3+0.8433s^2+0.1574s+0.008995}{s^4+1.333s^3+0.478s^2+0.064s+0.002844}$$

$$G_{\mathrm{l}}(s)=\frac{s^4+4s^3+2.4s^2+0.448s+0.0256}{9s^4+12s^3+4.32s^2+0.576s+0.0256}$$

一个可能的原因是选择的参考点不同。近似模型和原始模型 $1/\sqrt{s}$ 的 Bode 图比较可以由下面语句得出，如图 5-1 所示，其中，图中标记的英文字符是自动生成的，Bode Diagram 为 Bode 图，Magnitude (dB) 为幅值，单位为分贝 (dB)，Phase (deg) 为相位，单位为度，Frequency (rad/s) 为频率，单位为 rad/s。

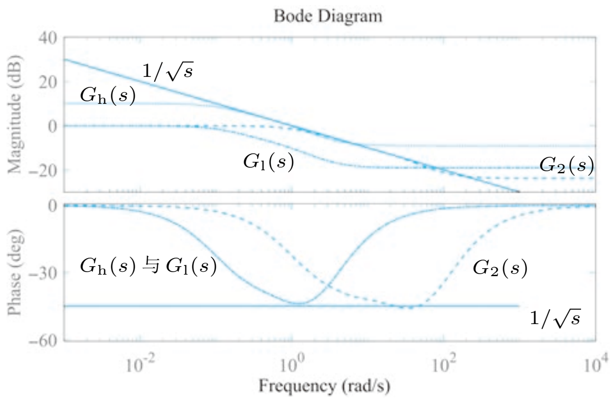


图 5-1　连分式拟合的 Bode 图比较

```
>> w=logspace(-3,3); s=fotf('s'); G0=s^-0.5; s=tf('s');
   G2=(s^4+112*s^3+1464*s^2+4864*s+4240)/(9*3^(1/2)*s^4+...
     288*3^(1/2)*s^3+1944*3^(1/2)*s^2+4032*3^(1/2)*s+2448*3^(1/2));
```

```
n=[0.3513 1.405 0.8433 0.1574 0.008995];
d=[1 1.333 0.478 0.064 0.002844]; Gh=tf(n,d);
n=[1 4 2.4 0.448 0.0256]; d=[9 12 4.32 0.576 0.0256];
Gl=tf(n,d); H=bode(G0,w); bode(H,G2,Gh,Gl)
bode(H,'-',G2,'--',Gh,':',Gl,'-.')
```

可以看出，模型 $G_2(s)$ 的拟合效果比 $G_{\rm h}(s)$ 的稍有改善，但是在低频段较差，从整个频率段的拟合看并不是很好。

例 5-2 试比较几个滤波器对 $f(t)=\mathrm{e}^{-t}\sin(3t+1)$ 信号的 0.5 阶积分的近似效果，并与 Grünwald–Letnikov 分数阶积分的计算结果进行比较。

解 可以用下面的语句得出这几个滤波器的滤波结果，这些结果是由 MATLAB 控制系统工具箱 `lsim()` 函数计算出来的，滤波结果与理论值曲线如图 5-2 所示。可以看出，这样的滤波结果与理论值的差距太大，所以这种滤波器不能真正用于分数阶系统的仿真。

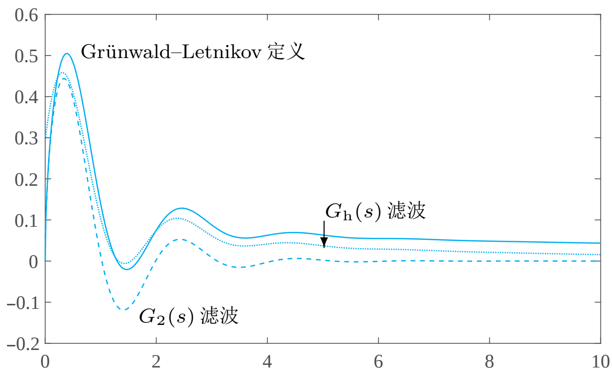


图 5-2 0.5 阶积分的比较

```
>> t=0:0.01:10; y=exp(-t).*sin(3*t+1);
   y0=glfdiff9(y,t,-0.5,5); y1=lsim(G2,y,t);
   y2=lsim(Gh,y,t); plot(t,y0,'-',t,y1,'--',t,y2,':')
```

5.2.2 Carlson 近似

另一个基于连分式的近似方法是 Carlson 方法[4]。Carlson 方法的目标是对给定整数阶基础模型 $G(s)$ 分数次幂的形式找到整数阶近似模型 $H(s)$，即

$$H(s)\approx G^{\alpha}(s) \tag{5-2-2}$$

算法 5-1 ▶ Carlson 滤波器设计算法

（1）输入整数阶基础模型 $G(s)$ 与分数阶幂次 α。

（2）记 $q=\alpha$，$m=q/2$，并令初始近似模型 $H_0(s)=1$。

（3）选择递推次数为 n。

（4）在递推的每一步中，从已知的模型 $H_{i-1}(s)$ 得出下一个模型 $H_i(s)$。

$$H_i(s)=H_{i-1}(s)\frac{(q-m)H_{i-1}^2(s)+(q+m)G(s)}{(q+m)H_{i-1}^2(s)+(q-m)G(s)} \tag{5-2-3}$$

（5）直接迭代得出的近似模型阶次比较高，但该模型有可以对消的零极点，所以应该对得出的近似模型做最小实现处理。

基于上述的算法，可以编写出下面的 MATLAB 函数设计 Carlson 滤波器：

```
function H=carlson_fod(alpha,G,iter,epsx)
   arguments
      alpha(1,1) double, G(1,1)
      iter(1,1) {mustBePositiveInteger}=2, epsx(1,1)=1e-4;
   end
   q=alpha; m=q/2; H=1;
   for i=1:iter
      H=minreal(H*((q-m)*H^2+(q+m)*G)/((q+m)*H^2+(q-m)*G),epsx);
end, end
```

该函数的调用格式为 H=`carlson_fod`(α,G,n,ϵ)，其中 G 为整数阶基础模型，α 是分数阶幂次，n 是递推的次数，H 为设计出的 Carlson 滤波器。请注意，滤波器的实际阶次随着递推次数增加而迅速增加，一般情况下选择 $n=2$ 即可。用户还可以选择较大的误差容限 ϵ，例如，取值 10^{-4}，对得出的近似模型做最小实现处理。

例 5-3　试用 Carlson 算法给 $1/\sqrt{s}$ 设计一个滤波器，并观察频域响应的拟合效果，另外，求给定函数 $f(t)=\mathrm{e}^{-t}\sin(3t+1)$ 的 0.5 阶积分，并评价其精度。

解　对这个具体的问题而言，基础模型为 $G(s)=1/s$，幂次为 $\alpha=0.5$，选择递推次数为 2，这样可以设计出 Carlson 滤波器：

```
>> s=tf('s'); G=1/s; alpha=0.5;
   H2=carlson_fod(alpha,G,2,1e-4)    %选择一个较大的误差容限
```

这里得出的模型如下，与文献 [4] 给出的是等效的：

$$H_2(s)=\frac{0.1111s^4+4s^3+14s^2+9.333s+1}{s^4+9.333s^3+14s^2+4s+0.1111}$$

可以由下面的语句绘制出频域响应拟合曲线，如图 5-3 所示。可以看出，如果选择感兴趣的频率区间 $10^{-1},10^{1}$ rad/s，则滤波器的逼近效果比较理想。

```
>> G3=tf([1 36 126 84 9],[9 84 126 36 1]); w=logspace(-3,3);
   s=fotf('s'); H=bode(s^-0.5,w); bode(H,'-',H2,'--')
```

如果用这些滤波器计算给定信号的 0.5 阶积分，得出的结果如图 5-4 所示。可见，得出的滤波效果与 Grünwald–Letnikov 定义得出的理论值几乎完全一致。

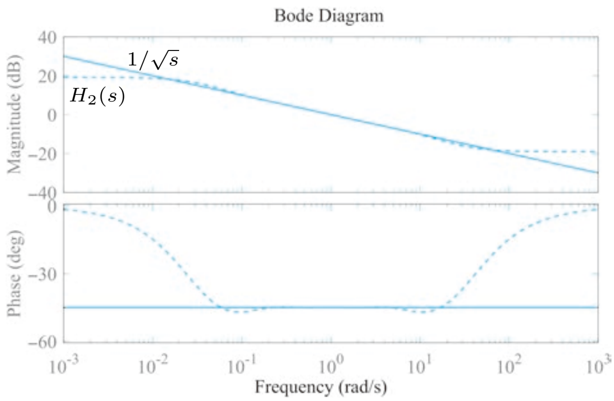


图 5-3　两个滤波器的 Bode 图比较

```
>> t=0:0.01:10;
   y=exp(-t).*sin(3*t+1); y0=glfdiff9(y,t,-0.5,5);
   y1=lsim(H2,y,t); plot(t,y0,'-',t,y1,'--')
```

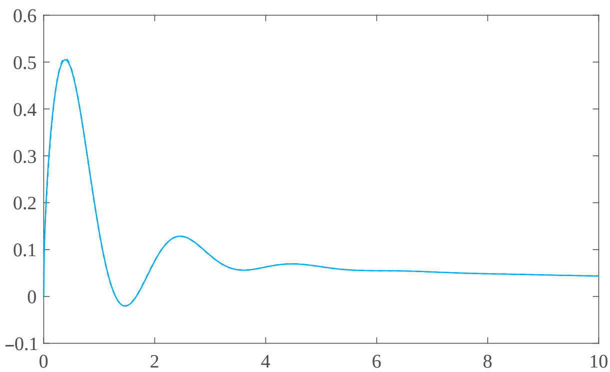


图 5-4　0.5 阶积分的计算效果比较

如果时间区间增加到 $t \in (0, 100)$，则可以得出滤波器与理论值的时域响应拟合比较，如图 5-5 所示。

```
>> t=0:0.01:100;
   y=exp(-t).*sin(3*t+1); y0=glfdiff9(y,t,-0.5,3);
   y1=lsim(H2,y,t); plot(t,y0,'-',t,y1,'--')
```

不过，这时的拟合效果在 t 较大时产生很大误差，而 t 很大对应于低频段，由此可知，误差产生的原因是低频处拟合不佳。结合图 5-3 中的频域响应拟合，可以看出，拟合区域下限选择为 10^{-1} rad/s 是不够的，应该扩展区间下限到 $\omega < 10^{-2}$ rad/s 甚至更低，不过这样的要求在 Carlson 滤波器的设计中是做不到的，所以需要允许用户自行选择拟合区间的更好算法。

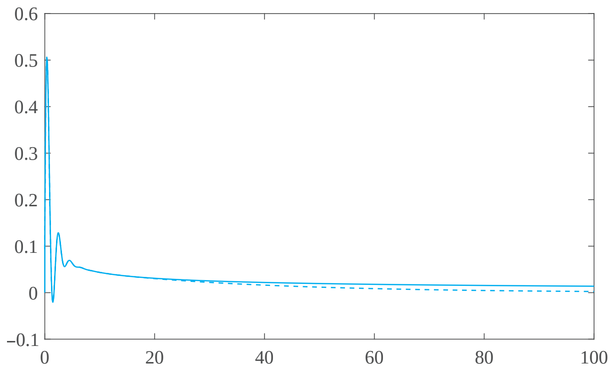


图 5-5　更大时间范围的 0.5 阶积分拟合效果

5.2.3　Matsuda–Fujii 近似

Matsuda 和 Fujii 提出了一种利用连分式技术近似无理传递函数 $G(s)$ 的整数阶传递函数的逼近方法[4,5]。遗憾的是，其原始算法的数学描述有误，这里只给出改正后的算法。

算法 5-2 ► 改正后的 Matsuda–Fujii 滤波器设计算法

（1）选择频率向量 ω_i（$i=1,2,\cdots,n$），并计算出原模型的频域响应数据 $G_0(\mathrm{j}\omega_i)$。

（2）选择一个初始序列 $v_1(\omega_k)=|G_0(\mathrm{j}\omega_k)|, k=1,2,\cdots,n$。

（3）对 k 进行循环，提取 $a_k=v_k(\omega_k)$，并由下式更新 $v_{k+1}(\omega)$：

$$v_{k+1}(\omega_j)=\frac{\omega_j-\omega_k}{v_k(\omega_j)-v_k(\omega_k)},\quad j=1,2,\cdots,n \tag{5-2-4}$$

（4）由得出的系数 a_k（$k=1,2,\cdots,n$）构造 Matsuda–Fujii 滤波器 $H(s)$。

$$H(s)=a_1+\cfrac{s-\omega_1}{a_2+\cfrac{s-\omega_2}{a_3+\cfrac{s-\omega_3}{a_4+\cfrac{s-\omega_4}{a_5+\cdots}}}} \tag{5-2-5}$$

下面给出 Matsuda–Fujii 滤波器算法的 MATLAB 实现：

```
function G=matsuda_fod(G0,n,wb,wh)
   arguments, G0,n,wb(1,1),wh(1,1){mustBeGreaterThan(wh,wb)}, end
   if nargin==2, f=G0; w=n; n=length(w);
   else
      if isa(G0,'double'), s=fotf('s'); G0=s^G0; end
      w=logspace(log10(wb),log10(wh),n); f=mfrd(G0,w);
```

```
    end
    v=abs(f(:).'); s=tf('s'); n=length(w);
    for k=1:n, a(k)=v(k); v=(w-w(k))./(v-v(k)); end
    G=a(n); for k=n-1:-1:1, G=a(k)+(s-w(k))/G; end
end
```

说明 5-1 Matsuda–Fujii 滤波器设计函数

（1）该函数有下面 3 种调用格式：

G=matsuda_fod($\boldsymbol{f}$,$\boldsymbol{\omega}$)，其中需要给定原系统的频域响应数据。

G=matsuda_fod(γ,n,$\omega_{\rm b}$,$\omega_{\rm h}$)，在频率 $\omega_{\rm b},\omega_{\rm h}$ 内对 s^γ 算子进行 n 点近似。

G=matsuda_fod(G_0,n,$\omega_{\rm b}$,$\omega_{\rm h}$)，在频率 $\omega_{\rm b},\omega_{\rm h}$ 内对 G_0 进行 n 点近似。

（2）由于 $\omega_{\rm b}$ 与 $\omega_{\rm h}$ 是选择频率边界内的内点，所以实际拟合的频率段要比指定的 $\omega_{\rm b}$,$\omega_{\rm h}$ 稍大些。

（3）如果想得到较好的拟合，则应该选择奇数 n，这样，实际滤波器的阶次等于 $(n-1)/2$。

（4）虽然拟合的目标是频域响应的幅值拟合，但相位的拟合效果也比较好。

例 5-4 试为 $1/\sqrt{s}$ 设计一个 Matsuda–Fujii 滤波器，并评价其拟合效果。

解 如果选择频率段 $10^{-1},10^1$ rad/s 和 $10^{-2},10^2$ rad/s，则可以直接设计出 Matsuda–Fujii 滤波器，得出的 Bode 图比较如图 5-6 所示。

```
>> s=fotf('s'); w=logspace(-3,3); G1=matsuda_fod(-0.5,9,1e-1,1e1)
   G2=matsuda_fod(-0.5,9,1e-2,1e2)     %更大频率段的滤波器设计
   H=bode(s^-0.5); bode(H,G1,'--',G2,':',{1e-3,1e3})
```

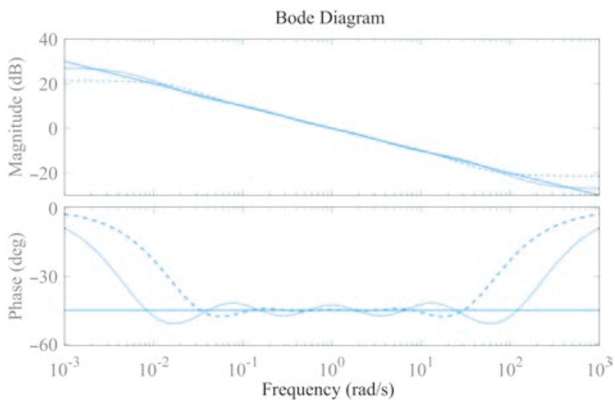


图 5-6 Bode 图比较

用两种调用格式设计出的滤波器是完全一致的，与文献 [4] 中给出的结果也是完全一致的。

$$G_1(s)=\frac{0.0855s^4+4.876s^3+20.84s^2+13s+1}{s^4+13s^3+20.84s^2+4.876s+0.0855}$$

$$G_2(s)=\frac{0.04401s^4+8.142s^3+58.85s^2+30.76s+1}{s^4+30.76s^3+58.85s^2+8.142s+0.04401}$$

从频域响应比较可见，低频拟合效果与Carlson滤波器接近，可以说，低频拟合的问题尚未解决。好在Matsuda–Fujii滤波器的频率段与阶次都是可以人为选定的，所以实际的频域响应拟合效果有很大的改善余地。例如，可以增加滤波器的阶次，并增大拟合的频率范围，则由下面语句直接设计8阶滤波器(取$n=17$)，这时得出的Bode图如图5-7所示。可见，这样设计的滤波器的效果有明显的改善，其Bode图拟合的频率范围更宽，拟合效果也更好。

```
>> w=logspace(-3,3); G1=matsuda_fod(-0.5,9,1e-2,1e2);
   G2=matsuda_fod(-0.5,17,1e-2,1e2), H=bode(s^-0.5);
   bode(H,G1,'--',G2,':',{1e-3,1e3})
```

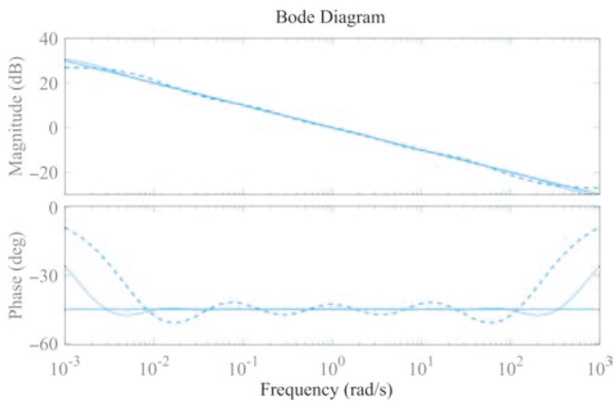


图 5-7　Bode 图比较

有了滤波器，就可以将已知信号馈入该滤波器，得出该信号的0.5阶积分，并和数值计算的精确解相比较，得出如图5-8所示的比较结果。可以看出，G_2滤波器与数值计算所得的曲线几乎重合，而G_1滤波器的拟合效果(虚线)比较差。

```
>> t=0:0.01:100; y=exp(-t).*sin(3*t+1);
   y0=glfdiff9(y,t,-0.5,3); y1=lsim(G1,y,t); y2=lsim(G2,y,t);
   plot(t,y0,'-',t,y1,'--',t,y2,':')
```

5.2.4　拟合效果与滤波器参数选择的关系

前面介绍了各种滤波器的设计方法，也在时域响应角度比较了滤波器阶次与感兴趣频段对逼近效果的影响，得出了一些定性结论。这里将结合Laplace变换的性质，更准确地解释这些滤波器参数对时域响应拟合的影响。

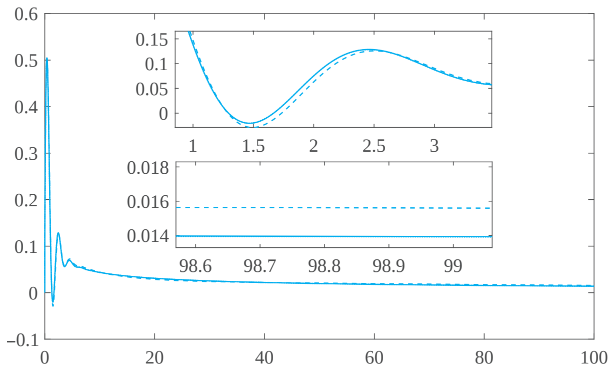


图 5-8　时域响应拟合效果

定理 5-1 ▶ 初值定理与终值定理

如果 $F(s)=\mathscr{L}\ f(t)$ ，则 Laplace 变换的初值定理与终值定理为

$$\lim_{s\to\infty} sF(s)=f(0),\quad \lim_{s\to 0} sF(s)=f(\infty) \tag{5-2-6}$$

因为频率与 s 的关系为 $s=\mathrm{j}\omega$，所以，由定理 5-1 可知，若想在初始时刻（即 t 比较小时）得到很好的拟合效果，则应该改善高频段的拟合，即提高感兴趣频率区域的上限 ω_h；如果想得到更好的时域响应精度，则应该减小感兴趣频率区间的下限 ω_b；若想得到更好的整体的时域响应精度，则应该使得中频段频域响应曲线更好地逼近理论值的斜线，而增加滤波器的阶次是一种有效的改善方法。增加滤波器的阶次当然会增大计算量，不过这里增加的计算量对当前的计算机软硬件系统而言并不会增加很大的负担。

5.3　Oustaloup 滤波器近似

前面已经通过例子演示过，由于基于连分式的滤波器设计方法在频域响应拟合上不甚理想，尤其是连分式类方法不允许使用者自由选择合适的拟合频率段，使得这样的滤波器应用大打折扣。本节将介绍更实用的 Oustaloup 滤波器。

5.3.1　常规的 Oustaloup 近似

假设感兴趣的频率段为 $\omega_\mathrm{b},\omega_\mathrm{h}$ ，可以考虑用图 5-9 中给出的一组折线逼近分数阶微积分算子幅频响应的直线特性。法国学者 Oustaloup 教授等基于这样的想法提出了滤波器设计的方法[6]，本书中称之为 Oustaloup 滤波器。所有这些折线都是由整数阶的零点与极点生成的，使得幅频特性渐近线的斜率在 0 dB/dec 与 −20 dB/dec 之间交替变化，这样频域响应本身会很好地逼近一条斜线。

定义 5-4 ▶ Oustaloup 滤波器

Oustaloup 滤波器的标准形式为[6]

$$G(s)=K\prod_{k=1}^{N}\frac{s+\omega_k'}{s+\omega_k} \tag{5-3-1}$$

式中，若 $k=1,2,\cdots,N$，零点、极点和增益可以如下计算：

$$\omega_k'=\omega_{\rm b}\omega_{\rm u}^{(2k-1-\gamma)/N},\quad \omega_k=\omega_{\rm u}^{2\gamma/N}\omega_k',\quad K=\omega_{\rm h}^{\gamma} \tag{5-3-2}$$

且

$$\omega_{\rm u}=\sqrt{\omega_{\rm h}/\omega_{\rm b}} \tag{5-3-3}$$

根据上面的公式，可以总结出设计 Oustaloup 滤波器的算法。

算法 5-3 ▶ 标准 Oustaloup 滤波器设计

（1）输入分数阶阶次 γ，选择频率段 $\omega_{\rm b}$，$\omega_{\rm h}$ 与滤波器阶次 N。

（2）由式（5-3-3）计算 $\omega_{\rm u}$，再通过向量点运算由式（5-3-2）计算 ω_k，ω_k' 与 K。

（3）通过式（5-3-1）构造 Oustaloup 滤波器 $G(s)$。

基于上述算法可以直接编写出 Oustaloup 滤波器的 MATLAB 函数如下：

```
function G=ousta_fod(gam,N,wb,wh)
   arguments
      gam(1,1) double, N(1,1) {mustBePositiveInteger}=9
      wb(1,1) double=1e-4;
      wh(1,1) double {mustBeGreaterThan(wh,wb)}=1e4;
   end
   if round(gam)==gam, G=tf('s')^gam;
   else
      k=1:N; wu=sqrt(wh/wb);
      wkp=wb*wu.^((2*k-1-gam)/N); wk=wu^(2*gam/N)*wkp;
      G=zpk(-wkp,-wk,wh^gam);
```

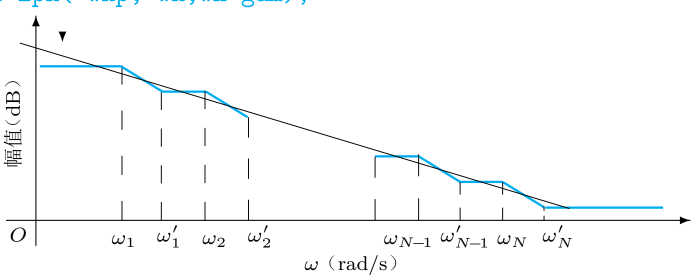


图 5-9　Oustaloup 滤波器的分段折线逼近

```
end, end
```

该函数的调用格式为 G=`ousta_fod(`γ,N,ω_b,ω_h`)`，其中，γ 为分数阶算子的阶次，N 为滤波器的阶次，ω_b 与 ω_h 为用户选择的感兴趣频率段的下限与上限。一般情况下，所设计的滤波器 G 的 Bode 图在指定频段效果很好，但在该频率段之外效果不好。

说明 5-2 Oustaloup 滤波器

（1）这里给出的滤波器实现避免了 $\omega_\mathrm{b}\omega_\mathrm{h}=1$ 限制，两端均可以独立选择。

（2）如果 γ 为整数，可以直接构造整数阶微积分器 $G(s)=s^\gamma$。

（3）这里的 γ 既可为正也可为负，分别表示微分与积分。

（4）γ 的绝对值允许大于 1，例如 $\gamma=3.7$ 仍可以直接拟合，不过在实际应用中，建议将阶次范围限定在 $-1<\gamma<1$，其余的部分用整数阶微积分算子表示，例如，$s^{3.7}=s^3s^{0.7}$，或 $s^{3.7}=s^4s^{-0.3}$。

（5）尽管从图 5-9 上看近似的幅频特性像折线，这些折线实际上是 Bode 图的渐近线，如果参数选择合适，实际的 Bode 图本身应该很接近斜线。

将 $y(t)$ 信号馈入 Oustaloup 滤波器，则滤波器的输出信号为 ${}_0^{\mathrm{RL}}\mathscr{D}_t^\gamma y(t)$，即输入信号的 Riemann–Liouville 分数阶导数或积分。

例 5-5 假设感兴趣的频率段为 $\omega_\mathrm{b}=0.01$，$\omega_\mathrm{h}=1000\,\mathrm{rad/s}$，可以选择不同的阶次设计 Oustaloup 滤波器。如果输入函数为 $f(t)=\mathrm{e}^{-t}\sin(3t+1)$，试设计 0.5 阶积分的滤波器，并评价滤波器的近似效果。

解 可以用下面的语句直接设计 5 阶 Oustaloup 滤波器 $G_1(s)$ 和 Matsuda–Fujii 滤波器 $G_2(s)$：

```
>> G1=ousta_fod(-0.5,5,0.01,1000), G1a=zpk(G1)
   G2=matsuda_fod(-0.5,11,0.01,1000), G2a=zpk(G2)
   s=fotf('s'); G=s^-0.5; w=logspace(-4,5);
   H=bode(G,w); bode(H,'-',G1,'--',G2,':')
```

设计出来的两个滤波器的零极点形式分别为

$$G_{1a}(s)=\frac{0.031623(s+562.3)(s+56.23)(s+5.623)(s+0.5623)(s+0.05623)}{(s+177.8)(s+17.78)(s+1.778)(s+0.1778)(s+0.01778)}$$

$$G_{2a}(s)=\frac{0.013865(s+1792)(s+71.46)(s+5.81)(s+0.503)(s+0.03427)}{(s+291.8)(s+19.88)(s+1.721)(s+0.1399)(s+0.00558)}$$

可以绘制出该滤波器的 Bode 图，如图 5-10 所示，同时绘制出分数阶算子 $s^{-0.5}$ 的 Bode 图，分数阶算子由 `fotf()` 函数定义，该函数将在第 6 章详细讨论。可以看出，对本例而言，Matsuda–Fujii 滤波器的拟合带宽要宽一些。

还可以得出两个滤波器的滤波结果与 Grünwald–Letnikov 0.5 阶积分的数值解，如

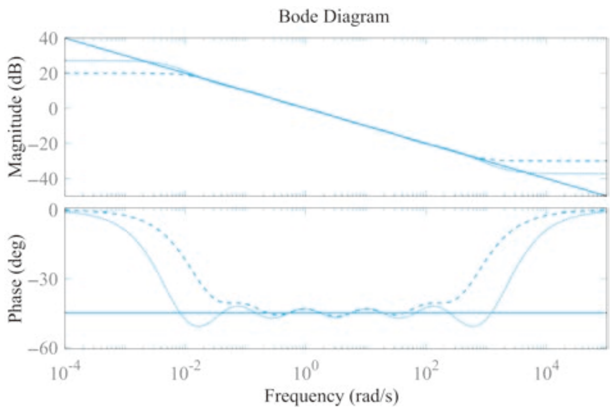


图 5-10　两个滤波器的 Bode 图拟合

图 5-11 所示。可以看出，这样得出的分数阶积分不是很精确，两个滤波器的近似效果差不多。若想改善近似效果，建议增大感兴趣频域区间，并增加阶次。

```
>> t=0:0.001:pi; y=exp(-t).*sin(3*t+1);
   y1=lsim(G1,y,t); y2=lsim(G2,y,t);
   y0=glfdiff9(y,t,-0.5,3); plot(t,y0,t,y1,'--',t,y2,':')
```

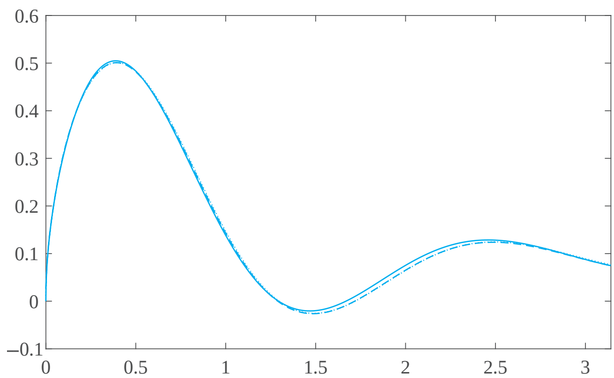


图 5-11　分数阶积分的近似效果

例 5-6　现在考虑为 Heaviside 函数设计 0.5 阶导数的 Oustaloup 滤波器，并假设时间范围很大，试观察 Oustaloup 滤波器参数对导数计算精度的影响。

解　Heaviside 函数的频率为零。现在选择感兴趣的频率段 (0.01, 1000) rad/s，并假设仿真的时间区间为 (0,20) s，就可以使用下面的语句设计 Oustaloup 滤波器。这样，用两种方法得出分数阶导数曲线，如图 5-12 所示。可以看出，当 $t=2$ 或 t 很大时用滤波器的计算结果很不精确。

```
>> t=0:0.01:20;
   y=ones(size(t)); G=ousta_fod(0.5,5,0.01,1000),
   y1=lsim(G,y,t); y2=glfdiff9(y,t,0.5,3);
```

```
plot(t,y1,t,y2,'--'), ylim([0,1])
```

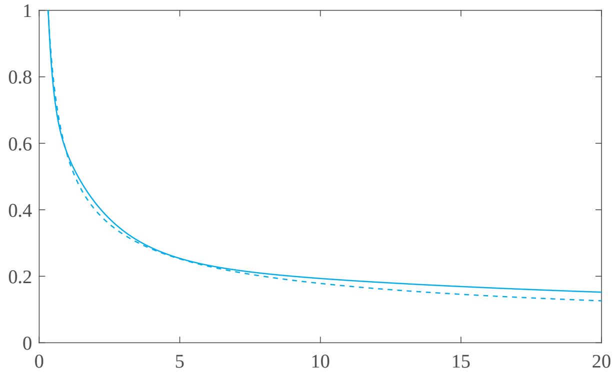


图 5-12　Heaviside 函数的分数阶导数

选择考虑增加 Oustaloup 滤波器的阶次，则由下面的语句可以重新计算分数阶导数，虽然在局部拟合上有所改进，在 t 较大时误差仍然很大。这意味着只改变滤波器的阶次并不能改进大时间区间的近似效果。

```
>> G=ousta_fod(0.5,9,0.01,1000); y1=lsim(G,y,t);
   plot(t,y1,t,y2,'--'), ylim([0,1])
```

因为时间 t 很大时存在大误差，这意味着低频段 Oustaloup 滤波器的拟合不好，所以应该降低 $\omega_{\rm b}$ 的值，例如选择 $\omega_{\rm b}=0.001\,{\rm rad/s}$，这时在新滤波器下分数阶导数的计算结果如图 5-13 所示。可以看出已经得到了很好的计算精度。

```
>> G=ousta_fod(0.5,9,0.001,1000); y1=lsim(G,y,t);
   plot(t,y1,t,y2,'--'), ylim([0,1])
```

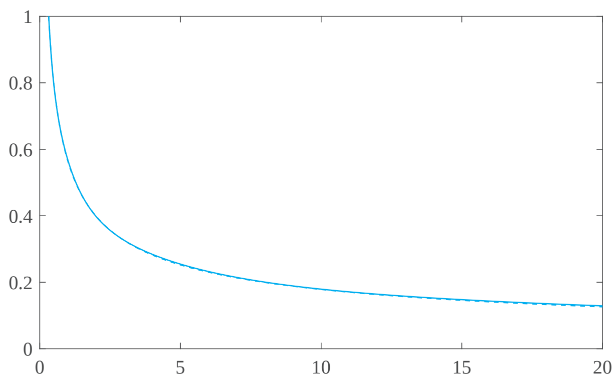


图 5-13　改进的计算结果

如果想进一步改进计算精度，则可以考虑选择更小的频率下限 $\omega_{\rm b}$，并选择更高的滤波器阶次。

例 5-7　从前面的例子可以看出，若想提高计算精度，则可以增大感兴趣的频率段，那么如何选择滤波器的阶次呢？

解　例 5-5 演示过，如果感兴趣的频率段为 $10^{-2}, 10^{3}$ rad/s，则 5 阶 Oustaloup 滤波器效果已经很好了。一般情况下，如果频率段每增加 1 个十倍频程，则应该至少增加阶次 1 到 2 阶。例如，如果感兴趣频率段选择为 $10^{-4}, 10^{4}$ rad/s，即频率段增加了 3 个十倍频程，则需要设计一个 11 阶的滤波器。一般情况下，如果计算机资源允许，不妨选择一个更高的阶次。不同阶次滤波器的 Bode 图如图 5-14 所示。

```
>> s=fotf('s'); G1=s^0.5;
   w=logspace(-5,5); H=bode(G1,w);
   G=ousta_fod(0.5,5,1e-4,1e4);    G1=ousta_fod(0.5,7,1e-4,1e4);
   G2=ousta_fod(0.5,9,1e-4,1e4);  G3=ousta_fod(0.5,11,1e-4,1e4);
   bode(H,G,'--',G1,':',G2,'-.')
```

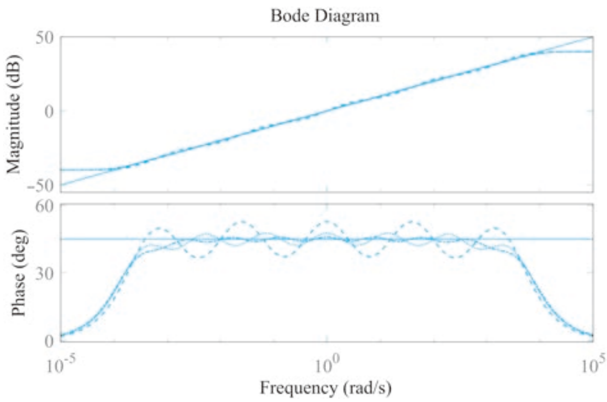


图 5-14　不同阶次下的滤波器

为了更好地演示 11 阶 Oustaloup 滤波器，给出如图 5-15 所示的拟合效果，对幅频特性曲线局部放大，可以看出，这时的频域响应曲线几乎处处都是直线，而不是折线。

说明 5-3　滤波器设计参数选择的建议

（1）如果时间 t 较大时滤波近似效果不佳，则应该相应地减小低频下限 $\omega_{\rm b}$，而初始时刻近似不佳时则应增大频率上限 $\omega_{\rm h}$；

（2）不要害怕使用很高的滤波器阶次和很宽的频段，例如 $(10^{-6}, 10^{6})$，$N = 30$ 并不会显著地增加运算时间。

5.3.2　一种改进的 Oustaloup 滤波器

从前面的例子可以看出，Oustaloup 滤波器的近似效果在选择的频率段边界 $\omega_{\rm h}$，$\omega_{\rm b}$ 处不太理想。另外，滤波器分子的阶次与分母的阶次相同，传递函数不是严格正则系统，所以可以考虑使用改进的滤波器模型。

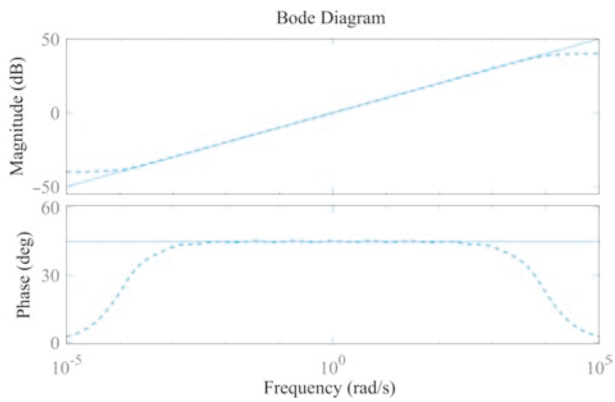


图 5-15　11 阶 Oustaloup 滤波器的频率响应近似效果

定义 5-5 ▶ 改进的 Oustaloup 滤波器

改进的 Oustaloup 滤波器的数学模型[7]为

$$s^{\gamma} \approx \left(\frac{d}{b}\right)^{\gamma} \left(\frac{ds^2 + b\omega_{\rm h}s}{d(1-\gamma)s^2 + b\omega_{\rm h}s + d\gamma}\right) K \prod_{k=1}^{N} \frac{s+\omega'_k}{s+\omega_k} \tag{5-3-4}$$

式中，

$$\omega'_k = \omega_{\rm b}\omega_{\rm u}^{(2k-1-\gamma)/N},\ \ \omega_k = \omega_{\rm u}^{2\gamma/N}\omega'_k,\ \ K = \omega_{\rm h}^{\gamma},\ \ \omega_{\rm u} = \sqrt{\omega_{\rm h}/\omega_{\rm b}} \tag{5-3-5}$$

这样，这一滤波器的阶次 γ 必须满足 $\gamma \in (0,1)$。正常情况下，加权参数可以选择为 $b = 10$，$d = 9$。对算法 5-3 稍加变化就可以设计出改进的 Oustaloup 滤波器。下面的 MATLAB 函数可以直接用于改进 Oustaloup 滤波器的设计：

```
function G=new_fod(r,N,wb,wh,b,d)
  arguments
     r(1,1) double, N(1,1){mustBePositiveInteger}=9
     wb(1,1)double=1e-4; wh(1,1){mustBeGreaterThan(wh,wb)}=1e4;
     b(1,1) double=10; d(1,1) double=9;
  end
  G=(d/b)^r*tf([d,b*wh,0],[d*(1-r),b*wh,d*r]);
  G=G*ousta_fod(r,N,wb,wh);
end
```

该函数的调用格式为 G=new_fod(γ,N,$\omega_{\rm b}$,$\omega_{\rm h}$,b,d)，其中，参数 b，d 可以省略，直接使用默认值。

例 5-8　考虑例 5-5 中的问题，选择 $\omega_{\rm b} = 0.01\,{\rm rad/s}$，$\omega_{\rm h} = 1000\,{\rm rad/s}$，试观测新滤波器的近似结果，并比较分数阶导数的近似效果。

解　由下面的语句可以设计这两个滤波器，并得出 Bode 图的近似效果，如图 5-16 所示。可见改进算法的拟合带宽更宽。

```
>> G1=ousta_fod(0.5,5,0.01,1000);
   G2=new_fod(0.5,5,0.01,1000); zpk(G2)
   s=fotf('s'); G0=s^0.5; w=logspace(-5,5,100);
   H=bode(G0,w); bode(G1,'-',G2,'--',H,':')
```

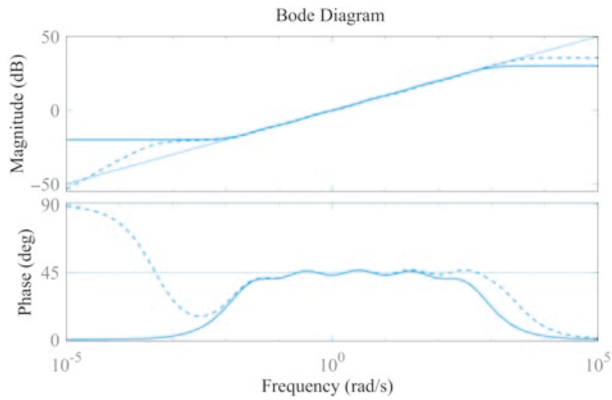


图 5-16　Bode 图比较

得出滤波器 $G_2(s)$ 的零极点表达式为

$$G_2(s)=\frac{60s(s+0.01778)(s+0.1778)(s+1.778)(s+17.78)(s+177.8)(s+1111)}{(s+0.056)(s+0.56)(s+5.623)(s+56.23)(s+562.3)(s+2222)(s+0.00045)}$$

计算得出的分数阶导数曲线如图 5-17 所示。可以看出，改进的滤波器得出的拟合结果稍有改善。

```
>> t=0:0.001:pi; y=exp(-t).*sin(3*t+1);
   y1=lsim(G1,y,t); y2=lsim(G2,y,t);
   y0=glfdiff9(y,t,0.5,3);
   plot(t,y1,t,y2,'--',t,y0,':'), axis([0,pi,-2 8])
```

例 5-9　考虑下面的分数阶传递函数模型：

$$G(s)=\frac{s+1}{10s^{3.2}+185s^{2.5}+288s^{0.7}+1}$$

试比较两个滤波器对分数阶传递函数模型的拟合效果。

解　如果设计 0.2 阶导数的滤波器，则得出的两个滤波器的 Bode 图如图 5-18 所示。可以由 FOTF 工具箱中提供的 `bode()` 重载函数直接绘制分数阶传递函数的 Bode 图。还可以由下面的代码绘制出两个滤波器的 Bode 图，如图 5-19 所示。可以看出，改进滤波器得出的拟合结果明显好于 Oustaloup 滤波器。

```
>> b=[1 1]; a=[10,185,288,1]; nb=[1 0]; na=[3.2,2.5,0.7,0];
   G0=fotf(a,na,b,nb); w=logspace(-4,4,200); H=bode(G0,w);
```

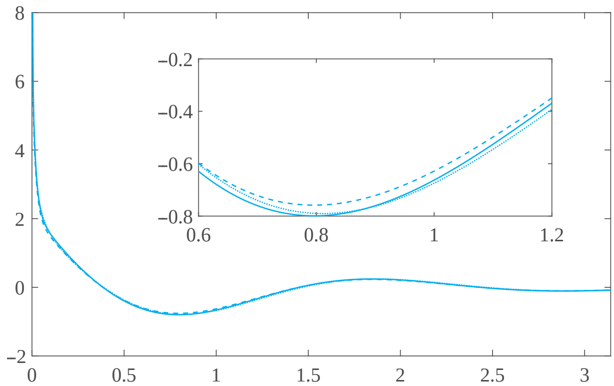


图 5-17　不同滤波器下的分数阶导数计算

```
s=zpk('s'); N=5; w1=1e-3; w2=1e3; b=10; d=9;
g1=ousta_fod(0.2,N,w1,w2); a1=g1;
g2=ousta_fod(0.5,N,w1,w2); g3=ousta_fod(0.7,N,w1,w2);
G1=(s+1)/(10*s^3*g1+185*s^2*g2+288*g3+1);
g1=new_fod(0.2,N,w1,w2); g2=new_fod(0.5,N,w1,w2);
g3=new_fod(0.7,N,w1,w2); bode(g1,a1,'--');
figure, G2=(s+1)/(10*s^3*g1+185*s^2*g2+288*g3+1);
bode(H,G1,'--',G2,':',w)
```

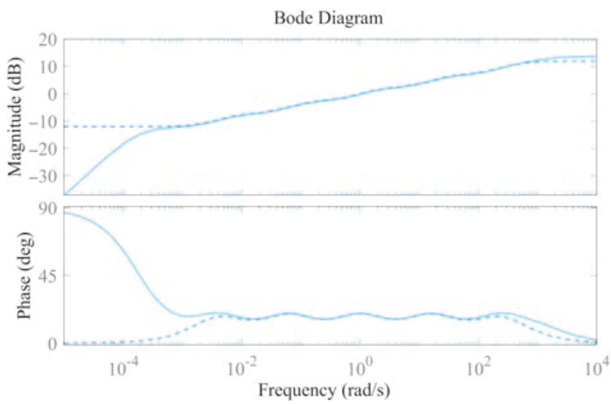


图 5-18　$s^{0.2}$ 的滤波器拟合

5.4　分数阶传递函数的整数阶近似

5.4.1　分数阶传递函数的高阶近似

如果遇到例 5-9 中给出的分数阶传递函数模型，可以用该例中给出的方法得出高阶整数阶近似模型。但是，如果每次都这样用底层命令建立高阶模型是很烦琐的，所以可以编写一个 MATLAB 函数进行这样的自动转换，得出高阶整数阶传递函数

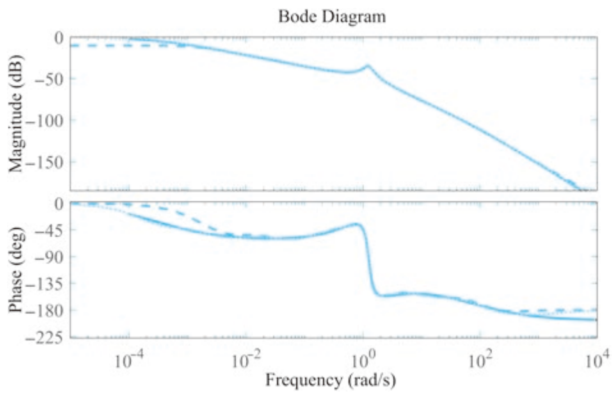


图 5-19　分数阶传递函数的 Bode 图比较

模型。

算法 5-4 ► 分数阶传递函数的整数阶近似

（1）编写子函数处理伪多项式，将每个 s^γ 因子分解成 $s^m s^\alpha$，其中 $m=\lfloor\gamma\rfloor$，$\alpha=\gamma-m$，将 s^α 替换成滤波器模型。

（2）分别处理分子和分母，从伪多项式得出高阶传递函数表示。

（3）对得出的高阶整数阶传递函数求最小实现，得出最简的传递函数模型。

基于上述的算法编写出 MATLAB 函数进行所期望的转换，其中用到了很多 FOTF 工具箱的函数，该函数应该置于 `@fotf` 文件夹，其清单如下：

```
function Ga=high_order(G0,filter,wb,wh,N)
   arguments
      G0, filter='ousta_fod', wb(1,1) double=1e-3
      wh(1,1) double {mustBeGreaterThan(wh,wb)}=1e3
      N(1,1) {mustBePositiveInteger}=5
   end
   [n,m]=size(G0); F=filter;
   for i=1:n, for j=1:m
      if G0(i,j)==fotf(0), Ga(i,j)=tf(0);
      else, G=simplify(G0(i,j)); [a na b nb]=fotfdata(G);
        G1=pseudo_poly(b,nb,F,wb,wh,N)/pseudo_poly(a,na,F,wb,wh,N);
        Ga(i,j)=minreal(G1);
end, end, end, end
%伪多项式的近似
function G1=pseudo_poly(a,na,filter,wb,wh,N)
   G1=0; s=tf('s');
   for i=1:length(a), na0=na(i); n1=floor(na0);
```

```
        if na0>n1, g1=eval([filter '(na0-n1,N,wb,wh)']);
        else, g1=1; end
        G1=G1+a(i)*s^n1*g1;
    end, end
```

函数的调用格式为 $\boldsymbol{G}_{\rm a}$=high_order($\boldsymbol{G}_0$,filter,$\omega_{\rm b}$,$\omega_{\rm h}$,N)，其中，$\boldsymbol{G}_0$ 为分数阶传递函数矩阵对象（FOTF 对象），可以直接处理多变量的问题；输入变量 filter 可以为'ousta_fod'，'new_fod' 或'matsuda_fod'；$\omega_{\rm b}$ 与 $\omega_{\rm h}$ 用于描述感兴趣频率段；N 为每个分数阶算子的近似阶次。可以看出，该函数的结构是开放的，如果有其他滤波器设计函数，也可以仿照这样的结构直接嵌入。

例 5-10 考虑例 5-9 中给出的 FOTF 对象，试得出整数阶的传递函数近似模型。

解 可以用下面的语句直接输入 FOTF 对象模型，并求出用 Oustaloup 滤波器或改进 Oustaloup 滤波器替代分数阶算子后的高阶传递函数近似模型，它们的频域响应拟合结果与图 5-19 中给出的完全一致。

```
>> b=[1 1]; a=[10,185,288,1]; nb=[1 0]; na=[3.2,2.5,0.7,0];
   G0=fotf(a,na,b,nb); N=4; w1=1e-3; w2=1e3; b=10; d=9;
   G10=high_order(G0,'ousta_fod',w1,w2,N)
   G20=high_order(G0,'new_fod',w1,w2,N)
   w=logspace(-3,3); bode(G0), hold on; bode(G10,'--',G20,':')
```

这样得出的近似模型分别为

$$G_{10}(s)=\frac{\begin{gathered}0.025119(s+595.7)(s+421.7)(s+251.2)(s+18.84)\times\\(s+13.34)(s+7.943)(s+1)(s+0.5957)(s+0.4217)\times\\(s+0.2512)(s+0.01884)(s+0.01334)\end{gathered}}{\begin{gathered}(s+595.7)(s+507)(s+154.7)(s+40.89)(s+18.78)(s+7.377)\times\\(s+2.041)(s+0.4418)(s+0.2511)(s+0.05445)(s+0.01334)\times\\(s+0.002349)(s^2+0.3824s+1.613)\end{gathered}}$$

$$G_{20}(s)=\frac{\begin{gathered}0.020523(s+1)(s+0.5957)(s+0.4217)(s+0.2512)(s+7.943)\times\\(s+13.34)(s+18.84)(s+251.2)(s+421.7)(s+595.7)(s+1389)\times\\(s+2222)(s+3704)(s+0.01884)(s+0.01334)(s+0.007943)\times\\(s+0.00063)(s+0.00045)(s+0.00018)\end{gathered}}{\begin{gathered}(s+3704)(s+2325)(s+1111)(s+595.7)(s+488.2)(s+152.6)\times\\(s+40.07)(s+18.78)(s+7.358)(s+2.041)(s+0.4422)(s+0.2511)\times\\(s+0.05453)(s+0.01334)(s+0.007943)(s+0.002197)(s+0.00045)\times\\(s+0.0002201)(s+0.00018)(s^2+0.3762s+1.574)\end{gathered}}$$

这两个滤波器模型得出的阶跃响应曲线可以由下面的语句得出，如图 5-20 所示。可以看出，两个滤波器的近似效果都不理想，原因还是低频段拟合不好。

```
>> t=0:0.01:20; y=step(G0,t); y1=step(G10,t);
   plot(t,y,t,y1,'--')
```

对这样的问题仍然需要降低 $\omega_{\rm b}$ 的值，比如令其为 10^{-4} rad/s，并适当提高滤波器的阶次，这样可以得出如图 5-21 所示的阶跃响应曲线。可以看出，即使这里采用了更大

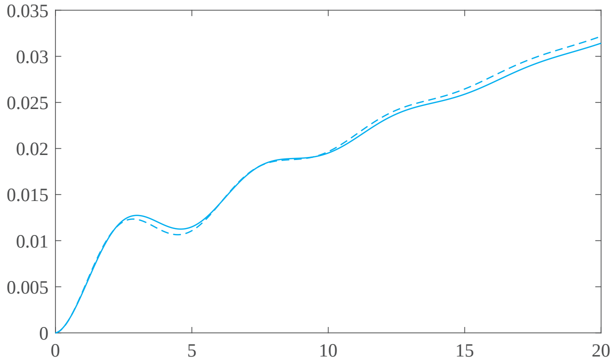


图 5-20　阶跃响应比较

的时间响应区域，计算精度仍然是令人满意的。

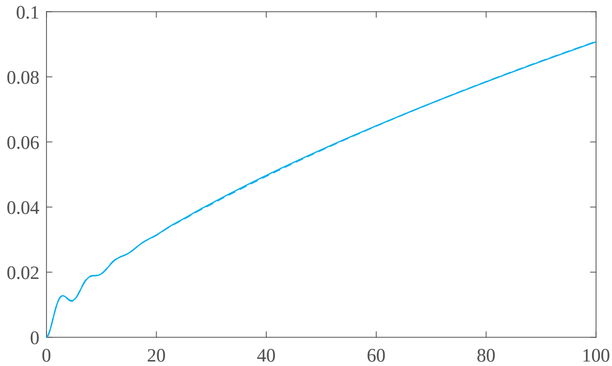


图 5-21　更长时间段的阶跃响应曲线比较

```
>> N=7; w1=1e-4; w2=1e3; G10=high_order(G0,'ousta_fod',w1,w2,N)
   t=0:0.01:100; y=step(G0,t); y1=step(G10,t);
   plot(t,y,t,y1,'--')
```

5.4.2　基于模型降阶技术的低阶近似方法

顾名思义，模型降阶就是用一个低阶模型逼近高阶模型的行为。由于使用滤波器逼近分数阶算子，经常会得到阶次特别高的整数阶模型，所以，这里主要研究用一个低阶整数阶模型逼近分数阶系统的行为。

定义 5-6 ▶ 降阶模型

整数阶降阶模型的一般形式为

$$G_{r/m,\tau}(s)=\frac{\beta_1 s^r+\beta_2 s^{r-1}+\cdots+\beta_{r+1}}{s^m+\alpha_1 s^{m-1}+\cdots+\alpha_{m-1}s+\alpha_m}\mathrm{e}^{-\tau s} \tag{5-4-1}$$

若想得到一个较好的降阶模型，则应该定义一个最优准则。比如，如果原始模

型和降阶模型用同样的信号激励，则由两个模型的输出之差可以定义误差信号，由这个误差信号再定义目标函数。根据这样的目标函数将要求解的问题转换成数值最优化问题。

定义 5-7 ▶ 模型降阶最优指标

模型的最优降阶问题可以由下面的数值最优化问题描述：

$$J = \min_{\boldsymbol{\theta}} \left\| \widehat{G}(s) - G_{r/m,\tau}(s) \right\|_2 \tag{5-4-2}$$

式中，$\boldsymbol{\theta}$ 为降阶模型参数构成的决策向量，即

$$\boldsymbol{\theta} = \left[\beta_1, \beta_2, \cdots, \beta_r, \beta_{r+1}, \alpha_1, \alpha_2, \cdots, \alpha_m, \tau \right]^{\mathrm{T}} \tag{5-4-3}$$

由于式（5-4-2）存在延迟项，可以考虑用 Padé 近似逼近延迟项。这样目标函数可以变成下面的范数求解形式，而最优化问题变换成

$$J = \min_{\boldsymbol{\theta}} \left\| \widehat{G}(s) - \widehat{G}_{r/m}(s) \right\|_2 \tag{5-4-4}$$

上述问题是没有解析解的，只能用数值最优化技术求解原始问题。由参考文献 [8] 给出的思路，可以引出模型最优降阶算法。

算法 5-5 ▶ 模型最优降阶算法

（1）输入高阶整数阶模型 $\widehat{G}(s)$。

（2）选择合适的降阶模型分子与分母阶次 r, m。

（3）用 MATLAB 编写出式（5-4-2）中描述最优化目标函数的程序。

（4）用最优化求解的 `fminsearch()` 函数直接得出最优降阶模型 $G_{r/m,\tau}(s)$。

可以看出，从原来分数阶传递函数模型得到的整数阶近似模型阶次非常高，所以可以用最优降阶的方法得出效果满意的低阶模型[8,9]。最优模型降阶的 MATLAB 函数可以写成

```
function Gr=opt_app(G,r,k,key,G0)
   arguments
      G, r(1,1) {mustBePositiveInteger}
      k(1,1) {mustBePositiveInteger}
      key(1,1){mustBeMember(key,[0,1])}=0, G0=0
   end
   GS=tf(G); Td=totaldelay(GS);
   GS.ioDelay=0; s=tf('s'); GS.InputDelay=0; GS.OutputDelay=0;
   if nargin<5, G0=(s+1)^r/(s+1)^k; end
```

```
    beta=G0.num{1}(k+1-r:k+1); alph=G0.den{1};
    Tau=1.5*Td; x=[beta(1:r),alph(2:k+1)];
    if abs(Tau)<1e-5, Tau=0.5; end
    dc=dcgain(GS); if key==1, x=[x,Tau]; end
    y=opt_fun(x,GS,key,r,k,dc);
    x=fminsearch(@opt_fun,x,[],GS,key,r,k,dc);
    alph=[1,x(r+1:r+k)]; beta=x(1:r+1); beta(r+1)=alph(end)*dc;
    if key==0, Td=0; end
    if key==1, Tau=x(end)+Td; else, Tau=0; end
    Gr=tf(beta,alph,'ioDelay',Tau);
end
%目标函数计算
function y=opt_fun(x,G,key,r,k,dc)
    ff0=1e10; a=[1,x(r+1:r+k)]; b=x(1:r+1); b(end)=a(end)*dc;
    if key==1, tau=x(end);
       if tau<=0, tau=eps; end, [n,d]=pade(tau,3); gP=tf(n,d);
    else, gP=1; end
    Ge=G-tf(b,a)*gP;
    Ge.num{1}=[0,Ge.num{1}(1:end-1)]; [y,ierr]=geth2(Ge);
    if ierr==1, y=10*ff0; else, ff0=y; end
end
%计算H2范数
function [v,iE]=geth2(G)
    G=tf(G); num=G.num{1}; den=G.den{1}; iE=0; v=0; n=length(den);
    if abs(num(1))>eps
       disp('System not strictly proper'); iE=1; return
    else, a1=den; b1=num(2:length(num)); end
    for k=1:n-1
       if (a1(k+1)<=eps), ierr=1; return
       else
          aa=a1(k)/a1(k+1); bb=b1(k)/a1(k+1); v=v+bb*bb/aa; k1=k+2;
          for i=k1:2:n-1,
             a1(i)=a1(i)-aa*a1(i+1); b1(i)=b1(i)-bb*a1(i+1);
    end, end, end, v=sqrt(0.5*v);
end
```

该函数的调用格式为 G_r=opt_app(G,r,m,key,G_0)，其中 r 和 m 为用户指定的降阶模型分子与分母多项式的阶次；key 表示在降阶模型中是否需要延迟项；如果有必要，用户也可以提供自己的初始模型 G_0。

例 5-11　考虑如下给出的分数阶传递函数模型

$$G(s)=\frac{-2s^{0.63}-4}{2s^{3.501}+3.8s^{2.42}+2.6s^{1.798}+2.5s^{1.31}+1.5}$$

试找出其最优降阶模型并评价降阶模型与原模型的匹配程度。

解　可以用下面的语句直接得出高阶近似模型：

```
>> b=[-2 -4]; nb=[0.63 0]; a=[2 3.8 2.6 2.5 1.5];
   na=[3.501 2.42 1.798 1.31 0]; G=fotf(a,na,b,nb);
   G1=high_order(G,'ousta_fod',1e-3,1e3,5); order(G1)
```

可以看出，这样得出的模型为 45 阶的整数阶模型。现在考虑找到一个降阶模型，例如选择 $r=4, m=5$，则可以直接得出降阶模型，并得出高阶与降阶模型的 Bode 图，如图 5-22 所示。这里的相位图看似有很大差异，不过相位图是平行的相差 360°，所以，可以认为它们是一致的。

```
>> G2=opt_app(G1,4,5,0), zpk(G2)
   w=logspace(-4,4); H=bode(G,w); bode(H,G1,'--',G2,':')
```

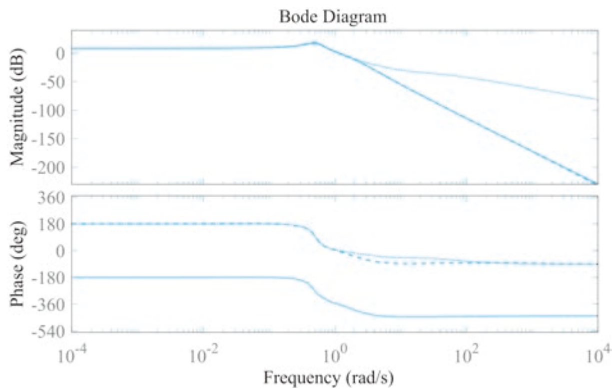


图 5-22　Bode 图比较

这样得出的低阶近似模型为

$$G_2(s)=\frac{-4.7911(s+0.2452)(s+0.01657)(s^2-8.983s+31.65)}{(s+140.9)(s+0.3799)(s+0.0173)(s^2+0.2303s+0.2479)}$$

可以看出，这样得出的降阶模型的频域响应还是很接近原始模型的，而高频部分的拟合看起来和原始模型有很大的误差。不过不必太担心，因为这里的 Bode 幅频特性使用的单位是 dB（分贝），当增益小于 -20 dB 时，其实际的倍数小于 -20 dB，即 10^{-1}，而 -50 dB 大约为 $10^{-2.5}$，故二者都是很小的增益，并没有太显著的差异，至少没有图 5-22 中看起来那么大的差异。相频特性拟合中，几条曲线基本上相差的是 360° 和 720°，这里给出的是手工修改后的相频曲线。可见，实际分数阶系统的频域响应曲线与高阶整数阶拟合模型几乎完全一致。

还可以求出各个模型的阶跃响应，如图 5-23 所示。可见，降阶模型和原始模型还是

很接近的。

```
>> t=0:0.01:10; y=step(G,t); y1=step(G1,t); y2=step(G2,t);
   plot(t,y,t,y1,'--',t,y2,':')
```

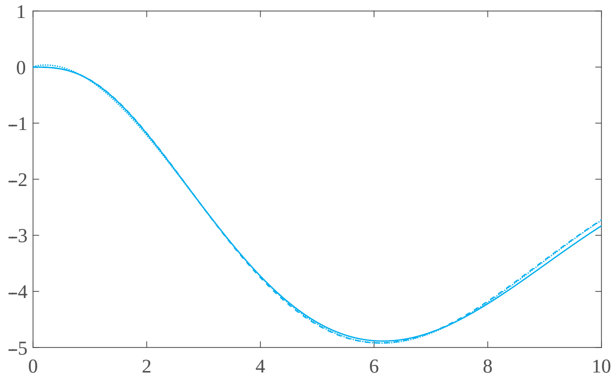


图 5-23　阶跃响应比较

例 5-12　考虑下面的分数阶传递函数模型,试得出降阶模型并比较其效果。

$$G(s)=\frac{5}{s^{2.3}+1.3s^{0.9}+1.25}$$

解　首先先得出原始模型的高阶整数阶近似模型,然后采用不同的降阶方法,可以得出如图 5-24 所示的阶跃响应比较。这里的阶跃响应曲线是 step() 函数自动绘制的。

```
>> w1=1e-3; w2=1e3; N=9;
   s=fotf('s'); G=5/(s^2.3+1.3*s^0.9+1.25);
   G0=high_order(G,'ousta_fod',w1,w2,N); zpk(G0)
   G1=opt_app(G0,1,2,0), G2=opt_app(G0,2,3,0)
   G3=opt_app(G0,3,4,0), step(G0,G1,'--',G2,':',G3,'-.')
```

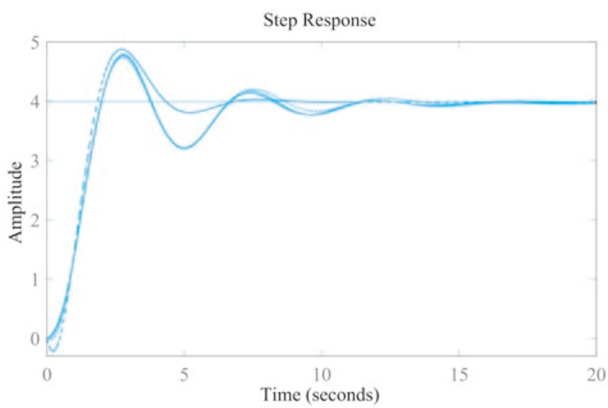


图 5-24　分数阶传递函数的阶跃响应比较

上面的语句首先得出一个 19 阶的高阶近似模型(其数学形式从略),还可以得出最

优降阶模型为

$$G_1(s)=\frac{-2.045s+7.654}{s^2+1.159s+1.917},\quad G_2(s)=\frac{-0.5414s^2+4.061s+2.945}{s^3+0.9677s^2+1.989s+0.7378}$$

$$G_3(s)=\frac{-0.2592s^3+3.365s^2+4.9s+0.3911}{s^4+1.264s^3+2.25s^2+1.379s+0.09797}$$

可以看出，二阶模型 $G_1(s)$ 与 3 阶模型 $G_2(s)$ 的拟合效果不甚理想，4 阶与 5 阶近似模型的响应很接近分数阶模型。

在一些特定的例子中，比如在例 5-9 的问题中，如果降阶模型的阶次选择得过低，则逼近效果会很差，所以得到降阶模型后应该检验一下才能真正使用。

5.5 无理分数阶模型的近似

在分数阶控制系统中，有的时候系统的某个组成部分不能很好地用有理分数阶传递函数直接描述。例如，若传递函数存在 $\left(as+b)/(cs+d)\right.^{\alpha}$ 这样的项，而 α 不是整数，则该项只能展开成无穷项的级数。如果存在 $\left.as^{\beta}+b\right.^{\alpha}$ 形式的项则更难以处理。本节将介绍这类无理传递函数的近似方法。

5.5.1 隐式无理模型的近似

考虑一种简单的无理传递函数

$$G(s)=\frac{1}{(\tau s+1)^{\nu}}=\tau^{-\nu}\frac{1}{(s+1/\tau)^{\nu}} \tag{5-5-1}$$

定理 5-2 ▶ 无理模型的时域表示

如果系统的传递函数由式 (5-5-1) 给出，则其输入信号 $u(t)$ 与输出信号 $y(t)$ 之间的关系满足[10]

$$y(t)=\tau^{-\nu}\mathrm{e}^{-t/\tau}\mathscr{D}^{-\nu}\left[\mathrm{e}^{t/\tau}u(t)\right] \tag{5-5-2}$$

证明 由传递函数模型可得 $Y(s)=G(s)U(s)=G(s+1/\tau)U(s)$，则 $Y(s-1/\tau)=G(s-1/\tau)U(s-1/\tau)=\tau^{-\nu}U(s-1/\tau)/s^{\nu}$。取 Laplace 反变换，则利用 Laplace 变换的平移性质 $\mathscr{L}[\mathrm{e}^{-at}f(t)]=F(s+a)$，取 $a=-1/\tau$，得出

$$\mathscr{L}[Y(s-1/\tau)]=\tau^{-\nu}\mathscr{D}^{-\nu}\left[\mathrm{e}^{t/\tau}u(t)\right] \tag{5-5-3}$$

再利用平移性质，则得出下式，定理得证。

$$\mathrm{e}^{t/\tau}y(t)=\tau^{-\nu}\mathscr{D}^{-\nu}\left[\mathrm{e}^{t/\tau}u(t)\right] \tag{5-5-4}$$

定理 5-3 ▶ 隐式无理微分方程

选择状态变量 $x(t)=\mathscr{D}^{-\nu}\left[\mathrm{e}^{t/\tau}u(t)\right]$，则无理系统（5-5-1）对应的分数阶状态方程模型如下[10]，该模型又称为隐式分数阶模型。

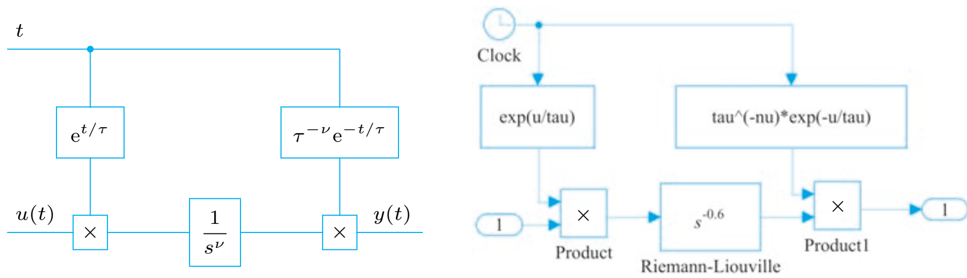
$$\begin{aligned}\mathscr{D}^{\nu}x(t)&=\mathrm{e}^{t/\tau}u(t)\\ y(t)&=\tau^{-\nu}\mathrm{e}^{-t/\tau}x(t)\end{aligned}\tag{5-5-5}$$

说明 5-4　隐式无理微分方程

（1）这里给出了一种近似方法，但该方法不能得出传递函数近似模型，无法比较频域响应拟合效果。

（2）利用这里介绍的方法，可以用如图 5-25（a）所示的框图描述近似方法。基于这里给出的框图，可以直接建立 Simulink 仿真模型，如图 5-25（b）所示。其中，$1/s^{\nu}$ 模块可以由 Oustaloup 滤波器实现。

（3）由于 Simulink 模型中存在 $t^{-\nu}$ 项，所以若 $\nu>0$，仿真过程中不宜将 t 的初值设置为 0，可以将其设置成微小的值，如 0.0001。



（a）框图表示　　（b）Simulink 实现（`c5mimp.slx`）

图 5-25　隐式无理模型建模

5.5.2　频域响应近似方法

当然，并不是所有的无理分数阶模型都能用式（5-5-1）的简单形式表示，所以需要探讨一般无理传递函数模型的近似方法。

如果测得或计算出某个模型的一些频率响应点的数据，则可以由 MATLAB 提供的 `invfreqs()` 函数获得整数阶近似模型。一般情况下，这样的模型可以很好地逼近给出的频域响应数据。该函数的调用格式为 G=`invfreqs`(H,$\boldsymbol{w}$,m,n)，其中，H 为频域响应数据，$\boldsymbol{w}$ 为频率点向量，m 和 n 分别为预期的分子与分母的阶次。

算法 5-6 ▶ 无理传递函数模型的有理拟合与近似

（1）由无理传递函数模型产生一组频域响应数据。

（2）选择合适的拟合模型分子与分母的阶次。

（3）调用 invfreqs() 函数直接获得频域响应数据的拟合模型。

（4）验证得到的模型，若拟合效果不满意，则增加预期模型的阶次然后转到（2）重新拟合，或重新生成一组频域响应数据转到（1），直到得到满意的模型。

例 5-13 考虑文献 [11] 给出的分数阶定量反馈理论（quantitative feedback theory，QFT）控制器模型

$$G_c(s) = 1.8393\left(\frac{s+0.011}{s}\right)^{0.96}\left(\frac{8.8\times10^{-5}s+1}{8.096\times10^{-5}s+1}\right)^{1.76}\frac{1}{(1+s/0.29)^2}$$

试找出较好的有理近似模型。

解 这样的控制器结构很复杂，只能借助于比较好的有理传递函数模型逼近。由于 MATLAB 控制系统工具箱中的 frd() 函数只能处理整数阶模型，该函数返回一个结构体变量，可以对其成员变量 ResponseData 进行处理，获得其非整数次幂，最终得出无理传递函数的精确频域响应数据。有了频域响应数据，就可以调用 invfreqs() 函数获得拟合模型。对本例而言，可以先选择感兴趣的频域段 $\omega\in\left(10^{-4},10^{2}\right)$ rad/s，再输入下面的 MATLAB 命令：

```
>> w=logspace(-4,2); G1=tf([1 0.011],[1 0]); F1=frd(G1,w);
   G2=tf([8.8e-5 1],[8.096e-5 1]); F2=frd(G2,w);
   s=tf('s'); G3=1/(1+s/0.29)^2; F3=frd(G3,w); F=F1;
   h1=F1.ResponseData; h2=F2.ResponseData; h3=F3.ResponseData;
   h=1.8393*h1.^0.96.*h2.^1.76.*h3; F.ResponseData=h;
   [n,d]=invfreqs(h(:),w,4,4); H1=zpk(tf(n,d))
```

这样得出的近似控制器为

$$H_1(s)=\frac{-1.5\times10^{-10}(s-3.913\times10^4)(s+2.635\times10^4)(s+0.01071)(s+0.0006019)}{(s+0.2922)(s+0.2878)(s+0.0007664)(s+1.284\times10^{-7})}$$

从结果可以看出，有的稳定零极点对相距很近，所以可以通过最小实现技术将它们对消，得到更低阶次的模型：

```
>> H1a=minreal(H1,1e-3), H1b=minreal(H1,1e-2)
```

这样得出的更低阶近似为

$$H_{1a}(s)=\frac{-1.5002\times10^{-10}(s-3.913\times10^4)(s+2.635\times10^4)(s+0.01071)}{(s+0.2922)(s+0.2878)(s+1.284\times10^{-7})}$$

$$H_{1b}(s)=\frac{-1.5002\times10^{-10}(s-3.913\times10^4)(s+2.635\times10^4)}{(s+0.2922)(s+0.2878)}$$

对于无理系统拟合的问题还可以尝试使用 Matsuda–Fujii 滤波器。用下面语句直接设计这样的滤波器，不过该滤波器 $H_2(s)$ 是不稳定的模型，说明 Matsuda–Fujii 设计方法并不能保证滤波器的稳定性，使用时需要慎重。

```
>> w=logspace(-4,2,11); F1=frd(G1,w); F2=frd(G2,w); F3=frd(G3,w);
```

```
h1=F1.ResponseData; h2=F2.ResponseData; h3=F3.ResponseData;
h=1.8393*h1.^0.96.*h2.^1.76.*h3; H2=zpk(matsuda_fod(h,w))
```

选择更大一些的频率段 $10^{-6}, 10^4$ rad/s验证得出的模型，就可以得出原控制器与各个低阶近似模型的Bode图，如图5-26所示。理论值用点线表示，可见各个幅频特性的逼近很理想，相频特性相差360°(这里做了手工平移)，也是比较理想的。

```
>> w=logspace(-6,4,200); F1=frd(G1,w); h1=F1.ResponseData;
   F2=frd(G2,w); h2=F2.ResponseData; F3=frd(G3,w);
   h3=F3.ResponseData; h=1.8393*h1.^0.96.*h2.^1.76.*h3;
   F=F1; F.ResponseData=h; bode(H1a,'--',H1b,F,':',H1,'-.',w)
```

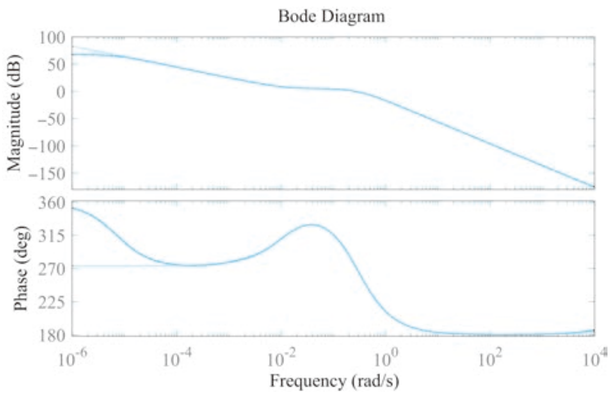


图5-26 频域响应比较

从比较结果看，最小实现模型 $H_{1a}(s)$ 与4阶控制器 $H_1(s)$ 甚至与原无理传递函数模型很接近，而二阶近似模型 $H_{1b}(s)$ 的效果很差，这意味着原来的无理传递函数模型至少需要3阶传递函数近似。上面得出的 $H_{1a}(s)$ 是通过最小实现得出的，其实更应该由 invfreqs() 函数直接得出：

```
>> w=logspace(-4,2); F1=frd(G1,w); F2=frd(G2,w); F3=frd(G3,w);
   h1=F1.ResponseData; h2=F2.ResponseData; h3=F3.ResponseData;
   h=1.8393*h1.^0.96.*h2.^1.76.*h3; [n,d]=invfreqs(h(:),w,3,3);
   H3=zpk(tf(n,d))
```

直接得出的3阶模型为

$$H_3(s) = \frac{-1.4886\times10^{-10}(s-3.93\times10^4)(s+2.644\times10^4)(s+0.01009)}{(s+0.3028)(s+0.2768)(s+1.103\times10^{-5})}$$

例5-14 试用隐式模块逼近例5-13中给出的QFT控制器。

解 对QFT控制器传递函数进行重组，可以写成如下形式：

$$G_c(s)=\frac{1.8393/0.011^{0.96}}{s^{0.95}\,(1+s/0.29)^2}\left(\frac{s}{0.011}+1\right)^{0.96}(8.8\times10^{-5}s+1)^{1.76}(8.096\times10^{-5}s+1)^{-1.76}$$

由这个数学形式可以直接构造出如图5-27所示的Simulink仿真模型，其整数阶部分由传递函数模型给出，其中，分子编辑框num填写 `1.8393/0.011^0.96`，分母编辑框

den 填写 `conv([1/0.29 1],[1/0.29 1])`。

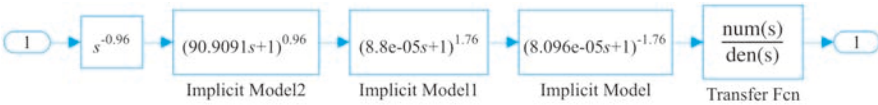


图 5-27　QFT 控制器的 Simulink 模型(`c5mqft.slx`)

5.5.3　Charef近似

Charef 近似技术是对下面无理传递函数模型的一种有效的近似方法[4,12]：

$$H(s)=\frac{1}{(1+s/p_{\mathrm{T}})^{\alpha}} \tag{5-5-6}$$

定义 5-8 ► Charef滤波器

Charef 滤波器技术类似于 Oustaloup 滤波器的设计方法，它采用分段的渐近线在 $(0,\omega_{\mathrm{M}})$ rad/s 频率段内逼近无理传递函数，得出

$$H_1(s)=\frac{(1+s/z_0)(1+s/z_1)\cdots(1+s/z_{n-1})}{(1+s/p_0)(1+s/p_1)\cdots(1+s/p_n)} \tag{5-5-7}$$

式中，先得出以下几个关键量：

$$p_0=p_{\mathrm{T}}10^{\delta/(20\alpha)},\quad a=10^{\delta/[10(1-\alpha)]},\quad b=10^{\delta/(10\alpha)} \tag{5-5-8}$$

然后由下面的式子计算出滤波器的零极点：

$$p_{i+1}=p_0(ab)^{i+1},\quad z_i=ap_i,\quad i=0,1,\cdots,n-1 \tag{5-5-9}$$

零极点对的个数可以由下式直接确定：

$$n=\left\lceil\frac{\ln(\omega_{\mathrm{M}}/p_0)}{\ln(ab)}\right\rceil \tag{5-5-10}$$

式中，变量 δ 是频域响应拟合的误差容限，正常情况下其取值可以为 $\delta=1$ dB。这里的容限是针对渐近线而言的，实际的拟合误差要小于该值。

可以建立起下面的算法设计 Charef 滤波器。

算法 5-7 ► Charef滤波器的设计

(1) 输入需要拟合的无理传递函数模型的参数 p_{T} 与 α。

(2) 选择指标 $\delta,\omega_{\mathrm{M}}$ 设计滤波器。

(3) 由式 (5-5-8) 与 (5-5-10) 直接计算关键参数 a,b,p_0 与 n。

(4) 由式 (5-5-9) 计算滤波器的零极点与增益，构造出 Charef 滤波器模型。

根据上面的算法可以编写出Charef滤波器设计的MATLAB函数：

```
function G=charef_fod(alpha,pT,delta,wM)
   arguments
      alpha(1,1) double, pT(1,1) double
      delta(1,1) double, wM(1,1) double
   end
   p0=pT*10^(delta/20/alpha);
   a=10^(delta/10/(1-alpha)); b=10^(delta/10/alpha);
   n=ceil(log(wM/p0)/log(a*b)); ii=1:n; p=p0*(a*b).^ii;
   p=[p0 p]; z=a*p(1:n); K=prod(p)/prod(z); G=zpk(-z,-p,K);
end
```

该函数的调用格式为 G=charef_fod(α,$p_{\rm T}$,δ,$\omega_{\rm M}$)，该函数的输入变量与对应的数学公式是一致的，返回变量则是设计出来的Charef滤波器。

例 5-15 试设计无理传递函数 $G(s)=1/(1+0.2s)^{0.5}$ 的Charef滤波器。

解 可以选择误差容限 $\delta=1\,{\rm dB}$ 与频率段上限 $\omega_{\rm M}=1000\,{\rm rad/s}$。这样可以由下面的语句直接设计出Charef滤波器：

```
>> G1=charef_fod(0.5,1/0.2,1,1000)
```

设计出来的Charef滤波器为

$$G_1(s)=\frac{99.763(s+9.976)(s+25.06)(s+62.95)(s+158.1)(s+397.2)(s+997.6)}{(s+6.295)(s+15.81)(s+39.72)(s+99.76)(s+250.6)(s+629.5)(s+1581)}$$

此外，由前面介绍的频域响应拟合方法也可以求取原模型的整数阶近似模型：

```
>> w=logspace(-2,3); G0=tf(1,[0.2 1]);
   H=frd(G0,w); h=H.ResponseData; h=h.^0.5;
   [n,d]=invfreqs(h(:),w,5,5); G2=zpk(tf(n,d))
```

得到的近似模型为

$$G_2(s)=\frac{0.01541(s+8899)(s+1121)(s+340.2)(s+91.78)(s+17.16)}{(s+2412)(s+609.2)(s+183.6)(s+41.34)(s+7.41)}$$

事实上，前面介绍的Carlson滤波器也适合于近似无理传递函数模型，可以通过下面的语句直接设计该滤波器：

```
>> s=tf('s'); G=1/(1+0.2*s); H1=carlson_fod(0.5,G,2); zpk(H1)
```

设计出的滤波器模型为

$$H_1(s)=\frac{0.11111(s+165.8)(s+20)(s+8.52)(s+6.667)(s+6.666)(s+5.662)(s+5)^6}{\begin{array}{c}(s+42.74)(s+12.1)(s+6.678)(s+4.887)(s^2+13.32s+44.37)\times\\(s^2+10.33s+26.67)(s^2+9.859s+24.31)(s^2+10.08s+25.42)\end{array}}$$

无理传递函数与Charef滤波器的Bode图比较如图5-28所示。可以看出，Charef滤波器的频域响应也很令人满意。

```
>> w=logspace(-3,5); M=tf(1,[0.2 1]); H=frd(M,w);
   h=H.ResponseData; H.ResponseData=h.^0.5;
   bode(H,G1,'--',G2,':',H1,'-.'), xlim([1e-3 1e5])
```

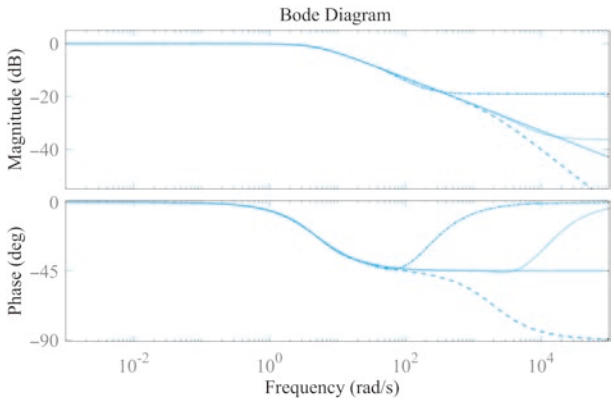


图 5-28　频域响应比较

无理传递函数 $G(s)$ 的阶跃响应可以通过数值Laplace反变换的方法求出，该阶跃响应与Charef滤波器和频率响应拟合模型阶跃响应的比较如图5-29所示，其中，函数 `INVLAP_new()` 是基于数值Laplace变换的求解函数，第6章将详细介绍。

```
>> [t,y]=INVLAP_new('1/(1+0.2*s)^0.5',0,3,1001,0,'1/s');
   y1=step(G1,t); y2=step(G2,t); y3=step(H1,t);
   plot(t,y,t,y1,'--',t,y2,':',t,y3,'-.')
```

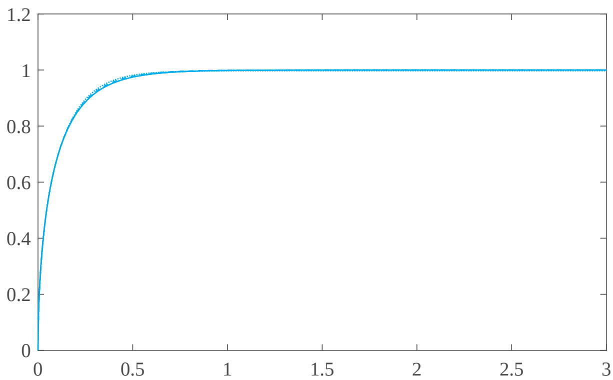


图 5-29　阶跃响应比较

例5-16　考虑下面带有两个无理因子的模型，试得出较好的Charef滤波器：

$$G(s)=\frac{1}{(1+s/1.6)^{0.6}\,(1+s/6.2)^{0.3}}$$

解　原始模型可以认为是两个无理传递函数的乘积，可以得出Charef滤波器。

```
>> G11=charef_fod(0.6,1.6,1,1000);
   G12=charef_fod(0.3,6.2,1,1000); G1=G11*G12; zpk(G1)
```

这样得出的近似模型为

$$G_1(s)=\frac{\begin{array}{c}12018(s+3.447)(s+8.997)(s+12.64)(s+23.48)\times\\(s+37.85)(s+61.3)(s+113.3)(s+160)(s+339.2)\times\\(s+417.6)(s+1015)(s+1090)\end{array}}{\begin{array}{c}(s+1.938)(s+5.06)(s+9.1)(s+13.21)(s+27.24)\times\\(s+34.47)(s+81.55)(s+89.97)(s+234.8)(s+244.1)\times\\(s+613)(s+730.8)(s+1600)(s+2188)\end{array}}$$

选择感兴趣频率段为 $\omega\in\ 10^{-2},10^{3}\ \mathrm{rad/s}$，还可以得出频域响应近似模型：

```
>> G01=tf(1,[1/1.6 1]); G02=tf(1,[1/6.2 1]);
   w=logspace(-2,3); H1=frd(G01,w); h1=H1.ResponseData;
   H2=frd(G02,w); h2=H2.ResponseData; H=H1;
   h=h1.^0.6.*h2.^0.3; [n,d]=invfreqs(h(:),w,5,5);
   H.ResponseData=h; G2=zpk(tf(n,d))
```

这样得出的模型为

$$G_2(s)=\frac{0.00011629(s+4.456\times10^4)(s+1348)(s+358.2)(s+81.78)(s+7.508)}{(s+1579)(s+407.5)(s+97.05)(s+10.91)(s+2.266)}$$

如果用两个频域响应拟合模型串联的形式逼近原始模型，则可以得出第 3 个整数阶传递函数的近似模型：

```
>> h1=h1.^0.6; h2=h2.^0.3; [n,d]=invfreqs(h1(:),w,5,5);
   g=tf(n,d); [n,d]=invfreqs(h2(:),w,5,5); G3=zpk(tf(n,d)*g)
```

这样得出的模型为

$$G_3(s)=\frac{\begin{array}{c}0.00029552(s+1.066\times10^4)(s+6244)(s+1122)(s+1016)\times\\(s+321.9)(s+319.3)(s+87.85)(s+76.86)(s+17.36)(s+9.993)\end{array}}{\begin{array}{c}(s+2975)(s+2079)(s+709.9)(s+526.2)(s+223.3)(s+143.8)\times\\(s+55.69)(s+24.43)(s+10.66)(s+2.536)\end{array}}$$

这 3 个模型与原始无理传递函数模型的 Bode 图比较如图 5-30 所示。可以看出，分段拟合模型 $G_1(s)$ 的效果最差。

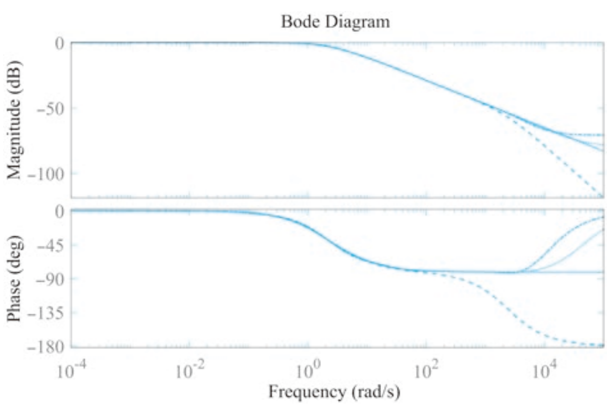


图 5-30　频域响应拟合的比较

```
>> w=logspace(-4,5); H1=frd(G01,w); h1=H1.ResponseData;
```

```
H2=frd(G02,w); h2=H2.ResponseData; H=H1;
h=h1.^0.6.*h2.^0.3; H.ResponseData=h;
bode(H,G1,'--',G2,':',G3,'-.',w)
```

各个模型的阶跃响应曲线如图5-31所示。可以看出，3个近似模型的效果都比较好。当然，Oustaloup滤波器方法效果最好，且有进一步改进的余地。

```
>> sys='1/(1+s/1.6)^0.6/(1+s/6.2)^0.3';
   [t,y]=INVLAP_new(sys,0,3,1001,0,'1/s');
   y1=step(G1,t); y2=step(G2,t); y3=step(G3,t);
   plot(t,y,t,y1,'--',t,y2,':',t,y3,'-.')
```

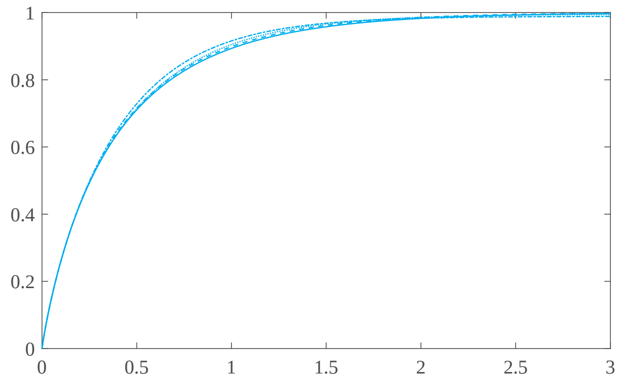


图5-31　阶跃响应的比较

例5-17　重新考虑例5-16中的无理模型，试利用Simulink近似该模型。

解　显然，原模型可以由两个隐式模型串联搭建，由此可以构造如图5-32所示的Simulink模型。在该模型下可以得出如图5-33所示的阶跃响应曲线，从曲线看不出与数值Laplace变换结果的区别。

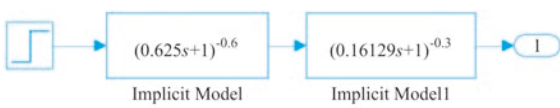


图5-32　Simulink仿真模型（`c5msim1.slx`）

```
>> sys='1/(1+s/1.6)^0.6/(1+s/6.2)^0.3';
   [t,y]=INVLAP_new(sys,0,3,1001,0,'1/s'); N=15;
   ww=[1e-4 1e4]; [t1,~,y1]=sim('c4msim1',3);
   plot(t,y,t1,y1,'--')
```

5.5.4　复杂无理模型的最优Charef滤波器设计

现在回顾一下定义5-8中的Charef滤波器，该滤波器中的比值为$z_i/p_i = a$与$p_{i+1}/z_i = b$，$i = 0, 1, \cdots, n-1$，其中a与b为固定的数值。如果比值不再固定，则它

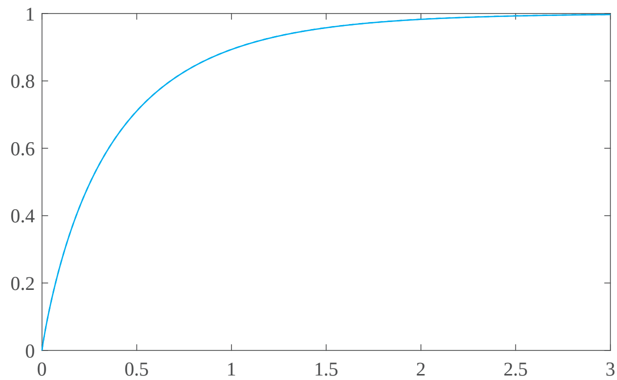


图 5-33　阶跃响应的比较

们可以在某种性能指标下寻优，引入最优的 Charef 滤波器[13] 的概念。

定义 5-9 ▶ Charef 滤波器格式

复杂的无理传递函数模型 $G(s)$ 可以在频率段 $\omega_\mathrm{m}, \omega_\mathrm{M}$ rad/s 内由 Charef 类滤波器拟合，模型的参数可以通过最优化得出。

$$H(s) = K\frac{(1+s/z_0)\cdots(1+s/z_{n-1})}{(1+s/p_0)(1+s/p_1)\cdots(1+s/p_n)} \tag{5-5-11}$$

对感兴趣频率段 $\omega_\mathrm{m}, \omega_\mathrm{M}$ 而言，可以选择一系列零极点对 z_i, p_i 的中心点频率 ω_{u_i}，使得

$$\frac{\omega_{u_{i+1}}}{\omega_{u_i}} = c,\quad \omega_{u_{i+1}} = c\omega_{u_i},\ i = 0, 1, 2, \cdots, n \tag{5-5-12}$$

这样，滤波器的零极点可以直接得出

$$p_i = 10^{y_i\omega_{u_i}}, i = 0, 1, \cdots, n;\ z_i = 10^{\omega_{u_i}/y_i},\ i = 0, 1, \cdots, n-1 \tag{5-5-13}$$

式中，y_i 为可优化的参数。

选择一组频率点 $\omega_1, \omega_2, \cdots, \omega_N$，并计算出无理传递函数的频域响应数据 $\boldsymbol{h}_a$，最优化的目标是得到一组决策变量 $\boldsymbol{x} = \ c, w_{u_0}, k, y_0, y_1, \cdots, y_n$ 的值，使得选定的目标函数最小化。这里最优化的目标函数为

$$\begin{aligned} f(\boldsymbol{x}) = &\ w_1||\Delta\hat{\boldsymbol{h}}(\mathrm{j}\omega_i)||_\infty + \frac{w_2}{N}\sum_{i=1}^{N}\Delta\hat{\boldsymbol{h}}(\mathrm{j}\omega_i) + \\ &\ w_3||\Delta\angle\hat{\boldsymbol{h}}(\mathrm{j}\omega_i)||_\infty + \frac{w_4}{N}\sum_{i=1}^{N}\Delta\angle\hat{\boldsymbol{h}}(\mathrm{j}\omega_i) \end{aligned} \tag{5-5-14}$$

式中，w_i 为加权系数；$\boldsymbol{h}(\mathrm{j}\omega)$ 为滤波器模型的频域响应，且

$$\Delta\hat{\boldsymbol{h}}(\mathrm{j}\omega_i) = \ |\boldsymbol{h}(\mathrm{j}\omega_i)| - |\boldsymbol{h}_a(\mathrm{j}\omega_i)| \tag{5-5-15}$$

$$\Delta\angle\hat{\boldsymbol{h}}(\mathrm{j}\omega_i) = \angle h(\mathrm{j}\omega_i) - \angle h_a(\mathrm{j}\omega_i) \tag{5-5-16}$$

如果选择无穷范数，则表示求向量所有元素绝对值的最大值。

定义 5-10 ▶ 最优化模型

文献 [13] 中给出了 Charef 滤波器设计的有约束最优化模型：

$$\min_{\boldsymbol{x}\ \text{s.t.}\ \begin{cases} c>1 \\ \omega_{u0}>0 \\ 0<y_i<1, i=0,1,\cdots,n \end{cases}} f(\boldsymbol{x}) \tag{5-5-17}$$

算法 5-8 ▶ 最优 Charef 滤波器设计

（1）选择频率向量 $\boldsymbol{\omega}_{\mathrm{r}}$，并计算无理传递函数的频域响应。

（2）基于式（5-5-14）写出最优化问题的目标函数。

（3）生成一个初始搜索点 $\boldsymbol{x}_0$ 并开始非线性最优化问题的求解。

（4）由最优化的结果构造出最优 Charef 滤波器。

基于上述的算法，可以编写出最优 Charef 滤波器设计的 MATLAB 函数：

```
function Ga=charef_opt(wr,n,G,wt,a,wc)
   arguments, wr(:,1), n(1,1){mustBePositiveInteger}
      G(:,1), w(1,4)=[1 1 1 1], a(1,1)=0.5, wc(1,1)=1
   end
   e=1; x0(1)=10^(e/(10*a*(1-a)));
   x0(2)=10^(e/(20*(1-a)))*wc*10^(e/(20*a));
   x0(3:n+3)=ones(1,n+1)/10^(e/(20*(1-a)));
   ff=optimset; ff.Display='iter'; A=[];B=[];Aeq=[];Beq=[];CF=[];
   xm=[1 0 zeros(1,n+1)]; xM=[3 10 ones(1,n+1)];
   x1=fmincon(@charef_obj,x0,A,B,Aeq,Beq,xm,xM,CF,ff,wr,G,wt);
   c=x1(1); wu0=x1(2); wu=wu0*c.^[0:n]; p=wu.*x1(3:end);
   z=wu.^2./p; k=prod(p)/prod(z); Ga=zpk(-z',-p',k);
end
%计算目标函数值的子函数
function f=charef_obj(x,wr,G,wt)
   n=length(x)-2;
   c=x(1); wu0=x(2); wu=wu0*c.^(0:(n-1)); p=wu.*x(3:end);
   z=wu.^2./p; k=prod(p)/prod(z); Ga=zpk(-z',-p',k);
   Ga1=frd(Ga,wr); h=Ga1.ResponseData; Ga_fr=h(:);
   f=wt(1)*norm(abs(angle(Ga_fr)-angle(G)),inf)+...
      wt(2)*sum(abs(angle(Ga_fr)-angle(G)))/length(wr)+...
```

```
        wt(3)*norm(abs(abs(Ga_fr)-abs(G)),inf)+...
        wt(4)*sum(abs(abs(Ga_fr)-abs(G)))/length(wr);
end
```

其中，这样的最优化目标函数可以用 MATLAB 子函数描述出来，$\boldsymbol{x}$ 为决策变量向量。除了 $\boldsymbol{x}$ 之外，其他变量都是附加变量，都应该以附加变量的形式传递给求解函数 `fmincon()`。这里的附加变量包括：a 为阶次的初值；ω_c 为一阶系统时间常数 ω_T 的初值；$\boldsymbol{\omega}_\mathrm{r}$ 为频率点构成的向量；$\boldsymbol{G}$ 为频率点 ω_r 处原无理传递函数的频域响应数据；$\boldsymbol{w}_\mathrm{t}$ 为加权系数 w_1, w_2, w_3 与 w_4 构成的向量，这些加权值的默认值都是 1。可以看出，在实际寻优过程中 ω_c 和 a 的值并不是很重要，它们就是数值最优化的初值，使用默认值就可以了。该函数调用了 MATLAB 函数 `fmincon()` 求最优化问题的数值解。

该函数调用格式为 G_a=`charef_opt(`$\boldsymbol{\omega}_\mathrm{r}$,$n$,$\boldsymbol{G}$,$\boldsymbol{w}_\mathrm{t}$,$a$,$\omega_\mathrm{c}$`)`，其中，$\boldsymbol{\omega}_\mathrm{r}$ 与 $\boldsymbol{G}$ 提供无理传递函数的频域响应信息，n 为期待的最优 Charef 滤波器的阶次。

说明 5-5　最优 Charef 滤波器

（1）感兴趣频率段由频率向量 $\boldsymbol{\omega}_\mathrm{r}$ 给出，本算法兼顾幅值与相位数据的拟合。

（2）频率 ω_m 的选择对最终结果并没有太大影响，实际的频率下界由受控对象模型的转折点确定。

（3）原始模型的静态增益必须为 1，否则应该事先提取出来。

（4）在实际应用中，最好将 c 的上限设置为 3，这样零极点对 (z_i, p_i) 可能不会交叉重叠。

（5）选择阶次 n 的建议：首先选择一个初始整数 n_0，得出最优 Charef 滤波器并显示拟合误差，再选择阶次 n_0+1，如果拟合误差显著降低则进一步选择更高的阶次，否则接受 n_0。

例 5-18　试为例 5-16 的模型设计最优的 Charef 滤波器。

解　选择一个感兴趣的频率段 $\omega = \ 10^{-2}, 10^3$ rad/s，就可以计算出无理传递函数的频域响应数据；再选择最优 Charef 滤波器的预期阶次为 6，就可以由下面语句设计出最优 Charef 滤波器：

```
>> G01=tf(1,[1/1.6 1]); G02=tf(1,[1/6.2 1]);
   w=logspace(-2,3); H1=frd(G01,w); h1=H1.ResponseData;
   H2=frd(G02,w); h2=H2.ResponseData; H=H1;
   h=h1.^0.6.*h2.^0.3; h=h(:); G1=charef_opt(w,6,h)
```

得出的最优 Charef 滤波器模型为

$$G_1(s)=\frac{\begin{array}{c}0.00096744(s+6.692)(s+14.16)(s+46.48)(s+126.5)\times\\(s+299.6)(s+929)(s+3537)\end{array}}{(s+2.274)(s+7.892)(s+17.67)(s+47.71)(s+148)(s+350.5)(s+676.4)}$$

两个模型的Bode图比较如图5-34所示。可以看出，这时得出的6阶滤波器模型的逼近效果与例5-16得出的17阶滤波器的效果相仿。

```
>> H.ResponseData=h; bode(H,G1,'--',w)
```

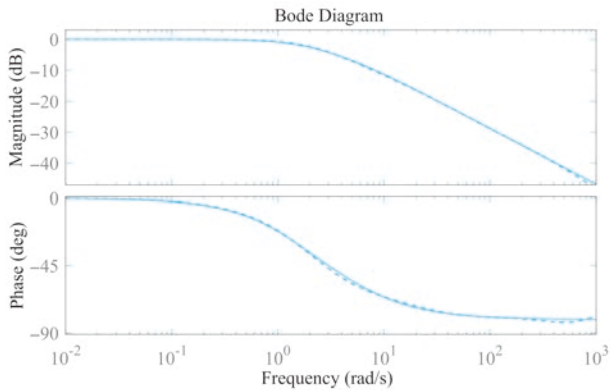


图5-34　频域响应比较

还可以指定一个更大的频率段 $10^{-2},10^{6}$ rad/s，从而设计出一个16阶的最优Charef滤波器，可以得到指定范围的完美拟合，如图5-35所示，在拟合结果上几乎看不出任何区别。

```
>> w=logspace(-2,7,100); H1=frd(G01,w); h1=H1.ResponseData;
   H2=frd(G02,w); h2=H2.ResponseData; H=H1;
   h=h1.^0.6.*h2.^0.3; h=h(:); H1.ResponseData=h;
   G2=charef_opt(w,15,h), bode(H1,G2,'--',w)
```

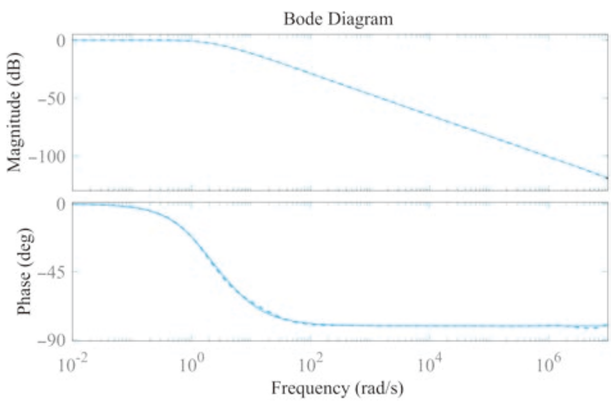


图5-35　更大区间的频域响应拟合比较

原始无理传递函数与滤波器模型的阶跃响应曲线如图5-36所示，两个滤波器的响应很接近，但与原始模型之间稍有差异。

```
>> f='1/(1+s/1.6)^0.6/(1+s/6.2)^0.3/s';
   [t,y]=INVLAP_new(f,0,3,1001);
   step(G1,G2,'--',t); hold on, plot(t,y,':'), hold off
```

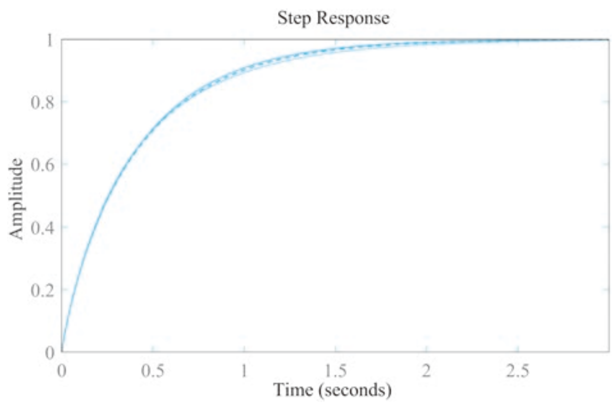


图5-36 阶跃响应比较

例5-19 现在考虑一个更复杂的无理传递函数模型：

$$G(s)=\frac{(1+s/6.2)^{0.3}}{(1+s/1.6)^{0.6}\left(s^{0.7}/3+1\right)^{0.2}}$$

试设计最优Charef滤波器并分析其效果。

解 在频率段 $10^{-4},10^{5}$ rad/s内产生一个向量，计算原始无理传递函数的频域响应，然后设计出最优Charef滤波器：

```
>> G01=tf(1,[1/1.6 1]); G02=tf([1/6.2 1],1);
   w=logspace(-4,5,200); H1=frd(G01,w); h1=H1.ResponseData;
   H2=frd(G02,w); h2=H2.ResponseData; H=H1;
   G3=fotf([1/3 1],[0.7 0],1,0); H3=bode(G3,w);
   h3=H3.ResponseData; h=h1.^0.6.*h2.^0.3.*h3.^0.2;
   H1.ResponseData=h; G1=charef_opt(w,11,h(:));
   bode(H1,G1,{1e-4,1e5})
```

原无理传递函数与整数阶拟合模型的Bode图如图5-37所示。可以看出，拟合效果还是很理想的。

利用数值Laplace反变换技术可得出无理系统的阶跃响应曲线，同时也可绘制近似模型 $G_1(s)$ 的阶跃响应曲线，如图5-38所示。可以看出，两个系统的响应是很接近的。

```
>> f='(1+s/6.2)^0.3/(1+s/1.6)^0.6/(s^0.7/3+1)^0.2';
   [t,y]=INVLAP_new(f,0,3,1000,0,'1/s'); step(G1,t); line(t,y)
```

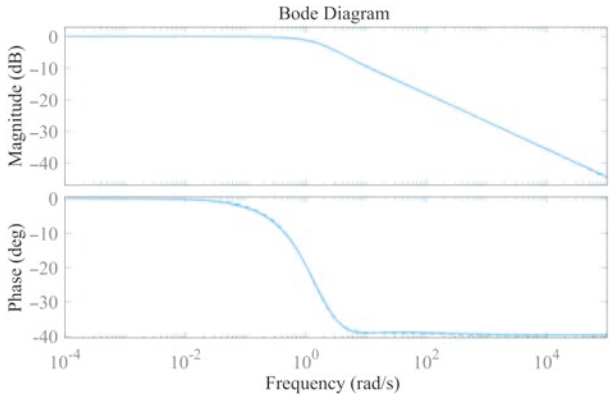


图 5-37　频域响应比较

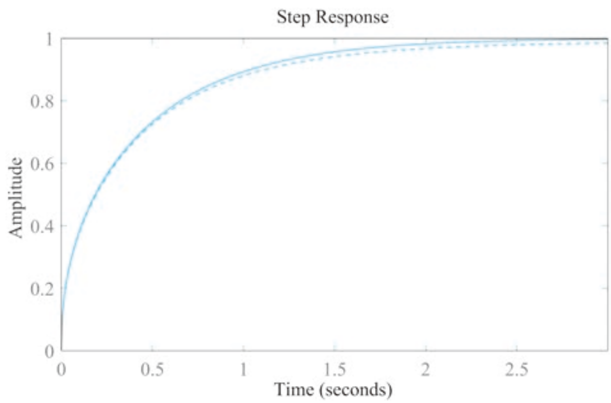


图 5-38　阶跃响应的比较

设计出来的12阶最优Charef滤波器为

$$G_1(s)=\frac{\begin{array}{c}0.0027751(s+3.591)(s+8.44)(s+23.97)(s+64.13)\times\\(s+179.1)(s+488.1)(s+1351)(s+3681)(s+1.019\times10^4)\times\\(s+2.772\times10^4)(s+7.561\times10^4)(s+2.515\times10^5)\end{array}}{\begin{array}{c}(s+1.757)(s+5.638)(s+14.97)(s+42.22)(s+114)\times\\(s+315.6)(s+860.2)(s+2380)(s+6487)\times\\(s+1.799\times10^4)(s+4.973\times10^4)(s+1.128\times10^5)\end{array}}$$

5.6　离散滤波器近似

前面介绍的是分数阶与无理系统的连续滤波器逼近。本节将介绍这些环节的离散滤波器近似方法。

定义 5-11 ▶ 离散滤波器

离散滤波器的数学形式为

$$H(z)=\frac{b_1+b_2z^{-1}+b_3z^{-2}+\cdots+b_{n+1}z^{-n}}{1+a_1z^{-1}+a_2z^{-2}+\cdots+a_mz^{-m}} \tag{5-6-1}$$

假设输入信号为$u(n)$,则经过滤波器后的输出信号可以由差分方程表示为

$$\begin{aligned}y(k)=&-a_1y(k-1)-a_2y(k-1)-\cdots-a_my(k-m)+\\&b_1u(k)+b_2u(k-1)+\cdots+b_{n+1}u(k-n)\end{aligned} \tag{5-6-2}$$

在实际应用中,经常使用的离散滤波器有下面两种特殊形式:

定义 5-12 ▶ FIR 滤波器

有限长冲激响应(finite impulse response,FIR)滤波器,需要将式(5-6-1)中的m值设置成$m=0$,这时a为标量,在控制领域也称移动平均模型(moving average,MA),这时用向量$\boldsymbol{b}$就可以表示该滤波器。

定义 5-13 ▶ IIR 滤波器

全极点无限长冲激响应(infinite impulse response,IIR)滤波器,也称为自回归(auto-regressive,AR)模型,这时$n=0$,即b为标量,这样用$\boldsymbol{a}$即可以表示该滤波器。和全极点IIR滤波器相比,FIR滤波器达到同样要求所需的滤波器阶数一般较高,但其优势是总可以设计出稳定的滤波器[14]。

5.6.1 FIR 滤波器逼近

对可以近似分数阶系统的离散滤波器来说,也可以考虑引入FIR滤波器[15]或IIR滤波器[16]的形式对其近似。利用MATLAB滤波器设计工具箱中的`filt()`函数,由Ivo Petráš教授开发的分数阶微积分器的FIR滤波器设计函数[17]。其核心部分清单(经过与本书风格一致的改写)如下:

```
function H=dfod2(n,T,r)   %这里根据需要对程序结构进行了修改
   arguments
      n(1,1) {mustBePositiveInteger}
      T(1,1) {mustBePositive}, r(1,1) double
   end
   if r>0
      bc=cumprod([1,1-((r+1)./[1:n])]); H=filt(bc,T^r,T);
   elseif r<0
      bc=cumprod([1,1-((-r+1)./[1:n])]); H=filt(T^(-r),bc,T);
end, end
```

其中,n为期望的滤波器阶次,T为滤波器的采样周期,r为所需的导数阶次,用该函

数可以直接设计出滤波器 H。若某信号经过滤波器 H，则可以得出该信号的 r 阶导数，经过滤波器得出的输出信号可以由 `lsim()` 函数计算出来。

例 5-20　试用离散 FIR 滤波器对例 5-5 中给出的信号求取分数阶微分。

解　用下面的语句可以直接设计出滤波器，并绘制出 Bode 图，如图 5-39 所示。由得出的频域响应看，$n=100$ 明显优于 $n=50$ 时的滤波器，但显然远不如前面介绍的用 Oustaloup 算法设计的连续滤波器。

```
>> t=0:0.001:pi;
   y=exp(-t).*sin(3*t+1); y3=glfdiff(y,t,0.5);
   G1=dfod2(50,0.001,0.5); G2=dfod2(100,0.001,0.5);
   bode(G1,G2,'--')
```

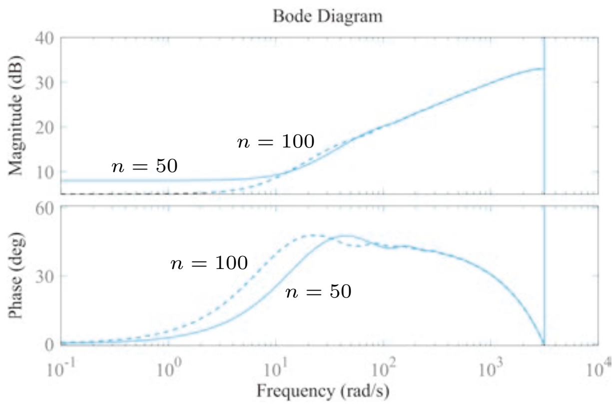


图 5-39　不同 (n,a) 组合下的数值微分器频域响应比较

在这些滤波器下还可以得出给定信号的分数阶微分，如图 5-40 所示，然而计算精度都很不理想，不能真正使用。

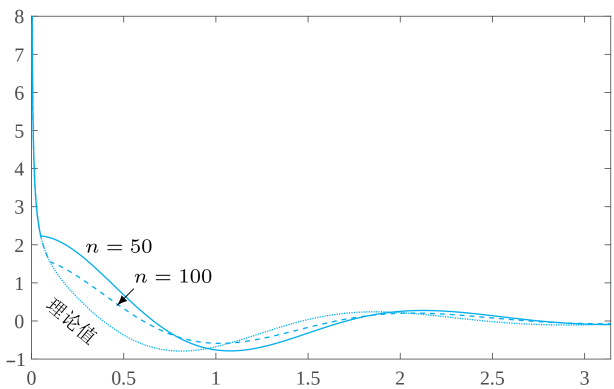


图 5-40　数值微分器的微分计算

```
>> y1=lsim(G1,y,t); y2=lsim(G2,y,t);
```

```
plot(t,y1,t,y2,'--',t,y3,':'), ylim([-1 8])
```

5.6.2　IIR 滤波器逼近

FIR 滤波器的阶次较高，所以可以考虑采用 IIR 滤波器生成分数阶微积分信号。从连续系统角度看，分数阶微积分可以用 Laplace 变换写成 $s^{\pm\gamma}$。对其进行离散化，就可以引入变换函数 $s=w\quad z^{-1}$ 近似分数阶微分或积分运算。

文献 [16] 给出了一种选择变换函数的新方法，该方法兼顾了 Simpson 积分算法和梯形积分算法，可以比较好地实现离散分数阶算子滤波器。

算法 5-9 ► IIR 滤波器设计

引入了加权系数 a，使得滤波器可以表示为

$$H(z)=aH_S(z)+(1-a)H_T(s),\quad a\in[0,1] \tag{5-6-3}$$

这样可以推导出 IIR 滤波器

$$G\quad z^{-1}\quad =k_0^{\pm\gamma}\quad \frac{1-z^{-2}}{(1+bz^{-1})^2}\quad ^{\pm\gamma} \tag{5-6-4}$$

其中，$b=(3+a-2\sqrt{3a})/(3-a)$，$k_0=6b/[T(3-a)]$，且 $\gamma\in(0,1)$，T 为离散化的采样周期。

对高阶微积分来说，可以先进行整数阶微积分，对其结果再利用各种滤波器进行分数阶微积分。文献 [16] 还给出了基于连分式的滤波器设计方法和求解函数。不过，该函数是基于 Maple 的连分式函数计算的，在当前版本 MATLAB 下已经不能使用；最新 MATLAB 版本下的 MuPAD 对连分式的支持也不正常，所以这里将其核心部分用 Padé 近似取代，修改后的函数如下：

```
function H=iir_pade(r,a,n,T)
   arguments, r(1,1) double, a(1,1){mustBeInRange(a,0,1)}=0
      n(1,1){mustBePositiveInteger}=3
      T(1,1){mustBePositive}=0.01
   end
   syms x, b=(3+a-2*sqrt(3*a))/(3-a);
   k0=6*b/T/(3-a); f=((1-x^2)/(1+b*x)^2)^r;
   c=taylor(f,x,'Order',2*n); c=sym2poly(c);
   c=c(end:-1:1); [N,D]=padefcn(c,n-1,n);
   H=k0^r*tf(N(end:-1:1),D(end:-1:1),'Variable','z^-1','Ts',T);
end
```

其中，r_0 为阶次，$a\in[0,1]$ 为加权系数，n 为滤波器阶次，T 为采样周期，这样可以由

该函数直接设计出离散的滤波器模型 H。下面将通过例子演示IIR滤波器的设计及分数阶微积分近似解法。

例 5-21　试用不同的 n, a 组合构造出 $s^{0.5}$ 阶滤波器，并比较其频域响应效果。

解　用前面给出的程序可以立即设计出如下的滤波器，并绘制出滤波器的Bode图，如图5-41所示。从得出的频率响应曲线看，若选择 $a=0.5$ 则相位曲线在给定的区域内不为恒值，所以拟合效果将不理想，但高频处赋值将有下降趋势，对噪声将有抑制作用。

```
>> H1=iir_pade(0.5,0,3,0.01); H2=iir_pade(0.5,0.5,3,0.01);
   H3=iir_pade(0.5,0,4,0.01); H4=iir_pade(0.5,0,5,0.01);
   bode(H1,'-',H2,'--',H3,':',H4,'-.')
```

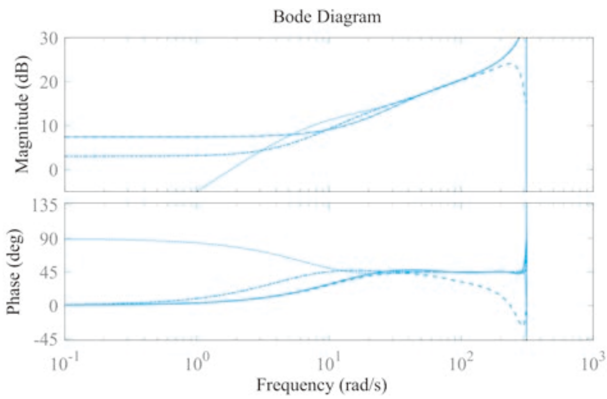


图 5-41　不同 (n, a) 组合下的频域频域响应拟合

用 $H_{4,a}\ \ z^{-1}$ 表示滤波器，可以得出设计出的各类滤波器。例如，

$$H_{4,0}\ \ z^{-1} = \frac{113.1 - 113.1z^{-1} - 56.57z^{-2} + 56.57z^{-3}}{8 - 8z^{-2} + z^{-4}}$$

如果提高滤波器的阶次，如，选择 $n = 10$，且 a 分别选择为0和0.5，就可以设计离散滤波器，得出的时域拟合结果如图5-42所示（由于 H_1 在高频增益过大，对噪声的放大很大，故在图中没有绘制）。可见，即使选取的 $a = 0.5$ 已经抑制了高频增益，总体时域响应曲线仍然不是很理想。

```
>> H1=iir_pade(0.5,0,10,0.01); H2=iir_pade(0.5,0.5,10,0.01);
   t=0:0.01:pi; y=exp(-t).*sin(3*t+1); y0=glfdiff(y,t,0.5);
   y1=lsim(H1,y,t); y2=lsim(H2,y,t); plot(t,y0,t,y2,'--')
```

和前面介绍的连续滤波器（尤其是Oustaloup滤波器）相比，离散滤波器拟合的频段远比连续滤波器窄，且往往伴随高频处的超高增益，不适合用于信号的分数阶导数的求取，不建议使用。

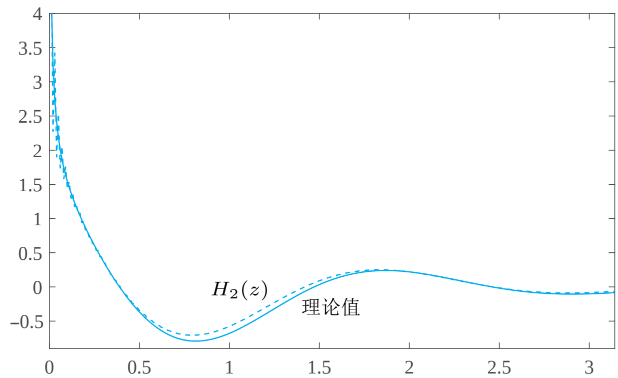


图 5-42 时域响应比较

5.6.3 基于阶跃或冲激响应不变性的离散滤波器

表 5-1 中列出了一组 MATLAB 函数，可以对分数阶微积分算子甚至分数阶无理传递函数设计离散滤波器。这些滤波器是在阶跃或冲激响应不变性基础上设计的[18]，FOTF 工具箱也提供了这 3 个函数，可以直接调用。

表 5-1 分数阶环节的离散滤波器

函数名	调用格式	函数说明
`irid_fod()`	G=`irid_fod(`α,T,N`)`	基于冲激响应不变性的 s^α 算子的滤波器
`srid_fod()`	G=`srid_fod(`α,T,N`)`	基于阶跃响应不变性的 s^α 算子的滤波器
`irid_folpf()`	G=`irid_folpf(`τ,α,T,N`)`	基于冲激响应不变性的 $(\tau s+1)^{-\alpha}$ 环节的滤波器

例 5-22 选择采样周期 $T=0.01\,\mathrm{s}$，并令滤波器阶次为 5，使利用前两个函数设计 0.5 阶积分器的离散滤波器，并比较其效果。

解 由下面的命令可以直接设计这两个滤波器，滤波器的 Bode 图如图 5-43 所示。

```
>> G1=irid_fod(-0.5,0.01,5); G2=srid_fod(-0.5,0.01,5);
   s=fotf('s'); H=bode(s^(-0.5),logspace(-3,3));
   bode(H), hold on, bode(G1,'--',G2,':'), hold off
```

由上面语句设计的离散滤波器如下：

$$G_1(z)=\frac{0.09354z^5-0.2395z^4+0.2094z^3-0.06764z^2+0.003523z+0.0008224}{z^5-3.163z^4+3.72z^3-1.966z^2+0.4369z-0.02738}$$

$$G_2(z)=\frac{2.38\times10^{-6}z^5+0.1128z^4-0.367z^3+0.4387z^2-0.2269z+0.04241}{z^5-3.671z^4+5.107z^3-3.259z^2+0.882z-0.05885}$$

可以将已知的 $y(t)=\mathrm{e}^{-t}\sin(3t+1)$ 信号馈入这两个滤波器，可以得出函数的积分曲线，如图 5-44 所示。从得出的效果看，$G_1(z)$ 滤波器的效果不佳，$G_2(z)$ 滤波器（即基于阶跃响应不变性设计的滤波器）对积分行为有比较好的逼近，如果提高滤波器的阶次，

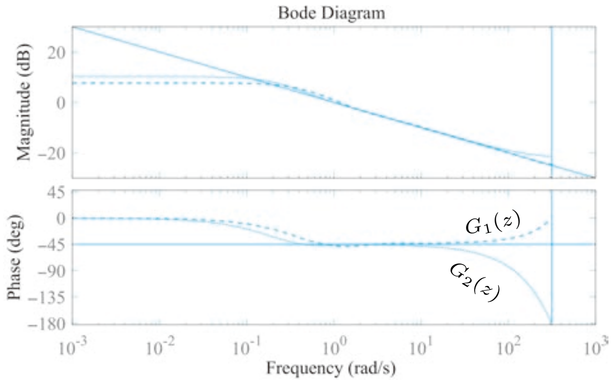


图 5-43　滤波器的 Bode 图比较

可能得到更好的效果。

```
>> t=0:0.01:pi; y=exp(-t).*sin(3*t+1);
   y0=glfdiff9(y,t,-0.5,5);
   y1=lsim(G1,y,t); y2=lsim(G2,y,t);
   plot(t,y0,t,y1,'--',t,y2,':')
```

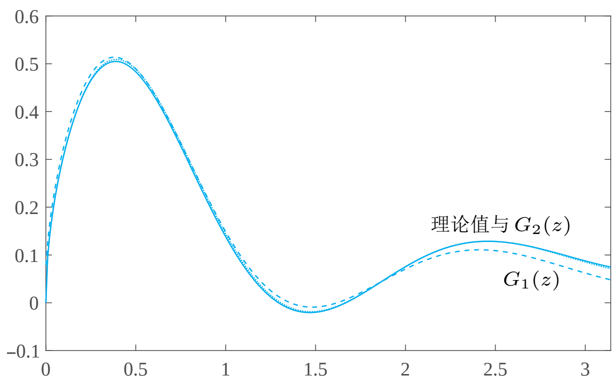


图 5-44　滤波器的积分效果比较

例 5-23　现在考虑例 4-9 的基准测试问题。试设计一个积分算子的离散滤波器，并和 Oustaloup 滤波器进行比较，评价其近似精度。

解　选择采样周期 $T=0.005\,\mathrm{s}$，可以设计两种离散滤波器，在这些滤波器下获得的分数阶积分效果与理论值的比较在图 5-45 所示。图中还给出了由 caputo9() 函数得出的计算结果，其最大误差为 0.0358，得出的曲线由点线表示，误差比较大；由阶跃响应不变性离散滤波器得出的最大误差为 0.0092，图中的虚线比较接近理论值；冲激响应不变性离散滤波器的最大误差为 0.0496，图中的点画线也存在较大的误差；而 Oustaloup 滤波器得出的最大误差为 0.0010，且不能在曲线上分辨出来，该曲线与理论值完全重合。可见，离散滤波器的效果比 Oustaloup 滤波器差很多。

```
>> f=@(t)1/gamma(1.3)*t.^0.3-2/gamma(2.3)*(t-1).^1.3.*(t>1);
```

```
y=@(t)t+1-(t-1).^2.*(t>1); T=0.005; t=[0:T:2]'; y0=f(t);
N=5; G=srid_fod(-0.7,T,N); ya=y(t); y1=lsim(G,y0,t)+1;
G=irid_fod(-0.7,T,N); ya=y(t); y4=lsim(G,y0,t)+1;
G1=ousta_fod(-0.7,15,1e-4,1e3); y3=lsim(G1,y0,t)+1;
t1=0:0.02:2; y01=f(t1); ya1=y(t1); y2=caputo9(y01,t1,-0.7,6)+1;
plot(t,ya,t,y1,'--',t1,y2,':',t,y4,'-.',t,y3)
max(abs(y1-ya)), max(abs(y2-ya1(:)))
max(abs(y3-ya)), max(abs(y4-ya))
```

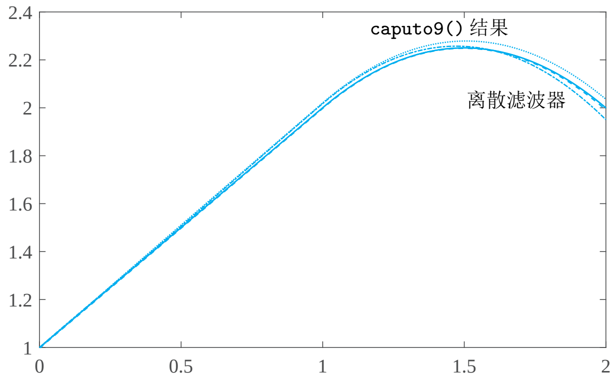


图 5-45　基准测试问题的求解比较

必须指出的是，在设计离散滤波器和控制器时，如果阶次取得过高，可能得出不稳定的结果。

本章习题

(1) 试用函数 contfrac() 近似分数阶积分算子 $1/s^{0.8}$，应用 Bode 图评价近似效果。

(2) 试用函数 carlson_fod() 近似分数阶积分算子 $1/s^{0.51}$，用得到的传递函数和前面介绍的函数 glfdiff9() 计算 ${}_0^{\mathrm{RL}}\mathscr{D}_t^{-0.51}\sin t^2$，检验得到的结果是否一致。

(3) 试用函数 matsuda_fod() 近似分数阶积分算子 $1/s^{0.6}$，设置不同的参数可以得到不同的传递函数。用这些传递函数计算 ${}_0^{\mathrm{RL}}\mathscr{D}_t^{-0.6}\mathrm{e}^{-t}$，应用第 2 章提供的函数 ml_func() 检验哪种参数下的计算结果更精确。

(4) 试用函数 ousta_fod() 近似分数阶积分算子 $1/s^{0.7}$，并计算 ${}_0^{\mathrm{RL}}\mathscr{D}_t^{-0.7}\sqrt{\sin t+3}$，设计数值算法检验之前的结果是否正确。

(5) 试用函数 ousta_fod() 近似分数阶微分算子 $s^{0.3}$，设置不同的参数可以得到不同的传递函数。用设计的这些传递函数计算 ${}_0^{\mathrm{RL}}\mathscr{D}_t^{0.3}\sqrt{\cosh t}$，应用第 4 章提供的函数 glfdiff9() 检验哪种参数下计算结果更精确。

(6) 试用函数 new_fod() 近似分数阶积分算子 $s^{0.2}$，并计算 ${}_0^{\mathrm{RL}}\mathscr{D}_t^{0.2}\left(1/\Gamma(t)\right)$。另外设

计一种计算 ${}_{0}^{\mathrm{RL}}\mathscr{D}_t^{0.2}\,(1/\Gamma(t))$ 的方法，检验之前的计算是否正确。

(7) 本章介绍了多种连续滤波器的设计方法，包括连分式方法、Matsuda–Fujii 方法、Carlson 方法、Oustaloup 方法及 Charef 方法等。同样选择 $\alpha=1/2$ 和 $\alpha=-1/2$，试通过调节设计参数的方法，找出能够在 $\omega\in[10^{-4},10^{4}]$ 频率段内得到的最好的拟合效果。同时，利用设计的滤波器评价分数阶导数与积分计算的精度。

(8) 考虑例 5-6 的问题，试找出合适的频率段与阶次，使得 Oustaloup 滤波器可以在 $t\in[0,5000]$ s 区间很好地拟合 $s^{-0.5}$ 分数阶算子的行为。

(9) 试将分数阶传递函数输入 MATLAB 工作空间[8]。

$$G(s)=\frac{\left(s^{0.3}+3\right)^2}{\left(s^{0.2}+3\right)\left(s^{0.4}+4\right)\left(s^{0.4}+3\right)}$$

(10) 假设典型单位负反馈控制系统的模型为

$$G(s)=\frac{0.8s^{1.2}+2}{1.1s^{1.8}+0.8s^{1.3}+1.9s^{0.5}+0.4},\quad G_{\mathrm{c}}(s)=\frac{1.2s^{0.72}+1.5s^{0.33}}{3s^{0.8}}$$

试将此模型输入 MATLAB 工作空间[8]。

(11) 将下面的分数阶传递函数模型输入 MATLAB 工作空间，并利用函数 `ousta_fod()` 和 `new_fod()` 对此模型进行拟合。

$$G(s)=\frac{0.5s^{0.9}+1}{s^3+23s^{2.5}+19s^{0.7}+1}$$

(12) 已知多变量系统的分数阶传递函数矩阵模型为

$$\boldsymbol{G}(s)=\begin{bmatrix}\dfrac{1}{1.35s^{1.2}+2.3s^{0.9}+1} & \dfrac{2}{4.13s^{0.7}+1}\\[2ex] \dfrac{1}{0.52s^{1.5}+2.03s^{0.7}+1} & -\dfrac{1}{3.8s^{0.8}+1}\end{bmatrix}$$

试将其输入 MATLAB 工作空间。

(13) 考虑下面的无理分数阶模型，应用频域响应近似方法找出它的有理近似模型。

$$G(s)=\frac{1}{\sqrt{s^2+s+1}}$$

(14) 考虑无理分数阶模型 $G(s)=1/\sqrt{s+1}$，试用函数 `charef_fod()` 找出其有理近似模型。

(15) 考虑无理模型 $G(s)=\mathrm{e}^{-\sqrt{s}}/(\sqrt{s}+1)$，试绘制阶跃响应曲线。已知系统阶跃响应的解析解如下，试评价不同计算步长下阶跃响应的计算精度。

$$y(t)=\mathrm{erfc}\left(\frac{1}{2\sqrt{t}}\right)-\mathrm{e}^{t+1}\mathrm{erfc}\left(\sqrt{t}+\frac{1}{2\sqrt{t}}\right)$$

(16) 考虑隐式无理系统模型 $G(s)=1/(5s+1)^{0.8}$。可以推导出其阶跃响应的解析解为 $y(t)=(t/5)^{0.8}\,\mathrm{E}_{1,1.8}^{0.8}\,(-t/5)$，试比较数值 Laplace 反变换、连续和离散滤波器，评价参数对求解精度的影响。

(17) 下面是离子交换聚合金属材料的辨识模型[19]：

$$G(s)=\frac{340}{s^{0.756}(s^2+3.85s+5880)^{1.15}}$$

试用本章提供的各种方法找出它的有理近似模型，并比较这些有理近似模型的频域特性。

(18) 考虑下面的无理分数阶模型，试用 Implicit model 模块搭建 Simulink 框图。设输入信号为 $u(t)=\cos t/\Gamma(t)$，利用搭建的 Simulink 框图计算模型的输出，并设法验证输出是否正确。

$$G(s)=\frac{1}{(1+2s)^{0.4}(1+4s)^{0.2}}$$

(19) 选择采样周期 $T=0.01\,\mathrm{s}$，滤波器阶次为 5，试用函数 `irid_fod()` 和 `srid_fod()` 设计 0.7 阶积分器的离散滤波器。应用得到的滤波器计算 ${}_0^{\mathrm{RL}}\mathscr{D}_t^{-0.7}\cosh t/\Gamma(t)$，并应用 `glfdiff9()` 函数检验计算结果。

(20) 试选择合适的整数阶传递函数，近似下面的分数阶模型，并比较频域响应拟合的效果[8]。

$$G(s)=\frac{25}{(s^2+8.5s+25)^{0.2}},\quad G(s)=\frac{562920(s+1.0118)^{0.6774}}{(s^2+54.7160s+590570)^{0.8387}}$$

(21) 已知分数阶模型[8]

$$G_1(s)=\frac{5}{s^{2.3}+1.3s^{0.9}+1.25},\quad G_2(s)=\frac{5s^{0.6}+2}{s^{3.3}+3.1s^{2.6}+2.89s^{1.9}+2.5s^{1.4}+1.2}$$

试求出能够较好拟合原始模型的整数阶模型，讨论采用何种阶次组合能得出较好的效果。试从频域响应和阶跃响应角度比较系统降阶模型。

参考文献

[1] Oustaloup A. La dérivation non entière：Théorie，synthèse et applications[M]. Paris：Hermès，1995.

[2] Petráš I. Fractional-order nonlinear systems：Modelling，analysis and simulation[M]. Beijing：Higher Education Press，2011.

[3] Chen Y Q，Vinagre B M. A new IIR-type digital fractional order differentiator[J]. Signal Processing，2003，83（11）：2359–2365.

[4] Petráš I，Podlubny I，O'Leary P. Analogue realization of fractional order controllers [R]. TU Košice：Fakulta BERG，2002.

[5] Matsuda K，Fujii H. $\mathcal{H}_\infty$-optimized wave-absorbing control： analytical and experimental results[J]. Journal of Guidance，Control and Dynammics，1993，16（6）：1146–1153.

[6] Oustaloup A，Levron F，Nanot F，et al. Frequency band complex non integer differentiator：characterization and synthesis[J]. IEEE Transactions on Circuits and Systems I：Fundamental Theory and Applications，2000，47（1）：25–40.

[7] Xue D Y, Zhao C N, Chen Y Q. A modified approximation method of fractional order system[C]. Proceedings of IEEE Conference on Mechatronics and Automation. Luoyang, China, 2006, 1043–1048.

[8] 薛定宇. 控制系统计算机辅助设计——MATLAB 语言与应用 [M]. 4 版. 北京: 清华大学出版社, 2022.

[9] Xue D Y, Chen Y Q. Suboptimum $\mathcal{H}_2$ pseudo-rational approximations to fractional-order linear time invariant systems. // Sabatier J, Agrawal O P, Machado J A T. Advances in fractional calculus — Theoretical developments and applications in physics and engineering[M]. Dordrecht: Springer, 2007.

[10] Sabatier J, Farges C. Analysis of fractional models physical consistency[J]. Journal of Vibration and Control, 2015, 23(6): 895–908.

[11] Monje C A, Chen Y Q, Vinagre B M, et al. Fractional-order systems and controls: Fundamentals and applications[M]. London: Springer, 2010.

[12] Charef A, Sun H H, Tsao Y Y, et al. Fractal system as represented by singularity function[J]. IEEE Transaction on Automatic Control, 1992, 37(9): 1465–1470.

[13] Meng L, Xue D Y. A new approximation algorithm of fractional order system models based optimization[J]. Journal of Dynamic Systems, Measurement and Control, 2012, 134(4): 504–511.

[14] MathWorks. Signal processing toolbox user's guide [Z], 2007.

[15] Tseng C-C, Pei S-C, Hsia S-C. Computation of fractional derivatives using Fourier transform and digital FIR differentiator[J]. Signal Processing, 2000, 80(1): 151–159.

[16] Chen Y Q, Vinagre B M. A new IIR-type digital fractional order differentiator[J]. Signal Processing, 2003, 83: 2359–2365.

[17] Petráš I. Digital fractional order differentiator/integrator — FIR type[OL]. [2022-10-2]. https://www.mathworks.com/matlabcentral/fileexchange/3672-digital-fractional-order-differentiator-integrator-iir-type.

[18] Chen Y Q. Impulse response invariant discretization of fractional order integrators/differentiators[OL]. [2022-10-2]. http://www.mathworks.cn/matlabcentral/fileexchange/21342-impulse-response-invariant-discretization-of-fractional-order-integrators-differentiators.

[19] Caponetto R, Dongola G, Fortuna L, et al. Fractional order systems: Modeling and control applications[M]. Singapore: World Scientific, 2010.