第 3 章

科学运算问题的MATLAB求解

控制系统的研究涉及各种各样的数学问题求解。例如，系统的稳定性分析需要求取矩阵的特征根，可控性与可观测性的判定需要求出矩阵的秩，而状态转移矩阵的求解需要矩阵指数的求解，这需要线性代数问题的求解。控制系统的仿真需要微分方程的求解，最优控制器设计涉及最优化问题的求解。如果能掌握利用MATLAB求解数学问题的方法和技术无疑会提高控制系统分析与设计的能力。

本章将介绍MATLAB语言在现代科学运算领域中的应用。这里所谓“运算”是指利用MATLAB不但能进行数值计算，还可以进行解析运算，所以从涵盖范围上比一般数值“计算”更广泛。MATLAB起源于线性代数的数值运算，在其长期的发展过程中，形成了几乎所有应用数学分支的求解函数与专门工具箱，并成功地引入了符号运算的功能，使得公式推导成为可能。所以一般情况下用一个语句就能直接获得数学问题的解。

MATLAB语言求解科学运算的功能是其广受科学工作者喜爱的重要原因，也是MATLAB语言的一大重要的特色。本章侧重于介绍以MATLAB为主要工具，直接求解和控制学科密切相关的数学问题的方法，为更好地探讨控制问题打下良好的基础。

3.1节介绍线性代数问题的解析与数值求解方法，介绍包括矩阵基本分析、矩阵变换的解析与数值解方法，并介绍矩阵函数的计算。3.2节介绍各种方程求解方法，包括线性代数方程、非线性方程和矩阵方程的求解方法，并试图求出多解方程全部的根。3.3节介绍一阶显式微分方程组的数值解方法，并介绍将一般常微分方程变换成可求解标准型的方法，还将介绍一般线性常系数微分方程的解析解方法。3.4节介绍最优化问题的数值求解方法，包括无约束最优化问题、有约束最优化问题的求解方法和曲线的最小二乘拟合方法。3.5节将介绍Laplace变换与z变换问题MATLAB求解方法。还将通过Laplace变换数值求解方法得出无理闭环系统的时域响应数值解。

本章涉及大量数学公式，但核心问题是引导读者如何避开数学问题本身及烦琐的底层解法，在MATLAB框架下直接获得可靠的解。关于应用MATLAB求解各种各样数学问题的详细内容可以参阅文献[1~5]。

3.1 线性代数问题的MATLAB求解

线性代数与矩阵运算是控制领域最重要的数学基础之一。很多线性代数问题是可以求取解析解的，不能求取解析解的问题往往也能得出数值解。本节将以数值解的介绍为主，其中很多函数同样可以利用MATLAB的符号运算工具箱中提供的相同函数名求出解析解。

3.1.1 矩阵的基本分析

矩阵的基本分析往往可以反映出矩阵的某些性质，比如在控制系统分析中，矩阵的特征值可以用来分析系统的稳定性，矩阵的秩可以用来分析系统的可控性和可观测性等，这里将系统地介绍矩阵基本分析的概念及其MATLAB实现。

1. 矩阵的行列式

MATLAB提供了内核函数`det(A)`，利用它可以直接求取矩阵$\boldsymbol{A}$的行列式（determinant）。如果矩阵$\boldsymbol{A}$为数值矩阵，则得出的行列式为数值计算结果；如果$\boldsymbol{A}$定义为符号矩阵，则`det()`函数将得出解析解。二者的区别是，对接近奇异的系统来说，解析解方法得出的结果更精确。不过，解析解算法也有一定的局限性，不适合大规模矩阵的求解。文献[3]通过实验探讨了随机符号矩阵的行列式求解，指出如果矩阵不含有变量，则可以在1 s内获得30×30符号矩阵行列式的解析解，而90×90矩阵耗时接近一分钟。

例3-1 试求Hilbert矩阵的行列式。

解 Hilbert矩阵的通项为$h_{i,j}=1/(i+j-1)$，用MATLAB的命令`hilb(n)`函数就可以在MATLAB工作空间中定义出来，而用`sym()`函数即可得出其符号型表示。下面的语句即可生成并计算出10阶Hilbert矩阵的行列式为：

```
>> H=hilb(10); d1=det(H)   %求解数值解
   H=sym(H); d2=det(H)     %先将矩阵变换成符号矩阵,再求解析解
```

用第1行语句可以求出数值解为$d_1=2.1644\times10^{-53}$，该结果不精确，故需要用解析解方法求解。由第2行命令可以得出解析解为

$$d_2=\frac{1}{46206893947914691316295628839036278726983680000000000}$$

例3-2 试求下面带有变量的Vandermonde矩阵的行列式。

$$\boldsymbol{A}=\begin{bmatrix}1&1&1\\a&b&c\\a^2&b^2&c^2\end{bmatrix}$$

解 先定义符号变量a、b、c,然后用下面的语句输入矩阵,并得出矩阵的特征多项式。

```
>> syms a b c                    %syms命令可以声明符号变量,用空格分隔
   A=[1,1,1; a,b,c; a^2,b^2,c^2];  %建立Vandermonde矩阵
   det(A); simplify(factor(ans))   %求行列式并进行因式分解
```

可以得出行列式的因式分解形式为$-(a-b)(a-c)(b-c)$。

2. 矩阵的迹

假设一个方阵为$\boldsymbol{A}=\{a_{ij}\}$,则矩阵$\boldsymbol{A}$的迹(trace)定义为该矩阵对角线上各个元素之和。由代数理论可知,矩阵的迹和该矩阵的特征值之和是相同的,矩阵$\boldsymbol{A}$的迹可以由MATLAB函数`trace(A)`求出。

3. 矩阵的秩

若矩阵所有的列向量中最多有r_c列线性无关,则称矩阵的列秩为r_c,如果$r_c=m$,则称$\boldsymbol{A}$为列满秩矩阵。相应地,若矩阵$\boldsymbol{A}$的行向量中有r_r个是线性无关的,则称矩阵$\boldsymbol{A}$的行秩为r_r。如果$r_r=n$,则称$\boldsymbol{A}$为行满秩矩阵。可以证明,矩阵的行秩和列秩是相等的,故称为矩阵的秩,记作$\mathrm{rank}(\boldsymbol{A})=r_c=r_r$,这时矩阵的秩(rank)为$\mathrm{rank}(\boldsymbol{A})$。矩阵的秩也表示该矩阵中行列式不等于0的子式的最大阶次,所谓子式,即为从原矩阵中任取k行及k列所构成的子矩阵。MATLAB提供了一个内核函数`rank(A,ε)`来用数值方法求取一个已知矩阵$\boldsymbol{A}$的数值秩,其中ε为机器精度。如果没有特殊说明,可以由`rank(A)`函数求出$\boldsymbol{A}$矩阵的秩。

4. 矩阵的范数

矩阵的常用范数(norm)定义为

$$||\boldsymbol{A}||_1=\max_{1\leqslant j\leqslant n}\sum_{i=1}^{n}|a_{ij}|,\quad ||\boldsymbol{A}||_2=\sqrt{s_{\max}(\boldsymbol{A}^{\mathrm{T}}\boldsymbol{A})},\quad ||\boldsymbol{A}||_\infty=\max_{1\leqslant i\leqslant n}\sum_{j=1}^{n}|a_{ij}| \tag{3-1-1}$$

其中,$s(\boldsymbol{X})$为$\boldsymbol{X}$矩阵的特征值,而$s_{\max}(\boldsymbol{A}^{\mathrm{T}}\boldsymbol{A})$即为$\boldsymbol{A}^{\mathrm{T}}\boldsymbol{A}$矩阵的最大特征值。事实上,$||\boldsymbol{A}||_2$为$\boldsymbol{A}$矩阵的最大奇异值。MATLAB提供了求取矩阵范数的函数`norm(A)`可以求出$||\boldsymbol{A}||_2$,矩阵的1-范数$||\boldsymbol{A}||_1$可以由`norm(A,1)`求解,矩阵无穷范数$||\boldsymbol{A}||_\infty$可以由`norm(A,inf)`求出。注意,该函数只能求数值解。

5. 矩阵的特征多项式、特征方程与特征根

矩阵$s\boldsymbol{I}-\boldsymbol{A}$的行列式可以写成一个关于$s$的多项式$C(s)$

$$C(s)=\det(s\boldsymbol{I}-\boldsymbol{A})=s^n+c_1s^{n-1}+\cdots+c_{n-1}s+c_n \tag{3-1-2}$$

这样的多项式$C(s)$称为矩阵$\boldsymbol{A}$的特征多项式,其中系数$c_i,i=1,2,\cdots,n$称为矩阵的特征多项式系数。

MATLAB提供了求取矩阵特征多项式系数的函数`p=poly(A)`,而返回的$\boldsymbol{p}$为一个行向量,其各个分量为矩阵$\boldsymbol{A}$的降幂排列的特征多项式系数。该函数的另外

一种调用格式是：如果给定的$\boldsymbol{A}$为向量，则假定该向量是一个矩阵的特征根，由此求出该矩阵的特征多项式系数，如果向量$\boldsymbol{A}$中有无穷大或NaN值，则首先剔除。

对方阵$\boldsymbol{A}$，如果存在一个非零的向量$\boldsymbol{x}$，且有一个标量λ满足$\boldsymbol{Ax}=\lambda\boldsymbol{x}$，则称$\lambda$为$\boldsymbol{A}$矩阵的一个特征值（eigenvalue），而$\boldsymbol{x}$为对应于特征值$\lambda$的特征向量，严格说来，$\boldsymbol{x}$应该称为$\boldsymbol{A}$的右特征向量。如果矩阵$\boldsymbol{A}$的特征值不包含重复的值，则对应的各个特征向量为线性无关的，这样由各个特征向量可以构成一个非奇异的矩阵，如果用它对原始矩阵作相似变换，则可以得出一个对角矩阵。矩阵的特征值与特征向量由MATLAB提供的函数eig()可以容易地求出，该函数的调用格式为[V,D]=eig(A)，其中$\boldsymbol{A}$为给定的矩阵，解出的$\boldsymbol{D}$为一个对角矩阵，其对角线上的元素为矩阵$\boldsymbol{A}$的特征值，而每个特征值对应于$\boldsymbol{V}$矩阵中的一列，称为该特征值的特征向量。MATLAB的矩阵特征值的结果满足$\boldsymbol{AV}=\boldsymbol{VD}$，且$\boldsymbol{V}$矩阵每个特征向量各元素的平方和（即列向量的2范数）均为1。如果调用该函数时至多只给出一个返回变元，则将返回矩阵$\boldsymbol{A}$的特征值。即使$\boldsymbol{A}$为复数矩阵，也照样可以由eig()函数得出其特征值与特征向量矩阵。

6. 多项式及多项式矩阵的求值

可以由C=polyval(a,x)命令求取基于点运算的多项式值，求出$\boldsymbol{C}=a_1\boldsymbol{x}.\hat{}n+a_2\boldsymbol{x}.\hat{}(n-1)+\cdots+a_{n+1}$，其中a=[$a_1,a_2,\cdots,a_n,a_{n+1}$]为多项式系数降幂排列构成的向量，$\boldsymbol{x}$为一个任意的向量或矩阵。

如果想求取真正的矩阵多项式的值，亦即

$$\boldsymbol{B}=a_1\boldsymbol{A}^n+a_2\boldsymbol{A}^{n-1}+\cdots+a_n\boldsymbol{A}+a_{n+1}\boldsymbol{I} \tag{3-1-3}$$

其中，$\boldsymbol{I}$为和$\boldsymbol{A}$同阶次的单位矩阵，则可以用B=polyvalm(a,A)。

7. 矩阵的逆与广义逆

对一个已知的$n\times n$非奇异方阵$\boldsymbol{A}$，如果有一个$\boldsymbol{C}$矩阵满足

$$\boldsymbol{AC}=\boldsymbol{CA}=\boldsymbol{I} \tag{3-1-4}$$

其中$\boldsymbol{I}$为单位矩阵，则称$\boldsymbol{C}$矩阵为$\boldsymbol{A}$矩阵的逆矩阵，并记作$\boldsymbol{C}=\boldsymbol{A}^{-1}$。MATLAB下提供的C=inv(A)函数即可求出矩阵$\boldsymbol{A}$的逆矩阵$\boldsymbol{C}$。

如果用户想得出奇异矩阵或长方形矩阵的一种“逆”阵，则需要使用广义逆的概念。对一个给定的矩阵$\boldsymbol{A}$，存在一个唯一的矩阵$\boldsymbol{M}$同时满足3个条件：

（1）$\boldsymbol{AMA}=\boldsymbol{A}$。

（2）$\boldsymbol{MAM}=\boldsymbol{M}$。

（3）$\boldsymbol{AM}$与$\boldsymbol{MA}$均为对称矩阵。

这样的矩阵$\boldsymbol{M}$称为矩阵$\boldsymbol{A}$的Moore–Penrose广义逆矩阵，又称伪逆（pseudo inverse），记作$\boldsymbol{M}=\boldsymbol{A}^{+}$。更进一步对复数矩阵$\boldsymbol{A}$来说，若得出的广义逆矩阵的第三个条件扩展为$\boldsymbol{MA}$与$\boldsymbol{AM}$均为Hermit矩阵，则这样构造的矩阵也是唯一的。

MATLAB 下 `B=pinv(A)` 可以求出 $\boldsymbol{A}$ 矩阵的 Moore–Penrose 广义逆矩阵 $\boldsymbol{B}$。

3.1.2　矩阵的分解

矩阵的相似变换在状态空间分析中有着重要的意义。这里将介绍矩阵变换与分解方面的内容，如三角分解、奇异值分解等。

1. 矩阵的相似变换

假设有一个 $n \times n$ 的方阵 $\boldsymbol{A}$，并存在一个和它同阶的非奇异矩阵 $\boldsymbol{T}$，则可以对 $\boldsymbol{A}$ 矩阵进行如下的变换

$$\widehat{\boldsymbol{A}} = \boldsymbol{T}^{-1}\boldsymbol{A}\boldsymbol{T} \tag{3-1-5}$$

这种变换称为 $\boldsymbol{A}$ 的相似变换（similarity transform）。可以证明，变换后矩阵 $\widehat{\boldsymbol{A}}$ 的特征值和原矩阵 $\boldsymbol{A}$ 是一致的，亦即相似变换并不改变原矩阵的特征结构。

2. 矩阵的三角分解

矩阵的三角分解又称为 LU 分解，其目的是将一个矩阵分解成一个下三角矩阵 $\boldsymbol{L}$ 和一个上三角矩阵 $\boldsymbol{U}$ 的乘积，即 $\boldsymbol{A} = \boldsymbol{L}\boldsymbol{U}$，其中 $\boldsymbol{L}$ 和 $\boldsymbol{U}$ 矩阵可以分别写成

$$\boldsymbol{L} = \begin{bmatrix} 1 & & & \\ l_{21} & 1 & & \\ \vdots & \vdots & \ddots & \\ l_{n1} & l_{n2} & \cdots & 1 \end{bmatrix}, \quad \boldsymbol{U} = \begin{bmatrix} u_{11} & u_{12} & \cdots & u_{1n} \\ & u_{22} & \cdots & u_{2n} \\ & & \ddots & \vdots \\ & & & u_{nn} \end{bmatrix} \tag{3-1-6}$$

MATLAB 下提供了 `[L,U]=lu(A)` 函数，可以对给定矩阵 $\boldsymbol{A}$ 进行 LU 分解。如果采用数值算法，则主元素可能进行必要的换行，所以有时得出的 $\boldsymbol{L}$ 矩阵不是真正的下三角矩阵，而是其基本置换形式；如果 $\boldsymbol{A}$ 为符号型矩阵，则可以得出真正的三角分解。

3. 对称矩阵的 Cholesky 分解

如若 $\boldsymbol{A}$ 矩阵为对称矩阵，则仍然可以用 LU 分解的方法对之进行分解，对称矩阵 LU 分解有特殊的性质，即 $\boldsymbol{L} = \boldsymbol{U}^{\mathrm{T}}$，令 $\boldsymbol{D}^{\mathrm{T}} = \boldsymbol{L}$ 为一个下三角矩阵，则可以将原来矩阵 $\boldsymbol{A}$ 分解成 $\boldsymbol{A} = \boldsymbol{D}^{\mathrm{T}}\boldsymbol{D}$，其中 $\boldsymbol{D}$ 矩阵可以形象地理解为原 $\boldsymbol{A}$ 矩阵的平方根。对该对称矩阵进行分解可以采用 Cholesky 分解算法。MATLAB 提供了 `chol()` 函数来求取矩阵的 Cholesky 分解矩阵 $\boldsymbol{D}$，该函数的调用格式可以写成 `[D,p]=chol(A)`，式中，返回的 $\boldsymbol{D}$ 为 Cholesky 分解矩阵，且 $\boldsymbol{A} = \boldsymbol{D}^{\mathrm{T}}\boldsymbol{D}$；而 $p-1$ 为 $\boldsymbol{A}$ 矩阵中正定的子矩阵的阶次，如果 $\boldsymbol{A}$ 为正定矩阵，则返回 $p = 0$。

4. 矩阵的正交基

对于一类特殊的相似变换矩阵 $\boldsymbol{T}$ 来说，如果它本身满足 $\boldsymbol{T}^{-1} = \boldsymbol{T}^{*}$，其中 $\boldsymbol{T}^{*}$ 为 $\boldsymbol{T}$ 的 Hermit 共轭转置矩阵，则称 $\boldsymbol{T}$ 为正交矩阵，并将之记为 $\boldsymbol{Q} = \boldsymbol{T}$。可见正交矩阵 $\boldsymbol{Q}$ 满足下面的条件：

$$\boldsymbol{Q}^{*}\boldsymbol{Q} = \boldsymbol{I}, \quad \text{且 } \boldsymbol{Q}\boldsymbol{Q}^{*} = \boldsymbol{I} \tag{3-1-7}$$

其中 $\boldsymbol{I}$ 为 $n \times n$ 的单位矩阵。MATLAB 中提供了 `Q=orth(A)` 函数来求 $\boldsymbol{A}$ 矩阵的正交基 $\boldsymbol{Q}$，其中 $\boldsymbol{Q}$ 的列数即为 $\boldsymbol{A}$ 矩阵的秩。

5. 矩阵的奇异值分解

假设$\boldsymbol{A}$矩阵为$n\times m$矩阵,且$\mathrm{rank}(\boldsymbol{A})=r$,则$\boldsymbol{A}$矩阵可以分解为$\boldsymbol{A}=\boldsymbol{L}\boldsymbol{\Lambda}\boldsymbol{M}^{\mathrm{T}}$,其中$\boldsymbol{L}$和$\boldsymbol{M}$均为正交矩阵,$\boldsymbol{\Lambda}=\mathrm{diag}(\sigma_1,\sigma_2,\cdots,\sigma_n)$为对角矩阵,其对角元素$\sigma_1,\sigma_2,\cdots,\sigma_n$满足不等式$\sigma_1\geqslant\sigma_2\geqslant\cdots\geqslant\sigma_n\geqslant 0$。

MATLAB提供了直接求矩阵奇异值分解的函数`[L,A1,M]=svd(A)`,其中,$\boldsymbol{A}$为原始矩阵,返回的$\boldsymbol{A}_1$为对角矩阵,而$\boldsymbol{L}$和$\boldsymbol{M}$均为正交变换矩阵,并满足$\boldsymbol{A}=\boldsymbol{L}\boldsymbol{A}_1\boldsymbol{M}^{\mathrm{T}}$。

6. 矩阵的条件数

矩阵的奇异值大小通常决定矩阵的性态,如果矩阵的奇异值的差异特别大,则矩阵中某个元素有一个微小的变化将严重影响到原矩阵的参数,这样的矩阵又称为病态矩阵或坏条件矩阵,而在矩阵存在等于0的奇异值时称为奇异矩阵。矩阵最大奇异值$\sigma_{\max}$和最小奇异值$\sigma_{\min}$的比值又称为该矩阵的条件数,记作$\mathrm{cond}(\boldsymbol{A})$,即$\mathrm{cond}(\boldsymbol{A})=\sigma_{\max}/\sigma_{\min}$,矩阵的条件数越大,则对元素变化越敏感。矩阵的最大和最小奇异值还分别经常记作$\bar{\sigma}(\boldsymbol{A})$和$\underline{\sigma}(\boldsymbol{A})$。在MATLAB下也提供了函数`cond(A)`来求取矩阵$\boldsymbol{A}$的条件数。

3.1.3 矩阵指数$\mathrm{e}^{\boldsymbol{A}}$和指数函数$\mathrm{e}^{\boldsymbol{A}t}$

如果已知方阵$\boldsymbol{A}$,则该矩阵的指数定义为

$$\mathrm{e}^{\boldsymbol{A}}=\sum_{i=0}^{\infty}\frac{1}{i!}\boldsymbol{A}^i=\boldsymbol{I}+\boldsymbol{A}+\frac{1}{2!}\boldsymbol{A}^2+\frac{1}{3!}\boldsymbol{A}^3+\cdots+\frac{1}{m!}\boldsymbol{A}^m+\cdots \tag{3-1-8}$$

矩阵指数可由MATLAB给出的`expm(A)`函数立即求出,矩阵的其他函数,如$\cos\boldsymbol{A}$可以由`funm(A,@cos)`函数求出。值得指出的是:`funm()`函数采用了特征值、特征向量的求解方式。若矩阵含有重特征根,则特征向量矩阵为奇异矩阵,这样该函数将失效,这时应该考虑用Taylor幂级数展开的方式进行求解[6]。一般矩阵函数还可以考虑文献[1]中介绍的解析解方法。

例3-3 已知矩阵如下,试求该矩阵的指数与指数函数。

$$\boldsymbol{A}=\begin{bmatrix}-11 & -5 & 5\\ 12 & 5 & -6\\ 0 & 1 & 0\end{bmatrix}$$

解 矩阵指数$\mathrm{e}^{\boldsymbol{A}}$和指数函数$\mathrm{e}^{\boldsymbol{A}t}$可以由下面语句直接求出:

```
>> A=[-11,-5,5; 12,5,-6; 0,1,0]; expm(A)   %求数值解
   A=sym(A); expm(A), syms t; expm(A*t)     %求解析解和指数函数
```

指数矩阵的数值解、解析解与$\mathrm{e}^{\boldsymbol{A}t}$分别为

$$\mathrm{e}^{\boldsymbol{A}}\approx\begin{bmatrix}0.24737701 & 0.30723864 & 0.42774107\\ 0.14460292 & -0.00080693 & -0.51328929\\ 0.88197566 & 0.82052793 & 0.30643171\end{bmatrix}$$

$$\mathrm{e}^{\boldsymbol{A}}=\begin{bmatrix} 15\mathrm{e}^{-3}-20\mathrm{e}^{-2}+6\mathrm{e}^{-1} & 5\mathrm{e}^{-1}-15\mathrm{e}^{-2}+10\mathrm{e}^{-3} & 5\mathrm{e}^{-2}-5\mathrm{e}^{-3} \\ 24\mathrm{e}^{-2}-18\mathrm{e}^{-3}-6\mathrm{e}^{-1} & -12\mathrm{e}^{-3}-5\mathrm{e}^{-1}+18\mathrm{e}^{-2} & -6\mathrm{e}^{-2}+6\mathrm{e}^{-3} \\ 6\mathrm{e}^{-1}-12\mathrm{e}^{-2}+6\mathrm{e}^{-3} & -9\mathrm{e}^{-2}+4\mathrm{e}^{-3}+5\mathrm{e}^{-1} & -2\mathrm{e}^{-3}+3\mathrm{e}^{-2} \end{bmatrix}$$

$$\mathrm{e}^{\boldsymbol{A}t}=\begin{bmatrix} 15\mathrm{e}^{-3t}-20\mathrm{e}^{-2t}+6\mathrm{e}^{-t} & 5\mathrm{e}^{-t}-15\mathrm{e}^{-2t}+10\mathrm{e}^{-3t} & 5\mathrm{e}^{-2t}-5\mathrm{e}^{-3t} \\ 24\mathrm{e}^{-2t}-18\mathrm{e}^{-3t}-6\mathrm{e}^{-t} & -12\mathrm{e}^{-3t}-5\mathrm{e}^{-t}+18\mathrm{e}^{-2t} & -6\mathrm{e}^{-2t}+6\mathrm{e}^{-3t} \\ 6\mathrm{e}^{-t}-12\mathrm{e}^{-2t}+6\mathrm{e}^{-3t} & -9\mathrm{e}^{-2t}+4\mathrm{e}^{-3t}+5\mathrm{e}^{-t} & -2\mathrm{e}^{-3t}+3\mathrm{e}^{-2t} \end{bmatrix}$$

3.1.4　矩阵的任意函数计算

矩阵任意函数 $f(\boldsymbol{A})$ 的数学定义为

$$f(\boldsymbol{A})=f(0)\boldsymbol{I}+\dot{f}(0)\boldsymbol{A}+\frac{1}{2!}\ddot{f}(0)\boldsymbol{A}^2+\cdots+\frac{1}{m!}f^{(m)}(0)\boldsymbol{A}^m+\cdots \tag{3-1-9}$$

很多矩阵函数可以由 `funm()` 直接求解，也可以利用文献 [1] 给出的 `funmsym()` 函数直接求取。下面将通过例子演示该函数的调用方法。

例 3-4　已知矩阵 $\boldsymbol{A}$ 如下，试求复合矩阵函数 $\psi(\boldsymbol{A})=\mathrm{e}^{\boldsymbol{A}\cos\boldsymbol{A}t}$。

$$\boldsymbol{A}=\begin{bmatrix} -7 & 2 & 0 & -1 \\ 1 & -4 & 2 & 1 \\ 2 & -1 & -6 & -1 \\ -1 & -1 & 0 & -4 \end{bmatrix}$$

解　如果想求出 $\psi(\boldsymbol{A})=\mathrm{e}^{\boldsymbol{A}\cos\boldsymbol{A}t}$，则应该构造原型函数为 f=exp(x*cos(x*t))，这样就可以用下面语句直接求取矩阵函数了。

```
>> A=[-7,2,0,-1; 1,-4,2,1; 2,-1,-6,-1; -1,-1,0,-4];     %输入矩阵
   syms x t; A=sym(A); A1=funmsym(A,exp(x*cos(x*t)),x) %直接运算
   A2=expm(A*funm(A*t,@cos)) %新版本MATLAB下还可以使用这个命令
```

得出的结果是很冗长的，这里不给出显示的内容。

3.2　代数方程的MATLAB求解

方程求解是科学与工程领域到处都会遇到的问题。本节首先介绍各类线性方程的解析解与数值解方法，然后计算非线性方程的设置求解方法，并介绍矩阵方程的数值解法，还将试图得出多解矩阵方程全部的根。

3.2.1　线性方程求解问题及MATLAB实现

本节将介绍各种矩阵方程的求解方法，首先介绍矩阵逆和伪逆的求解方法，然后介绍一般线性代数方程的求解、Lyapunov 方程与 Riccati 方程求解问题。

1. 线性方程求解

前面已经介绍过矩阵的左除和右除，可以用来求解线性方程。若线性方程为 $\boldsymbol{AX}=\boldsymbol{B}$，则用 $\boldsymbol{X}=\boldsymbol{A}\backslash\boldsymbol{B}$ 即可求出方程的解；若方程为 $\boldsymbol{XA}=\boldsymbol{B}$，则用 $\boldsymbol{X}=\boldsymbol{B}/\boldsymbol{A}$ 即可求出方程的解。

更严格地，求解线性代数方程 $\boldsymbol{AX}=\boldsymbol{B}$ 应该分下面几种情况考虑[1]：

（1）若矩阵 $\boldsymbol{A}$ 为非奇异方阵，则方程的唯一解为 $\boldsymbol{X}$=inv($\boldsymbol{A}$)*$\boldsymbol{B}$。

（2）若 $\boldsymbol{A}$ 为奇异方阵，如果 $\boldsymbol{A}$ 和 $\boldsymbol{C}=[\boldsymbol{A},\boldsymbol{B}]$ 矩阵的秩均为 m，则线性代数方程有无穷多解，这时可以由 $\hat{\boldsymbol{x}}$=null($\boldsymbol{A}$) 得出齐次方程 $\boldsymbol{Ax}=\boldsymbol{0}$ 的基础解系，用 $\boldsymbol{x}_0$=pinv($\boldsymbol{A}$)*$\boldsymbol{B}$ 求出原方程的一个特解，这时定义符号变量 $a_1,a_2,\cdots,a_{n-m}$，则原方程的解为

$$\boldsymbol{x}=a_1*\hat{\boldsymbol{x}}(:,1)+a_2*\hat{\boldsymbol{x}}(:,2)+\cdots+a_{n-m}*\hat{\boldsymbol{x}}(:,n-m)+\boldsymbol{x}_0$$

（3）若 $\boldsymbol{A}$ 和 $\boldsymbol{C}=[\boldsymbol{A},\boldsymbol{B}]$ 矩阵的秩不同，则方程没有解，只能用 $\boldsymbol{x}$=pinv($\boldsymbol{A}$)*$\boldsymbol{B}$ 命令求出方程的最小二乘解。

例3-5 试求解线性代数方程组并验证得出的结果。

$$\begin{bmatrix}1&2&3&4\\2&2&1&1\\2&4&6&8\\4&4&2&2\end{bmatrix}\boldsymbol{X}=\begin{bmatrix}1\\3\\2\\6\end{bmatrix}$$

解 由下面的语句可以求出矩阵 $\boldsymbol{A}$ 和判定矩阵 $\boldsymbol{C}=[\boldsymbol{A},\boldsymbol{B}]$ 的秩。

```
>> A=[1 2 3 4; 2 2 1 1; 2 4 6 8; 4 4 2 2]; B=[1;3;2;6];
   C=[A B]; [rank(A), rank(C)]
```

通过检验秩的方法得出矩阵 $\boldsymbol{A}$ 和 $\boldsymbol{C}$ 的秩相同，都等于2，小于矩阵的阶次4，由此可以得出结论，原线性代数方程组有无穷多组解。如需求解原代数方程组，可以先求出化零空间 $\boldsymbol{Z}$，并得出满足方程的一个特解 $\boldsymbol{x}_0$。

```
>> syms a1 a2; Z=null(sym(A)); x0=sym(pinv(A))*B;
   x=a1*Z(:,1)+a2*Z(:,2)+x0, A*x-B
```

由上面结果可以写出方程的解析解如下，经检验可见误差矩阵为零。

$$\boldsymbol{x}=\alpha_1\begin{bmatrix}2\\-5/2\\1\\0\end{bmatrix}+\alpha_2\begin{bmatrix}3\\-7/2\\0\\1\end{bmatrix}+\begin{bmatrix}125/131\\96/131\\-10/131\\-39/131\end{bmatrix}=\begin{bmatrix}2a_1+3a_2+125/131\\-5a_1/2-7a_2/2+96/131\\a_1-10/131\\a_2-39/131\end{bmatrix}$$

如果采用 $\boldsymbol{D}$=rref($\boldsymbol{C}$) 函数，利用基本行变换得出方程的解析解，得出

$$\boldsymbol{D}=\begin{bmatrix}1&0&-2&-3&2\\0&1&5/2&7/2&-1/2\\0&0&0&0&0\\0&0&0&0&0\end{bmatrix}$$

这样可以写出方程的解为 $x_1=2x_3+3x_4+2, x_2=-5x_3/2-7x_4/2-1/2$，其中，$x_3,x_4$ 可以取任意常数。

2. $\boldsymbol{AXB}=\boldsymbol{C}$ 型方程

如果已知矩阵为相容矩阵，则可以利用Kronecker乘积技术将原方程变换为

$$(\boldsymbol{B}^{\mathrm{T}}\otimes\boldsymbol{A})\boldsymbol{x}=\boldsymbol{c} \tag{3-2-1}$$

其中，$\otimes$为Kronecker乘积，后面将给出定义与求解方法，$\boldsymbol{x}=\mathrm{vec}(\boldsymbol{X})$，$\boldsymbol{c}=\mathrm{vec}(\boldsymbol{C})$为矩阵$\boldsymbol{X}$，$\boldsymbol{C}$按列展开的列向量。其中，MATLAB命令`C(:)`可以将矩阵$\boldsymbol{C}$按列展开，获得$\boldsymbol{c}$列向量，而由`C=reshape(c,n,m)`命令则可以从向量变换回矩阵$\boldsymbol{C}$，如果维数n、m已知；若维数未知，则可以由`C1=reshape(c,size(C))`命令还原。

矩阵$\boldsymbol{A}$、$\boldsymbol{B}$的Kronecker乘积$\boldsymbol{A}\otimes\boldsymbol{B}$的数学定义为

$$\boldsymbol{A}\otimes\boldsymbol{B}=\begin{bmatrix} a_{11}\boldsymbol{B} & a_{12}\boldsymbol{B} & \cdots & a_{1n}\boldsymbol{B} \\ a_{21}\boldsymbol{B} & a_{22}\boldsymbol{B} & \cdots & a_{2n}\boldsymbol{B} \\ \vdots & \vdots & \ddots & \vdots \\ a_{m1}\boldsymbol{B} & a_{m2}\boldsymbol{B} & \cdots & a_{mn}\boldsymbol{B} \end{bmatrix} \tag{3-2-2}$$

矩阵的Kronecker乘积可以由`kron()`函数直接计算，`C=kron(A,B)`。

例3-6 试求解如下的矩阵方程。

$$\begin{bmatrix} 8 & 1 & 6 \\ 3 & 5 & 7 \\ 4 & 9 & 2 \end{bmatrix}\boldsymbol{X}\begin{bmatrix} 0 & 1 & 0 & 0 & 1 \\ 1 & 0 & 1 & 2 & 2 \\ 1 & 2 & 0 & 0 & 2 \\ 0 & 0 & 1 & 1 & 1 \\ 1 & 0 & 0 & 2 & 1 \end{bmatrix}=\begin{bmatrix} 0 & 2 & 0 & 0 & 2 \\ 1 & 2 & 1 & 0 & 0 \\ 2 & 1 & 1 & 1 & 0 \end{bmatrix}$$

解 可以利用下面的语句直接求解给出的方程，得出方程的解析解。

```
>> B=[0,1,0,0,1; 1,0,1,2,2; 1,2,0,0,2; 0,0,1,1,1; 1,0,0,2,1];
   C=[0,2,0,0,2; 1,2,1,0,0; 2,1,1,1,0];
   A=[8,1,6; 3,5,7; 4,9,2]; x=inv(kron(sym(B'),A))*C(:)
   X=reshape(x,size(C)), A*X*B-C
```

得出的结果如下，经验证该解确实满足原始方程，误差为零。

$$\boldsymbol{X}=\begin{bmatrix} 257/360 & 7/15 & -29/90 & -197/360 & -29/180 \\ -179/180 & -8/15 & 23/45 & 119/180 & 23/90 \\ -163/360 & -8/15 & 31/90 & 223/360 & 31/180 \end{bmatrix}$$

3. Lyapunov方程求解

下面的方程称为Lyapunov方程

$$\boldsymbol{A}\boldsymbol{X}+\boldsymbol{X}\boldsymbol{A}^{\mathrm{T}}=-\boldsymbol{C} \tag{3-2-3}$$

其中$\boldsymbol{A}$、$\boldsymbol{C}$为给定矩阵，且$\boldsymbol{C}$为对称矩阵。MATLAB下提供的`X=lyap(A,C)`可以立即求出满足Lyapunov方程的矩阵$\boldsymbol{X}$。该函数亦可用于不对称$\boldsymbol{C}$矩阵时方程的求解。

描述离散系统的Lyapunov方程标准型为

$$\boldsymbol{A}\boldsymbol{X}\boldsymbol{A}^{\mathrm{T}}-\boldsymbol{X}+\boldsymbol{Q}=\boldsymbol{0} \tag{3-2-4}$$

该方程可以直接用MATLAB现成函数`dlyap()`求解，即`X=dlyap(A,Q)`。

4. Sylvester方程求解

Sylvester方程实际上是Lyapunov方程的推广，有时又称为广义Lyapunov方程，该方程的数学表示为

$$\boldsymbol{AX}+\boldsymbol{XB}=-\boldsymbol{C} \tag{3-2-5}$$

其中$\boldsymbol{A}$、$\boldsymbol{B}$、$\boldsymbol{C}$为给定矩阵。MATLAB下提供的`X=lyap(A,B,C)`可以立即求出满足该方程的$\boldsymbol{X}$矩阵。

其实可以证明，利用Kronecker乘积可以将方程改写成线性代数方程。

$$\left(\boldsymbol{I}_m\otimes\boldsymbol{A}+\boldsymbol{B}^{\mathrm{T}}\otimes\boldsymbol{I}_n\right)\boldsymbol{x}=-\boldsymbol{c} \tag{3-2-6}$$

其中，$\boldsymbol{c}=\mathrm{vec}(\boldsymbol{C})$，$\boldsymbol{x}=\mathrm{vec}(\boldsymbol{X})$为矩阵按列展开的列向量。

文献[1,3]基于该方法给出了求取一般Lyapunov方程和Sylvester方程的解析解函数`lyapsym()`。针对不同的方程类型，可以由下面的格式分别求解。

```
X=lyapsym(sym(A),C)                    % 连续Lyapunov方程
X=lyapsym(sym(A),-inv(A'),Q*inv(A'))   % 离散Lyapunov方程
X=lyapsym(sym(A),B,C)                  % Sylvester方程
```

例3-7 求解下面的Sylvester方程。

$$\begin{bmatrix}8&1&6\\3&5&7\\4&9&2\end{bmatrix}\boldsymbol{X}+\boldsymbol{X}\begin{bmatrix}16&4&1\\9&3&1\\4&2&1\end{bmatrix}=\begin{bmatrix}1&2&3\\4&5&6\\7&8&0\end{bmatrix}$$

解 调用`lyap()`函数可以立即得出原方程的数值解。

```
>> A=[8,1,6; 3,5,7; 4,9,2]; B=[16,4,1; 9,3,1; 4,2,1];
   C=-[1,2,3; 4,5,6; 7,8,0]; X=lyap(A,B,C), norm(A*X+X*B+C)
```

可以得出该方程的数值解如下，经检验该解的误差为9.5337×10^{-15}，精度较高。

$$\boldsymbol{X}=\begin{bmatrix}0.0749&0.0899&-0.4329\\0.0081&0.4814&-0.2160\\0.0196&0.1826&1.1579\end{bmatrix}$$

如果想获得原方程的解析解，则可以使用下面的语句直接求解。

```
>> x=lyapsym(sym(A),B,C), norm(A*x+x*B+C)
```

得出方程的解如下，经检验误差为零，该解是原方程的解析解。

$$\boldsymbol{x}=\begin{bmatrix}1349214/18020305&648107/7208122&-15602701/36040610\\290907/36040610&3470291/7208122&-3892997/18020305\\70557/3604061&1316519/7208122&8346439/7208122\end{bmatrix}$$

5. Riccati方程求解

下面的方程称为Riccati代数方程。

$$\boldsymbol{A}^{\mathrm{T}}\boldsymbol{X}+\boldsymbol{XA}-\boldsymbol{XBX}+\boldsymbol{C}=\boldsymbol{0} \tag{3-2-7}$$

其中 $\boldsymbol{A}$、$\boldsymbol{B}$、$\boldsymbol{C}$ 为给定矩阵，且 $\boldsymbol{B}$ 为非负定对称矩阵，$\boldsymbol{C}$ 为对称矩阵，则可以通过 MATLAB 的 `are()` 函数得出 Riccati 方程的解：$\boldsymbol{X}$=`are(`$\boldsymbol{A}$`,`$\boldsymbol{B}$`,`$\boldsymbol{C}$`)`，且 $\boldsymbol{X}$ 为对称矩阵。离散系统的 Riccati 方程可以用 `dare()` 函数直接求解。

例 3-8　试求解并检验下面给出的 Riccati 方程。

$$\begin{bmatrix}-2&-1&0\\1&0&-1\\-3&-2&-2\end{bmatrix}\boldsymbol{X}+\boldsymbol{X}\begin{bmatrix}-2&1&-3\\-1&0&-2\\0&-1&-2\end{bmatrix}-\boldsymbol{X}\begin{bmatrix}2&2&-2\\-1&5&-2\\-1&1&2\end{bmatrix}\boldsymbol{X}+\begin{bmatrix}5&-4&4\\1&0&4\\1&-1&5\end{bmatrix}=\boldsymbol{0}$$

解　对比所述方程和式（3-2-7）给出的标准型可见

$$\boldsymbol{A}=\begin{bmatrix}-2&1&-3\\-1&0&-2\\0&-1&-2\end{bmatrix},\quad \boldsymbol{B}=\begin{bmatrix}2&2&-2\\-1&5&-2\\-1&1&2\end{bmatrix},\quad \boldsymbol{C}=\begin{bmatrix}5&-4&4\\1&0&4\\1&-1&5\end{bmatrix}$$

可以用下面的语句直接求解该方程，经验证得出解的误差为 1.4215×10^{-14}。

```
>> A=[-2,1,-3; -1,0,-2; 0,-1,-2]; B=[2,2,-2; -1 5 -2; -1 1 2];
   C=[5 -4 4; 1 0 4; 1 -1 5]; X=are(A,B,C); norm(A'*X+X*A-X*B*X+C)
```

原方程的数值解为

$$\boldsymbol{X}=\begin{bmatrix}0.98739&-0.798330&0.41887\\0.57741&-0.130790&0.57755\\-0.28405&-0.073037&0.69241\end{bmatrix}$$

3.2.2　一般非线性方程的求解

非线性方程如果不借助计算机也是难以求解的问题。本节将探讨一般非线性方程的准解析解方法，还将介绍二元非线性方程的图解方法与一般非线性方程的数值解方法。

1. 非线性方程的解析解法

MATLAB 符号运算工具箱中提供了 `solve()` 函数可以直接求出某些方程的解析解。如果方程没有解析解，还可以通过 `vpasolve()` 函数得出方程的高精度数值解。用户只需用符号表达式表示出所需求解的方程即可以直接得出方程的解析解或高精度数值解。值得指出的是，早期版本允许使用字符串描述方程，不过在新版本下这样的描述已经不再支持了，只能用符号表达式描述方程。该函数尤其适用于可以转化成多项式类方程的准解析解[1]。

例 3-9　试求解鸡兔同笼问题，其数学形式为下面给出的二元一次方程组。

$$\begin{cases}x+y=35\\2x+4y=94\end{cases}$$

解　求解鸡兔同笼问题有各种各样的方法，这里介绍基于 `vpasolve()` 函数的求解方法，需要首先将方程表示为符号表达式，然后直接求解方程即可。注意，方程中的等号需要用 == 表示；如果方程右侧为零，则可以略去 ==0。

```
>> syms x y; [x0,y0]=vpasolve(x+y==35,2*x+4*y==94)
```

例3-10 试求解下面给出的联立方程。

$$\begin{cases} \dfrac{1}{2}x^2+x+\dfrac{3}{2}+\dfrac{2}{y}+\dfrac{5}{2y^2}+\dfrac{3}{x^3}=0 \\ \dfrac{y}{2}+\dfrac{3}{2x}+\dfrac{1}{x^4}+5y^4=0 \end{cases}$$

解 用手工方法不可能求解原方程,所以应该考虑采用fsolve()函数直接求解。这样由下面的MATLAB语句可以得出原方程全部26个根,全部为共轭复数根。可以看出,求解这样复杂的方程对用户而言,和求解鸡兔同笼问题一样简单。

```
>> syms x y
   [x,y]=vpasolve(x^2/2+x+3/2+2/y+5/(2*y^2)+3/x^3==0,...
                  y/2+3/(2*x)+1/x^4+5*y^4==0);
   size(x)
```

2. 非线性方程的图解法

前面介绍过,满足隐式方程的解可以由fimplicit()函数或ezplot()函数直接绘制出来。如果想求出若干隐式方程构成的联立方程的解,则可以将这些隐式方程用fimplicit()函数或ezplot()在同一坐标系下绘制出来,这样,这些曲线的交点就是原联立方程的解,可以利用局部放大的方法把感兴趣的解从图形上读出来。具体求解方法可以参见例2-40中给出的演示。

3. 一般非线性方程的MATLAB数值解法

前面介绍了非线性方程组的两种解法,但这些解法均有一定的局限性。例如,solve()函数适合于求解可以转换成多项式形式的方程解,对一般超越方程没有较好的解决方法,而图解法适合求解一元、二元方程的解,且由于坐标轴数据显示只能保留有限小数点位数,所以求解精度较低,且只能得到方程的实根。

MATLAB提供了fsolve()函数,利用搜索的方法求解一般非线性方程组。该函数求解一般非线性方程组的步骤如下:

(1)**变换成方程的标准型**。$\boldsymbol{Y}=\boldsymbol{F}(\boldsymbol{X})=\boldsymbol{0}$,其中,$\boldsymbol{X}$和$\boldsymbol{F}$是同维数的矩阵。

(2)**用MATLAB描述方程**。可以采用匿名函数或M-函数直接描述方程。

(3)**选定初值求解方程**。求解函数调用格式为

[$\boldsymbol{x}$,f_1,flag,details]=fsolve(fun,$\boldsymbol{x}_0$,options)

其中,fun为步骤(2)中建立的方程组MATLAB表示,$\boldsymbol{x}_0$为给定的初值。变量$\boldsymbol{x}$为搜索出来的方程的解,f_1为该解代入原方程得出的误差值。返回的flag变量为标志量,如果其值大于0表示求解成功。details还将返回一些中间信息,如迭代步数等。如果想修改求解的误差限,则可以设置options模板,该值可以通过optimset()函数设定。

(4)**解的验证**。将得出的解代入方程$\boldsymbol{F}$=fun($\boldsymbol{X}$),由norm($\boldsymbol{F}$)检验结果。

例 3-11　试利用数值解法重新求解例 2-40 中给出的超越方程。

解　由于该方程是关于自变量 x 和 y 的，而标准型方程是针对自变量 $\boldsymbol{x}$ 的，所以可以考虑引入变量 $x_1 = x, x_2 = y$，这样原方程可以写成下面的标准型。

$$\boldsymbol{y} = \boldsymbol{f}(\boldsymbol{x}) = \begin{bmatrix} x_1^2 \mathrm{e}^{-x_1 x_2^2/2} + \mathrm{e}^{-x_1/2} \sin(x_1 x_2) \\ x_2^2 \cos(x_2 + x_1^2) + x_1^2 \mathrm{e}^{x_1 + x_2} \end{bmatrix} = \boldsymbol{0}$$

其中，$\boldsymbol{x} = [x_1, x_2]^{\mathrm{T}}$。这样可以由如下的匿名函数描述原始的非线性方程组。

```
>> f=@(x)[x(1)^2*exp(-x(1)*x(2)^2/2)+exp(-x(1)/2)*sin(x(1)*x(2));
          x(2)^2*cos(x(2)+x(1)^2)+x(1)^2*exp(x(1)+x(2))];
```

选定例子中得出的某随机点作为初始搜索点，则得出的解为 $x = 2.7800$，$y = -3.3902$，代入原方程可见误差达 10^{-11} 级别，求解精度大大增加。

```
>> x0=-5+10*rand(2,1); x=fsolve(f,x0); y=x(2), x=x(1)
```

改变初值 $\boldsymbol{x}_0$，则可以得到方程其他的实根。例如，由下面的语句可能得出另一对根 $x = 0, y = 1.5708$，代入原方程可见精度达到 10^{-7} 级，基本满足一般要求。反复使用上述的语句还可能得出很多其他的根。

```
>> x0=rand(2,1); x=fsolve(f,x0), f(x)
```

利用数值求解函数 `fsolve()`，还可以人为设置求解精度等控制量，例如可以用下面的语句直接求解方程，得到精确些的解。例如，用下面的语句重新求解原方程，则上述第一个根的精度可以增加到 10^{-14} 级。

```
>> x0=[2.7795; -3.3911]; ff=optimset;
   ff.TolX=1e-20; ff.TolFun=1e-20; x=fsolve(f,x0,ff), f(x)
```

如果选择复数初值，如 $\boldsymbol{x}_0 = [5 + 4\mathrm{j}, -1 + 2\mathrm{j}]$，则可以用下面语句求解方程，得出 $\boldsymbol{x} = [3.5470 + 0.0480\mathrm{j}, -3.5422 + 0.0479\mathrm{j}]$，误差也为 10^{-14} 级。

```
>> x0=[5+4i; -1+2i]; x=fsolve(f,x0,ff), f(x)
```

3.2.3　非线性矩阵方程的 MATLAB 求解

前面介绍的 `fsolve()` 函数可以直接用于求解非线性矩阵方程的一个根。如果给定初值，则可以通过这个初值搜索出其他的根。若给出多个初值，则可能求出其他的根。可以编写一个求解函数，一次性得出方程的多个根，该函数是文献 [4] 给出函数的改进形式，方程运行时间越长得出的结果可能越精确。

```
function more_sols(f,X0,A,tol,tlim,ff)
   arguments, f(1,1)
      X0, A(1,:)=1000, tol(1,1)=eps, tlim(1,1)=20, ff=optimset;
   end
   X=X0; if isscalar(A), b=0.5*A; a=-b; else, a=A(1); b=A(2); end
   ar=real(a);br=real(b); ai=imag(a);bi=imag(b); [n,m,i]=size(X0);
   ff.Display='off'; ff.TolX=tol; ff.TolFun=1e-20; tic
```

```
    if i==0, X0=zeros(n,m);    %判定零矩阵是不是方程的孤立解
       if norm(f(X0))<tol, i=1; X(:,:,i)=X0; end
    end
    while (1) %死循环结构,可以按Ctrl+C组合键中断,也可以等待
       x0=ar+(br-ar)*rand(n,m);        %生成搜索初值的随机矩阵
       if ~isreal(A), x0=x0+(ai+(bi-ai)*rand(n,m))*1i; end  %复矩阵
       try [x,~,key]=fsolve(f,x0,ff); catch, continue; end
       t=toc; if t>tlim, break; end
       if key>0, N=size(X,3);          %解集记录个数,若已记录,则放弃
          for j=1:N, if norm(X(:,:,j)-x)<1e-4; key=0;break; end,end
          if key==0                    %如果解比存储的更精确,则替换
             if norm(f(x))<norm(f(X(:,:,j))), X(:,:,j)=x; end
          elseif key>0, X(:,:,i+1)=x; %记录找到的根
             if norm(imag(x))>1e-8, xa=conj(x);        %共轭复数
                if norm(f(xa))<1e-8, i=i+1; X(:,:,i+1)=xa; end
             end, assignin('base','X',X); i=i+1, tic  %更新解集
          end, assignin('base','X',X);
end, end, end
```

该函数调用格式为more_sols(f,$\boldsymbol{X}_0$,A,ϵ,$t_{\lim}$,控制选项),其中,f为原函数的MATLAB表示,可以为匿名函数、MATLAB函数等,其他的参数可以采用默认的值,一般无经验的用户无须给出。$\boldsymbol{X}_0$是一个三维数组,表示以前已经得到的方程根,A为随机数初值范围,表示初值在$(-A/2, A/2)$范围内取均匀分布随机数,默认值为1000。ϵ为求解的默认精度要求,默认值为5倍的eps。$t_{\lim}$为允许的等待时间,默认值为20,表示20 s,即20 s内如果没有找到新解,则停止搜索。

观察图2-10(a)中的曲线交点,可以看出,$x=0, y=0$这个点不是由两条曲线演化而来,所以这类解称为方程的孤立根,是不能通过搜索方法得出的。在上面给出的求解函数中,首先尝试原点是不是孤立解,如果是则记录下来。另外,如果搜索到了一个复数根,则尝试一下其共轭复数是不是方程的根,如果是也记录下来,这样可以通过程序的使用效率。

由于使用了死循环while(1),只能由用户给出的中断命令Ctrl+C键停止运行,或等待$t_{\lim}$后没有新解后自动终止,这样该函数不能返回任何变元。为解决这样的问题,使用assignin()函数将得到的解$\boldsymbol{X}$和求解方程次数M写入MATLAB的工作空间。其中,$\boldsymbol{X}$为三维数组,$\boldsymbol{X}(:,:,i)$对应于第i个根。已找到根的个数可以由size($\boldsymbol{X}$,3)读出。

例3-12 试求出例3-8中给出的Riccati方程全部的根。

解 由于该方程是关于$\boldsymbol{X}$的二次型方程,从常理看该方程可能存在其他的根,但are()函数只能求出一个根。其他的根可以使用搜索的方法求出。现在可以使用more_sols()函

数来求解 Riccati 方程其他可能的根。首先，将矩阵方程用下面匿名函数直接描述出来。

```
>> A=[-2,1,-3; -1,0,-2; 0,-1,-2]; B=[2,2,-2; -1 5 -2; -1 1 2];
   C=[5 -4 4; 1 0 4; 1 -1 5]; f=@(X)A'*X+X*A-X*B*X+C;
   more_sols(f,zeros(3,3,0))
```

可见，上述函数的定义格式和 Riccati 方程的数学描述一样简洁。定义了方程函数，则直接调用 `more_sols()` 函数即可得出方程的解。该方程有 8 个根，所以显示 $i=8$ 以后就可以用 Ctrl+C 键结束程序运行，或等待程序自动停止。这时，工作空间中的三维数组 $\boldsymbol{X}$ 将返回方程的所有 8 个根，前面 `are()` 函数求出的解矩阵只是其中之一。

$$\boldsymbol{X}_1=\begin{bmatrix}0.8878 & -0.9608 & -0.2446\\ 0.1071 & -0.8984 & -2.5562\\ -0.0185 & 0.3604 & 2.4619\end{bmatrix},\quad \boldsymbol{X}_2=\begin{bmatrix}-0.1538 & 0.1086 & 0.4622\\ 2.0277 & -1.7436 & 1.3474\\ 1.9003 & -1.7512 & 0.5057\end{bmatrix}$$

$$\boldsymbol{X}_3=\begin{bmatrix}1.2212 & -0.4165 & 1.9775\\ 0.3577 & -0.4893 & -0.8863\\ -0.7414 & -0.8197 & -2.3559\end{bmatrix},\quad \boldsymbol{X}_4=\begin{bmatrix}-2.1032 & 1.2977 & -1.9697\\ -0.2466 & -0.3563 & -1.4899\\ -2.1493 & 0.7189 & -4.5464\end{bmatrix}$$

$$\boldsymbol{X}_5=\begin{bmatrix}0.9873 & -0.7983 & 0.4188\\ 0.5774 & -0.1307 & 0.5775\\ -0.2840 & -0.0730 & 0.6924\end{bmatrix},\quad \boldsymbol{X}_6=\begin{bmatrix}0.6664 & -1.3222 & -1.7200\\ 0.3120 & -0.5640 & -1.1910\\ -1.2272 & -1.6129 & -5.5939\end{bmatrix}$$

$$\boldsymbol{X}_7=\begin{bmatrix}-0.7618 & 1.3312 & -0.8400\\ 1.3182 & -0.3173 & -0.1718\\ 0.6371 & 0.7884 & -2.1996\end{bmatrix},\quad \boldsymbol{X}_8=\begin{bmatrix}23.947 & -20.667 & 2.4528\\ 30.146 & -25.983 & 3.6699\\ 51.967 & -44.911 & 4.6409\end{bmatrix}$$

如果将求解范围用复数表示，则还可以得出方程的复数根。可见，通过搜索可以得出方程全部 20 个根。

```
>> more_sols(f,X,1000+1000i)
```

利用这里给出的 `more_sols()` 函数可以求解由其他方法很难求解的矩阵方程，例如下面 Riccati 方程扩展形式得出的特殊方程也可以直接求解。

$$\boldsymbol{AX}+\boldsymbol{XD}-\boldsymbol{XBX}+\boldsymbol{C}=\boldsymbol{0} \tag{3-2-8}$$

$$\boldsymbol{AX}+\boldsymbol{XD}-\boldsymbol{XBX}^{\mathrm{T}}+\boldsymbol{C}=\boldsymbol{0} \tag{3-2-9}$$

例 3-13　试求解式(3-2-9)中给出的矩阵方程，其中

$$\boldsymbol{A}=\begin{bmatrix}2 & 1 & 9\\ 9 & 7 & 9\\ 6 & 5 & 3\end{bmatrix},\quad \boldsymbol{B}=\begin{bmatrix}0 & 3 & 6\\ 8 & 2 & 0\\ 8 & 2 & 8\end{bmatrix},\quad \boldsymbol{C}=\begin{bmatrix}7 & 0 & 3\\ 5 & 6 & 4\\ 1 & 4 & 4\end{bmatrix},\quad \boldsymbol{D}=\begin{bmatrix}3 & 9 & 5\\ 1 & 2 & 9\\ 3 & 3 & 0\end{bmatrix}$$

解　可以通过下面的语句直接求出该方程的 16 个实根，从略。

```
>> A=[2,1,9; 9,7,9; 6,5,3]; B=[0,3,6; 8,2,0; 8,2,8];
   C=[7,0,3; 5,6,4; 1,4,4]; D=[3,9,5; 1,2,9; 3,3,0];
   f=@(X)A*X+X*D-X*B*X.'+C; more_sols(f,zeros(3,3,0))
```

例3-14 例2-40中的方程也可以看成是一种特殊的矩阵方程,试求解该方程。

解 仿照例3-11中介绍的方法,由匿名函数描述该方程,则可以给出如下的命令,直接求解该方程。

```
>> f=@(x)[x(1)^2*exp(-x(1)*x(2)^2/2)+exp(-x(1)/2)*sin(x(1)*x(2));
          x(2)^2*cos(x(2)+x(1)^2)+x(1)^2*exp(x(1)+x(2))];
   more_sols(f,zeros(2,1,0),4*pi)
```

经过一段时间的等待,或用Ctrl+C键强制停止搜索过程,用下面的命令将得出的解叠印在图解法得出的图上,如图3-1所示,搜索到的解用圈表示。可见,绝大部分的交点均已找到。将所有的解代入原方程,则可以得出最大误差$e = 5.9706\times10^{-13}$,可见得出的解精度远远高于图解法。如果程序运行一段时间后自然停止,则可能找到该方程在感兴趣区域的全部41个根。

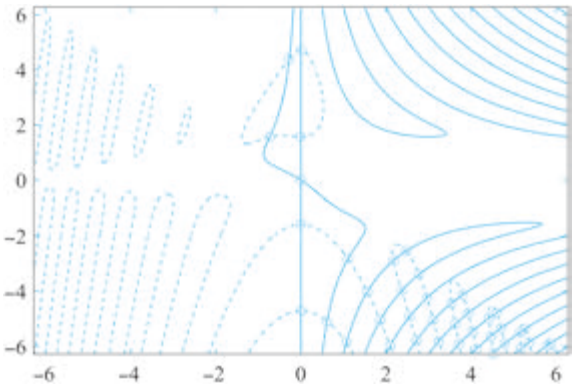


图3-1 联立方程图解法及搜索到的解

```
>> syms x y; f1=x^2*exp(-x*y^2/2)+exp(-x/2)*sin(x*y);
   f2=y^2*cos(y+x^2)+x^2*exp(x+y); L=2*pi;
   fimplicit([f1,f2],[-2*pi,2*pi],'MeshDensity',800), hold on
   x0=X(1,1,:); y0=X(2,1,:); ii=find(abs(x0)<=L & abs(y0)<=L);
   n=length(ii), x0=x0(ii);y0=y0(ii); plot(x0(:),y0(:),'o'), hold off
   F=[]; for i=1:length(x0), F=[F,f([x0(i),y0(i)])]; end, norm(F)
```

3.3 常微分方程问题的MATLAB求解

微分方程问题是动态系统仿真的核心,由强大的MATLAB语言可以对一阶微分方程组求取数值解,其他类型的微分方程可以通过合适的算法变换成可解的一阶微分方程组进行求解,这里将介绍微分方程的求解方法。

3.3.1 一阶常微分方程组的数值解法

假设一阶常微分方程组由下式给出

$$\dot{x}_i(t) = f_i(t, \boldsymbol{x}(t)),\ i = 1, 2, \cdots, n \qquad (3\text{-}3\text{-}1)$$

其中，$\boldsymbol{x}(t)$ 为状态变量 $x_i(t)$ 构成的向量，即 $\boldsymbol{x}(t) = [x_1(t), x_2(t), \cdots, x_n(t)]^{\mathrm{T}}$，常称为系统的状态向量，$n$ 称为系统的阶次，$f_i(\cdot)$ 为任意非线性函数，t 为时间变量。一般来说，非线性的微分方程是没有解析解的，只能采用数值方法，在初值 $\boldsymbol{x}(0)$ 下求解常微分方程组。

求解常微分方程组的数值方法是多种多样的，如常用的 Euler 法、Runge–Kutta 算法、Adams 线性多步法、Gear 算法等。为解决刚性（stiff）问题又有若干专用的刚性问题求解算法；另外，如需要求解隐式常微分方程组和含有代数约束的微分代数方程组时，则需要对方程进行相应的变换，方能进行求解。微分方程的数值解法详参文献 [5]，本节只给出简单微分方程的直接数值求解方法。

MATLAB 中给出了若干求解一阶常微分方程组的函数，如 `ode23()`（二阶三级 Runge–Kutta 算法）、`ode45()`（四阶五级 Runge–Kutta 算法）、`ode15s()`（变阶次刚性方程求解算法）等，其调用格式都是一致的：

[$\boldsymbol{t}, \boldsymbol{x}$]=ode45(Fun,tspan,$\boldsymbol{x}_0$,options,附加参数)

其中，$\boldsymbol{t}$ 为自变量构成的向量，一般采用变步长算法，返回的 $\boldsymbol{x}$ 是一个矩阵，其列数为 n，即微分方程的阶次，行数等于 $\boldsymbol{t}$ 的行数，每一行对应于相应时间点处的状态变量向量的转置。`Fun` 为用 MATLAB 编写的固定格式的 M-函数或匿名函数，描述一阶微分方程组，`tspan` 为数值解时的初始和终止时间等信息，$\boldsymbol{x}_0$ 为初始状态变量，`options` 为求解微分方程的一些控制参数，还可以将一些附加参数在求解函数和方程描述函数之间传递。下面将通过例子介绍微分方程求解过程。

例 3-15　试求解下面给出的著名的 Rössler 微分方程组，选定 $a = b = 0.2$, $c = 5.7$, 且 $x(0)=y(0)=z(0)=0$。

$$\begin{cases} \dot{x}(t) = -y(t) - z(t) \\ \dot{y}(t) = x(t) + ay(t) \\ \dot{z}(t) = b + [x(t) - c]\, z(t) \end{cases}$$

解　由于该方程是非线性微分方程，一般没有解析解，只能通过数值解的方法来研究该方程。引入新状态变量 $x_1(t)=x(t), x_2(t)=y(t), x_3(t)=z(t)$，则可将原微分方程改写成

$$\begin{cases} \dot{x}_1(t) = -x_2(t) - x_3(t) \\ \dot{x}_2(t) = x_1(t) + ax_2(t) \\ \dot{x}_3(t) = b + [x_1(t) - c]\, x_3(t) \end{cases} \quad \text{其矩阵形式为 } \dot{\boldsymbol{x}}(t) = \begin{bmatrix} -x_2(t) - x_3(t) \\ x_1(t) + ax_2(t) \\ b + [x_1(t) - c]x_3(t) \end{bmatrix}$$

若想求解这个微分方程，则需要用户自己编写一个 MATLAB 函数描述它。

```
function dx=rossler(~,x) %不显含时间，可以用~占位
   dx=[-x(2)-x(3); x(1)+0.2*x(2); 0.2+(x(1)-5.7)*x(3)]; %三行描述
end
```

对比此函数和给出的数学方程，应该能看出编写这样的函数还是很直观的。只要能得出一阶微分方程组，则可以立即编写出 MATLAB 函数来描述它。编写了该函数，就可

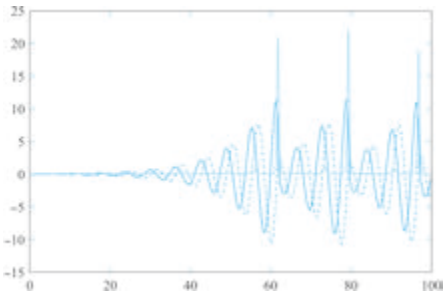
以将其存成rossler.m文件。除了用MATLAB函数描述微分方程之外，对简单问题还可以用匿名函数的方式描述原始的微分方程。

```
>> f=@(t,x)[-x(2)-x(3); x(1)+0.2*x(2); 0.2+(x(1)-5.7)*x(3)];
```

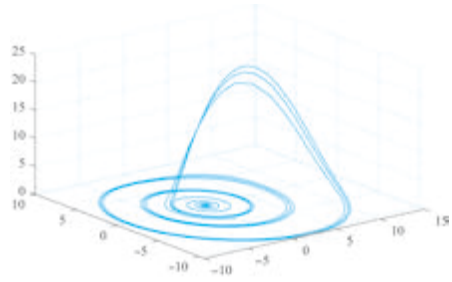
这样就可以用下面的MATLAB语句求出微分方程的数值解。

```
>> x0=[0; 0; 0]; [t,y]=ode45(@rossler,[0,100],x0); %解方程
      %或采用[t,y]=ode45(f,[0,100],x0)命令求解方程,得出一致结果
   plot(t,y)   %绘制各个状态变量的时间响应
   figure; plot3(y(:,1),y(:,2),y(:,3)), grid %绘制相空间轨迹
```

上面的命令将直接得出该微分方程在$t \in [0,100]$内的数值解，该数值解可以用图形更直观地表示出来，若绘制各个状态变量和时间之间的关系曲线，则得出如图3-2(a)所示的时域响应曲线，该方程研究的另外一种实用的表示是三维曲线绘制，用三维图形可以绘制出相空间曲线，如图3-2(b)所示。如果用comet3()函数取代plot3()，则可以看出轨迹走行的动画效果。



（a）状态变量的时间曲线



（b）系统响应的相空间表示

图3-2 Rössler方程的数值解表示

现在演示附加参数的使用方法，假设a、b、c这三个参数需要用外部命令给出，则可以按下面的格式写出一个新的M-函数来描述微分方程组。

```
function dx=rossler1(~,x,a,b,c) %加入附加参数
   dx=[-x(2)-x(3); x(1)+a*x(2); b+(x(1)-c)*x(3)];
end
```

这样就可以用下面的语句直接求解该方程并绘制曲线了。

```
>> a=0.2; b=0.2; c=5.7; %从函数外部定义这三个变量,无须修改函数本身
   [t,y]=ode45(@rossler1,[0,100],x0,[],a,b,c); %用附加参数解方程
```

这样编写M-函数有很多好处，例如想改变β等参数，没有必要修改M-函数，只需在求解该方程时将参数代入即可，这样会很方便。假设现在想研究$a=2$时的微分方程数值解，则可以给出下面的命令直接求解。

```
>> a=2; [t,y]=ode45(@rossler1,[0,100],x0,[],a,b,c);
```

事实上，对简单的微分方程而言，用匿名函数可以避开附加变量的使用，直接描述带有参数的微分方程。例如，下面的语句同样可以求解前面的微分方程。值得指出的是，

若想修改方程的参数，应该重新定义匿名函数，再调用方程求解函数。

```
>> a=0.2; b=0.2; c=5.7;
   f=@(t,x)[-x(2)-x(3); x(1)+a*x(2); b+(x(1)-c)*x(3)];
   [t,y]=ode45(f,[0,100],x0);  %利用匿名函数可以直接描述方程
```

在许多领域中，经常遇到一类特殊的常微分方程，其中一些解变化缓慢，另一些变化快，且相差较悬殊，用ode45()函数长时间得不出结果。这类方程常常称为刚性（stiff）方程。刚性问题一般不适合由ode45()这类函数求解，而应该采用求解函数ode15s()，其调用格式与ode45()一致。

3.3.2 常微分方程的转换

MATLAB下提供的微分方程数值解函数只能处理以一阶微分方程组形式给出的微分方程，所以在求解之前需要先将给定的微分方程变换成一阶微分方程组，而微分方程组的变换中需要选择一组状态变量，由于状态变量的选择是可以比较任意的，所以一阶显式微分方程组的变换也不是唯一的。这里介绍微分方程组变换的一般方法。

首先考虑单个高阶微分方程的处理方法，假设微分方程可以写成

$$f(t,y,\dot{y},\ddot{y},\cdots,y^{(n)})=0 \tag{3-3-2}$$

比较简单的状态变量选择方法是令$x_1=y,x_2=\dot{y},\cdots,x_n=y^{(n-1)}$，这样显然有$\dot{x}_1=x_2,\dot{x}_2=x_3,\cdots,\dot{x}_{n-1}=x_n$，另外，求解式（3-3-2），得出$y^{(n)}$的显式表达式，$y^{(n)}=\hat{f}(t,y,\dot{y},\cdots,y^{(n-1)})$，这时就可以写出该微分方程对应的一阶微分方程组为

$$\begin{cases}\dot{x}_i=x_{i+1}, & i=1,2,\cdots,n-1\\ \dot{x}_n=\hat{f}(t,x_1,x_2,\cdots,x_n)\end{cases} \tag{3-3-3}$$

这样，原微分方程就可以用MATLAB提供的常微分方程求解ode45()、ode15s()等函数直接求解了。

再考虑高阶微分方程组的变换方法，假设已知高阶微分方程组为

$$\begin{cases}f(t,x(t),\dot{x}(t),\cdots,x^{(m-1)}(t),x^{(m)}(t),y(t),\cdots,y^{(n-1)}(t),y^{(n)}(t))=0\\ g(t,x(t),\dot{x}(t),\cdots,x^{(m-1)}(t),x^{(m)}(t),y(t),\cdots,y^{(n-1)}(t),y^{(n)}(t))=0\end{cases} \tag{3-3-4}$$

则仍旧可以选择状态变量$x_1(t)=x(t)$，$x_2(t)=\dot{x}(t)$，$\cdots$，$x_m(t)=x^{(m-1)}(t)$，$x_{m+1}(t)=y(t),x_{m+2}(t)=\dot{y}(t),\cdots,x_{m+n}(t)=y^{(n-1)}(t)$，并将其代入式（3-3-4），则

$$\begin{cases}f(t,x_1(t),x_2(t),\cdots,x_m(t),\dot{x}_m(t),x_{m+1}(t),\cdots,x_{m+n}(t),\dot{x}_{m+n}(t))=0\\ g(t,x_1(t),x_2(t),\cdots,x_m(t),\dot{x}_m(t),x_{m+1}(t),\cdots,x_{m+n}(t),\dot{x}_{m+n}(t))=0\end{cases} \tag{3-3-5}$$

求解该方程则可以得出$\dot{x}_m(t)$与$\dot{x}_{m+n}(t)$，从而得出所需的一阶微分方程组，最终使用MATLAB中提供的函数求解这些高阶微分方程组。

例3-16 考虑著名的Van der Pol方程$\ddot{y}(t)+[y^2(t)-1]\dot{y}(t)+y(t)=0$，已知$y(0)=-0.2$，$\dot{y}(0)=-0.7$，试用数值方法求出的Van der Pol方程的解。

解 由给出的方程可知，因为它不是显式一阶微分方程组，所以不能直接求解，而必须先进行转换，再进行求解。选择状态变量$x_1(t)=y(t)$，$x_2(t)=\dot{y}(t)$，则原方程可以变换成

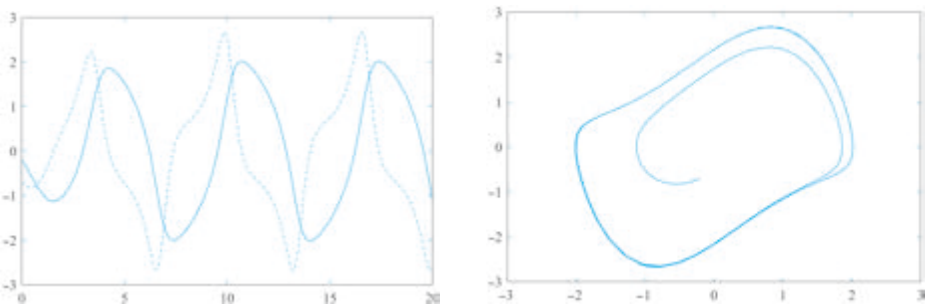
$$\begin{cases}\dot{x}_1(t)=x_2(t)\\ \dot{x}_2(t)=-[x_1^2(t)-1]x_2(t)-x_1(t),\end{cases}\qquad \text{其矩阵形式为}\dot{\boldsymbol{x}}\begin{bmatrix}x_2(t)\\ -[x_1^2(t)-1]x_2(t)-x_1(t)\end{bmatrix}$$

这样可以写出如下描述此方程的匿名函数。

```
>> f=@(t,x)[x(2); -(x(1)^2-1)*x(2)-x(1)];
```

由选定的状态变量可知，其初值可以描述成$\boldsymbol{x}_0=[-0.2,-0.7]^{\mathrm{T}}$，所以该方程最终可以由下面的语句直接求解并绘图，得出的时间响应曲线如图3-3(a)所示，相平面曲线如图3-3(b)所示。

```
>> x0=[-0.2; -0.7]; tn=20; [t,x]=ode45(f,[0,tn],x0);
   plot(t,x)                          %显示两个状态变量的时间曲线
   figure; plot(x(:,1),x(:,2))        %相平面曲线
```



(a) 状态变量的时间曲线　　(b) 系统响应的相平面表示

图3-3 Van der Pol方程的数值解表示

3.3.3 微分方程数值解的验证

前面介绍了微分方程的求解方法，求解步骤总结如下：

(1) **转换成标准型**。由于现有的求解函数只能求解一阶显式微分方程组，所以需要首先将原始的微分方程手工变换成标准形式。

(2) **描述标准型**。用MATLAB函数或者匿名函数描述原始微分方程。对简单问题而言，用匿名函数是最好的选择，但如果原始微分方程较烦琐，则只能用M-函数描述。

(3) **调用求解函数**。调用求解函数`ode45()`直接求解，对特殊的方程需要采用刚性微分方程求解函数，如`ode15s()`。

（4）**解的验证**。MATLAB 采用变步长算法求解微分方程，其关键的监测指标是容许相对误差限 RelTol 的设置。默认的 RelTol 控制量为 10^{-3}，相当于千分之一的误差，其值过大，所以应该采用较小的值。如果两次选择不同的 RelTol 值解的结果一致，则可以认为得出的解是正确的，否则应该试更小的控制量。另外，选择不同的求解算法也可以验证解的正确性。如果想追求双精度数据结构下最精确的结果，可以将 RelTol 的值设置成 3×10^{-14}。这些控制选项可以用下面的语句设置。

```
options=odeset; options.RelTol=1e-7;
```

例 3-17　已知 Apollo 卫星的运动轨迹 (x,y) 满足下面的方程 [7]。

$$\begin{cases} \ddot{x}(t)=2\dot{y}(t)+x(t)-\dfrac{\mu^*(x(t)+\mu)}{r_1^3(t)}-\dfrac{\mu(x(t)-\mu^*)}{r_2^3(t)} \\ \ddot{y}(t)=-2\dot{x}(t)+y(t)-\dfrac{\mu^* y(t)}{r_1^3(t)}-\dfrac{\mu y(t)}{r_2^3(t)} \end{cases}$$

其中，$\mu=1/82.45$，$\mu^*=1-\mu$，且

$$r_1(t)=\sqrt{(x(t)+\mu)^2+y^2(t)},\quad r_2(t)=\sqrt{(x(t)-\mu^*)^2+y^2(t)}$$

若已知初值为 $x(0)=1.2,\dot{x}(0)=0,y(0)=0,\dot{y}(0)=-1.04935751$，试求解该方程。

解　选择一组状态变量 $x_1(t)=x(t),x_2(t)=\dot{x}(t),x_3(t)=y(t),x_4(t)=\dot{y}(t)$，这样就可以得出标准型为

$$\begin{cases} \dot{x}_1(t)=x_2(t) \\ \dot{x}_2(t)=2x_4(t)+x_1(t)-\mu^*(x_1(t)+\mu)/r_1^3(t)-\mu(x_1(t)-\mu^*)/r_2^3(t) \\ \dot{x}_3(t)=x_4(t) \\ \dot{x}_4(t)=-2x_2(t)+x_3(t)-\mu^* x_3(t)/r_1^3(t)-\mu x_3(t)/r_2^3(t) \end{cases}$$

式中，$r_1(t)=\sqrt{(x_1(t)+\mu)^2+x_3^2(t)}$，$r_2(t)=\sqrt{(x_1(t)-\mu^*)^2+x_3^2(t)}$，且 $\mu=1/82.45$，$\mu^*=1-\mu$。

有了数学模型描述，则可以立即写出其相应的 MATLAB 函数如下：

```
function dx=apolloeq(~,x)
   mu=1/82.45; mu1=1-mu; r1=sqrt((x(1)+mu)^2+x(3)^2);
   r2=sqrt((x(1)-mu1)^2+x(3)^2); %中间变量赋值，不宜采用匿名函数
   dx=[x(2);
       2*x(4)+x(1)-mu1*(x(1)+mu)/r1^3-mu*(x(1)-mu1)/r2^3;
       x(4); -2*x(2)+x(3)-mu1*x(3)/r1^3-mu*x(3)/r2^3];
end
```

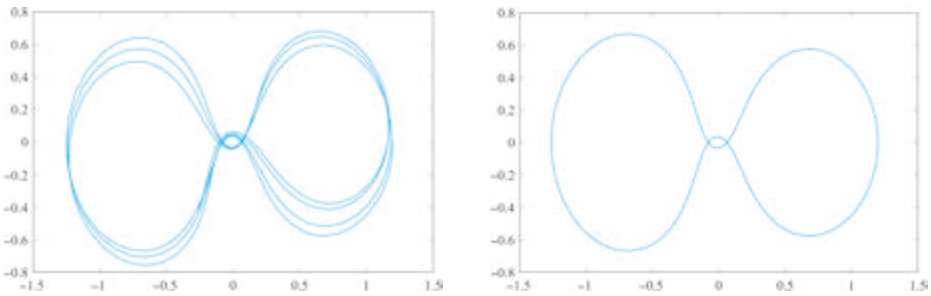
可以看出，由于在描述微分方程的语句中含有中间变量（如 $r_1(t)$、$r_2(t)$）的计算，所以，M-函数是比较方便的描述方法。

调用 ode45() 函数可以求出该方程的数值解，得出的轨迹如图 3-4(a)所示。

```
>> x0=[1.2; 0; 0; -1.04935751];
   [t,y]=ode45(@apolloeq,[0,20],x0); plot(y(:,1),y(:,3))
```

得出方程的数值解后，需要如下的语句对其检验，得出的新解如图3-4(b)所示，可见，这样得出的解和前面的解不同。再进一步减小RelTol值得到的解没有太大的变化，所以这样得出的解是正确的。

```
>> options=odeset; options.RelTol=1e-7;
   [t,y]=ode45(@apolloeq,[0,20],x0,options); plot(y(:,1),y(:,3))
```



（a）默认的解　　（b）正确的解

图3-4　Apollo卫星的轨迹曲线

如果实在想使用匿名函数描述微分方程，则可以先将$r_1(\boldsymbol{x})$和$r_1(\boldsymbol{x})$做成匿名函数，再用匿名函数表示原微分方程。

```
>> r1=@(x)sqrt((x(1)+mu)^2+x(3)^2);
   r2=@(x)sqrt((x(1)-mu1)^2+x(3)^2);
   f=@(t,x)[x(2);
       2*x(4)+x(1)-mu1*(x(1)+mu)/r1(x)^3-mu*(x(1)-mu1)/r2(x)^3;
       x(4); -2*x(2)+x(3)-mu1*x(3)/r1(x)^3-mu*x(3)/r2(x)^3];
```

MATLAB提供了大量的微分方程求解函数，网络上也可以下载很多求解函数。文献[5]对常用的求解函数做了比较，得出的结论为：对常规微分方程而言，ode45()函数的性能是最好的；对刚性微分方程而言，则建议优先采用ode15s()函数。另外，从求解精度看，可以将RelTol成员变量设置成3×10^{-14}，以期得出双精度数据结构下最精确的结果。

3.3.4　线性常微分方程的解析求解

由微分方程理论可知，常系数线性微分方程是存在解析解的，变系数的线性微分方程的可解性取决于其特征方程的可解性，一般是不可解析求解的，非线性的微分方程是不存在解析解的，在MATLAB语言中提供了dsolve()函数，可以用于线性常系数微分方程的解析解求解。求解微分方程时，首先应该用syms命令声明符号变量，以区别于MATLAB语言的常规数值变量，然后就可以用dsolve(表达式)命令直接求解了。类似于前面介绍的solve()和vpasolve()函数，微分方程应该由符号表达式描述。下面通过例子来演示该函数的使用方法。

例 3-18　试求解下面的常系数线性微分方程。

$$\frac{\mathrm{d}^4y(t)}{\mathrm{d}t^4}+11\frac{\mathrm{d}^3y(t)}{\mathrm{d}t^3}+41\frac{\mathrm{d}^2y(t)}{\mathrm{d}t^2}+61\frac{\mathrm{d}y(t)}{\mathrm{d}t}+30y(t)=\mathrm{e}^{-6t}\cos 5t$$

解　可以采用下面的 MATLAB 语句求解该微分方程。其中，引入了若干个中间变量，此外，微分方程的符号表达式中，等号应该由双等号表示。

```
>> syms t y(t)                    %声明符号变量
   y1=diff(y); y2=diff(y1); y3=diff(y3);
   Y=dsolve(diff(y)+11*y3+41*y2+61*y1+30*y==exp(-6*t)*cos(5*t));
   pretty(simplify(Y)) %以更好看的形式显示解析解结果
```

上面的语句得出结果的可读性不是很好，这里采用 LaTeX 变换后的结果为

$$y(t)=-\frac{79\mathrm{e}^{-6t}}{181220}\cos 5t+\frac{109\mathrm{e}^{-6t}}{181220}\sin 5t+C_1\mathrm{e}^{-t}+C_2\mathrm{e}^{-2t}+C_3\mathrm{e}^{-3t}+C_4\mathrm{e}^{-5t}$$

其中，C_i 为待定系数，应该由方程的初值或边值等求出，dsolve()函数可以直接求出带有初值或边值的微分方程解。例如已知方程的初值或边值条件为 $y(0)=1$，$\dot{y}(0)=1$，$\ddot{y}(0)=0$，$y^{(3)}(0)=0$，则可以由下面的语句求出方程的解。

```
>> Y=dsolve(diff(y)+11*y3+41*y2+61*y1+30*y==exp(-6*t)*cos(5*t),...
       y(0)==1,y1(0)==1,y2(0)==0,y3(0)==0);
```

则可以得出该微分方程的解析解为

$$y(t)=-\frac{79\mathrm{e}^{-6t}}{181220}\cos 5t+\frac{109\mathrm{e}^{-6t}}{181220}\sin 5t+\frac{611}{80}\mathrm{e}^{-t}-\frac{1562}{123}\mathrm{e}^{-2t}+\frac{921}{136}\mathrm{e}^{-3t}-\frac{443}{624}\mathrm{e}^{-5t}$$

3.4　最优化问题的MATLAB求解

最优化方法在系统仿真与控制系统计算机辅助设计中占有很重要的地位，求解最优化问题的数值算法有很多，MATLAB 中提供了各种各样的最优化问题求解函数，可以求解无约束最优化问题、有约束最优化问题及线性规划、二次型规划问题等，还实现了基于最小二乘算法的曲线拟合方法。

3.4.1　无约束最优化问题求解

无约束最优化问题的一般描述为

$$\min_{\boldsymbol{x}} f(\boldsymbol{x}) \tag{3-4-1}$$

其中，$\boldsymbol{x}=[x_1,x_2,\cdots,x_n]^{\mathrm{T}}$，该数学表示的含义亦即求取一个 $\boldsymbol{x}$ 向量，使得标量最优化目标函数 $f(\boldsymbol{x})$ 的值为最小，故这样的问题又称为最小化问题。其实，最小化是最优化问题的通用描述，它不失普遍性。如果要求解最大化问题，那么只需给目标函数 $f(\boldsymbol{x})$ 乘以 -1 就能立即将原始问题转换成最小化问题。

MATLAB 提供了基于单纯形算法[8] 求解无约束最优化的 fminsearch() 函数，该函数的调用格式为：[$\boldsymbol{x}$,f_{opt},key,c]=fminsearch(Fun,$\boldsymbol{x}_0$,options)，其中，

Fun为要求解问题的数学描述，它可以是一个MATLAB函数，也可以是一个函数句柄，$\boldsymbol{x}_0$为自变量的起始搜索点，需要用户自己去选择，options为最优化工具箱的选项设定；$\boldsymbol{x}$为返回的解；而f_{opt}是目标函数在$\boldsymbol{x}$点处的值。返回的key表示函数返回的条件，1表示已经求解出方程的解，而0表示未搜索到方程的解。返回的c为解的附加信息，该变元为一个结构体变量，其iterations成员变量表示迭代的次数，而其中的成员funcCount是目标函数的调用次数。MATLAB的最优化工具箱中提供的fminunc()函数与fminsearch()功能和调用格式均很相似，有时求解无约束最优化问题可以选择该函数。在第7、8章中将介绍基于数值最优化算法的最优控制器设计方法。

另外，如果决策变量$\boldsymbol{x}$需要满足$\boldsymbol{x}_{\text{m}} \leqslant \boldsymbol{x} \leqslant \boldsymbol{x}_{\text{M}}$前提条件，则可以采用后面将介绍的有约束最优化求解方法，或采用John D'Errico编写的fminsearchbnd()函数直接求解[9]，第11章将直接使用该函数求解最优控制器设计问题。

3.4.2 有约束最优化问题求解

有约束非线性最优化问题的一般描述为

$$\min_{\boldsymbol{x}} \quad f(\boldsymbol{x}) \qquad \text{s.t.} \begin{cases} \boldsymbol{A}\boldsymbol{x} \leqslant \boldsymbol{B} \\ \boldsymbol{A}_{\text{eq}}\boldsymbol{x} = \boldsymbol{B}_{\text{eq}} \\ \boldsymbol{x}_{\text{m}} \leqslant \boldsymbol{x} \leqslant \boldsymbol{x}_{\text{M}} \\ \boldsymbol{C}(\boldsymbol{x}) \leqslant \boldsymbol{0} \\ \boldsymbol{C}_{\text{eq}}(\boldsymbol{x}) = \boldsymbol{0} \end{cases} \tag{3-4-2}$$

其中，$\boldsymbol{x} = [x_1, x_2, \cdots, x_n]^{\text{T}}$。在约束条件中直接给出了线性等式约束$\boldsymbol{A}_{\text{eq}}\boldsymbol{x} = \boldsymbol{B}_{\text{eq}}$，线性不等式约束$\boldsymbol{A}\boldsymbol{x} \leqslant \boldsymbol{B}$，一般非线性等式约束$\boldsymbol{C}_{\text{eq}}(\boldsymbol{x}) = \boldsymbol{0}$，非线性不等式约束$\boldsymbol{C}(\boldsymbol{x}) \leqslant \boldsymbol{0}$和决策变量的上下界约束$\boldsymbol{x}_{\text{m}} \leqslant \boldsymbol{x} \leqslant \boldsymbol{x}_{\text{M}}$。注意，这里的不等式约束全部是$\leqslant$不等式，若原问题关系为$\geqslant$，则可以将不等式两端同时乘以$-1$就能将其转换成$\leqslant$不等式。

该数学表示的含义亦即求取一组$\boldsymbol{x}$向量，使得函数$f(\boldsymbol{x})$在满足全部约束条件的基础上最小化。而满足所有约束的问题称为可行解问题(feasible problem)。

MATLAB最优化工具箱中提供了一个fmincon()函数，专门用于求解各种约束下的最优化问题。该函数的调用格式为：

[$\boldsymbol{x}$, f_{opt}, key, c]=fmincon(Fun, $\boldsymbol{x}_0$, $\boldsymbol{A}$, $\boldsymbol{B}$, $\boldsymbol{A}_{\text{eq}}$, $\boldsymbol{B}_{\text{eq}}$, $\boldsymbol{x}_{\text{m}}$, $\boldsymbol{x}_{\text{M}}$, CFun, OPT)

其中，Fun为给目标函数写的M-函数，$\boldsymbol{x}_0$为初始搜索点。各个矩阵约束如果不存在，则应该用空矩阵来占位。CFun为给非线性约束函数写的M-函数，OPT为控制选项。最优化运算完成后，结果将在变元$\boldsymbol{x}$中返回，最优化的目标函数将在f_{opt}变元中返回，选项有时是很重要的。返回变元key若不是正数，则说明这时因故未发现原问题的解，可以考虑改变初值，或修改控制参数OPT，再进行寻优，以得出期望的最

优值。另外，如果发现最优化问题不是可行问题，则在求解结束后将给出提示："No feasible solution found（未找到可行解）"，这时可以考虑更换初值，重新尝试求解。Fun 变量还可以用结构体的形式描述原问题。

例 3-19　试求解下面的非线性最优化问题。

$$\boldsymbol{x}\ \begin{matrix}\min \\ \text{s.t.}\end{matrix}\ \begin{cases} \mathrm{e}^{x_1}(4x_1^2+2x_2^2+4x_1x_2+2x_2+1) \\ x_1+x_2\leqslant 0 \\ -x_1x_2+x_1+x_2\geqslant 1.5 \\ x_1x_2\geqslant -10 \\ -10\leqslant x_1,x_2\leqslant 10 \end{cases}$$

解　若想求解本最优化问题，应该用下面的语句先描述出目标函数和约束函数。

```
function y=c3exmobj(x)
   y=exp(x(1))*(4*x(1)^2+2*x(2)^2+4*x(1)*x(2)+2*x(2)+1);
end
function [c,ce]=c3exmcon(x)
   ce=[]; c=[x(1)+x(2); x(1)*x(2)-x(1)-x(2)+1.5; -10-x(1)*x(2)];
end
```

注意，约束函数应该返回两个变元，不等式约束 c 和等式约束 ce，其中第 2、3 条约束都应该先乘以 -1 变换成 $\leqslant$ 不等式。另外，由约束条件可知，第 1 条约束实际上是线性不等式约束，所以还可以用定义 $\boldsymbol{A}$、$\boldsymbol{B}$ 矩阵的形式来描述，但这样需要从 M-函数中先剔除第 1 条约束。调用非线性最优化问题求解函数可以得出如下结果。

```
>> A=[]; B=[]; Aeq=[]; Beq=[];
   xm=[-10; -10]; xM=[10; 10]; x0=[5;5];
   ff=optimset; ff.TolX=1e-10; ff.TolFun=1e-20;
   x=fmincon(@c3exmobj,x0,A,B,Aeq,Beq,xm,xM,@c3exmcon,ff)
```

该语句给出的显示为"Local minimum possible（可能是局部最小值）"。这样得出的"最优解"为 $\boldsymbol{x}^{\mathrm{T}}=[0.4195,0.4195]$，可以考虑用得出的"最优解"作为初值再进一步求解，如此可以利用循环结构得出原问题的真正最优解为 $\boldsymbol{x}^{\mathrm{T}}=[-9.5474,1.0474]$，循环次数为 $i=5$。

```
>> i=1; x=x0;
   while (1)
      [x,a,b]=fmincon(@c3exmobj,x,A,B,Aeq,Beq,xm,xM,@c3exmcon,ff);
      if b>0, break; end, i=i+1;    %如果求解成功则结束循环
   end
```

有约束最优化还有几种特殊的形式，如线性规划问题、二次型规划问题，可以使用最优化工具箱中的 linprog() 和 quadprog() 函数直接求解。此外，整数规划、0-1 规划等问题可以下载专门的工具求解[1]。

3.4.3 全局最优解的尝试

与线性规划等凸问题不同，常规的非线性规划问题可能含有局部最优解。如果初始搜索点选择不当，则很容易陷入局部最优解。仿照前面介绍的多解代数方程求解方法，很自然地考虑在`fminsearch()`或`fmincon()`函数外再加一层循环，每步循环在允许的区域内随机生成初始搜索点，则可以构造出新的求解函数`fminunc_global()`和`fmincon_global()`，与底层函数相比，这两个函数更可能获得原始问题的全局最优解。这两个函数的调用格式为

`[`$\boldsymbol{x}$`,`f_0`]=fminunc_global(fun,`a`,`b`,`n`,`N`)`

`[`$\boldsymbol{x}$`,`f_0`]=fmincon_global(fun,`a`,`b`,`n`,`N`,其他参数)`

其中，`fun`可以是结构体变量，也可以是目标函数的函数句柄；a与b为决策变量所在的区间，如果$\boldsymbol{x}_{\mathrm{m}}$与$\boldsymbol{x}_{\mathrm{M}}$为有限的向量，还可以直接将$a$和$b$设置成$\boldsymbol{x}_{\mathrm{m}}$和$\boldsymbol{x}_{\mathrm{M}}$向量；$n$为决策变量的个数；$N$为每次寻优中，底层函数`fminsearch()`或`fmincon()`的调用次数，通常情况下$N=5\sim10$就足够。

文献[4]对这两个函数与MATLAB全局优化工具箱提供遗传算法、粒子群算法等智能优化算法进行了对比研究。对测试的例子而言，这两个函数在全局性与求解耗时方面明显优于所比较的智能优化算法。这里只通过例子演示求解全局最优解函数的使用方法与优势。

例3-20 试找出下面非线性规划问题的全局最优解。

$$\min_{\boldsymbol{q},w,k\ \text{s.t.}}\ k\quad\begin{cases}q_3+9.625q_1w+16q_2w+16w^2+12-4q_1-q_2-78w=0\\16q_1w+44-19q_1-8q_2-q_3-24w=0\\2.25-0.25k\leqslant q_1\leqslant 2.25+0.25k\\1.5-0.5k\leqslant q_2\leqslant 1.5+0.5k\\1.5-1.5k\leqslant q_3\leqslant 1.5+1.5k\end{cases}$$

解 从给出的最优化问题看，这里要求解的决策变量为$\boldsymbol{q}$、w和k，而标准最优化方法只能求解向量型决策变量，所以应该做变量替换，把需要求解的决策变量由决策变量向量表示出来。对本例来说，可以引入$x_1=q_1,x_2=q_2,x_3=q_3,x_4=w,x_5=k$，另外，需要将一些不等式进一步处理，可以将原始问题手工改写成

$$\min_{\boldsymbol{x}\ \text{s.t.}}\ x_5\quad\begin{cases}x_3+9.625x_1x_4+16x_2x_4+16x_4^2+12-4x_1-x_2-78x_4=0\\16x_1x_4+44-19x_1-8x_2-x_3-24x_4=0\\-0.25x_5-x_1\leqslant -2.25\\x_1-0.25x_5\leqslant 2.25\\-0.5x_5-x_2\leqslant -1.5\\x_2-0.5x_5\leqslant 1.5\\-1.5x_5-x_3\leqslant -1.5\\x_3-1.5x_5\leqslant 1.5\end{cases}$$

从手工变换后的结果看，原始问题有两个非线性等式约束，没有不等式约束，所以可以由下面语句描述原问题的非线性约束条件。

```
function [c,ce]=c3mnls(x)
   c=[];       %非线性约束条件,其中,不等式约束为空矩阵
   ce=[x(3)+9.625*x(1)*x(4)+16*x(2)*x(4)+16*x(4)^2+12-...
       4*x(1)-x(2)-78*x(4);
       16*x(1)*x(4)+44-19*x(1)-8*x(2)-x(3)-24*x(4)];
end
```

原模型的线性约束可以写成线性不等式的矩阵形式 $\boldsymbol{A}\boldsymbol{x} \leqslant \boldsymbol{b}$，其中

$$\boldsymbol{A}=\begin{bmatrix} -1 & 0 & 0 & 0 & -0.25 \\ 1 & 0 & 0 & 0 & -0.25 \\ 0 & -1 & 0 & 0 & -0.5 \\ 0 & 1 & 0 & 0 & -0.5 \\ 0 & 0 & -1 & 0 & -1.5 \\ 0 & 0 & 1 & 0 & -1.5 \end{bmatrix},\ \boldsymbol{b}=\begin{bmatrix} -2.25 \\ 2.25 \\ -1.5 \\ 1.5 \\ -1.5 \\ 1.5 \end{bmatrix}$$

该问题没有线性等式约束，也没有决策变量的下界与上界约束，所以可以将这些约束条件用空矩阵表示，或直接采用结构体描述最优化问题，可以不用考虑这些约束的设置。为方便起见这里采用结构体形式描述原始问题。可以看出，这里的目标函数值为 x_5。采用 fmincon() 函数求解，很容易得出局部最优解 $x_5 = 1.1448$。现在试用全局最优化函数求解该问题。一般情况下，每次都能得出该问题的全局最优解为 $\boldsymbol{x} = [2.4544, 1.9088, 2.7263, 1.3510, 0.8175]^{\mathrm{T}}$，目标函数为 0.8175，耗时 0.342 s。

```
>> clear P; P.objective=@(x)x(5);
   P.nonlcon=@c3mnls; P.solver='fmincon';
   P.Aineq=[-1,0,0,0,-0.25; 1,0,0,0,-0.25;
            0,-1,0,0,-0.5; 0,1,0,0,-0.5;
            0,0,-1,0,-1.5; 0,0,1,0,-1.5]; %结构体描述
   P.Bineq=[-2.25; 2.25; -1.5; 1.5; -1.5; 1.5];
   P.options=optimset; P.x0=rand(5,1);
   tic, [x,f0]=fmincon_global(P,-10,10,5,10), toc %求全局最优解
```

如果调用 100 次这个求解程序，极有可能得出原问题的全局最优解。就本例而言，测试的 100 次全得出了全局最优解，总的测试时间大约 38.7 s，平均每次 0.387 s。

```
>> tic, X=[];  %启动秒表,并设置空矩阵记录每次求解结果
   for i=1:100 %运行100次求解函数,并评价找到全局最优解的成功率
      [x,f0]=fmincon_global(P,-10,10,5,10); X=[X; x']; %记录结果
   end, toc    %显示100次求解所需的总时间
```

3.4.4　基于问题的最优化描述与求解方法

常规求解函数使用时，用户首先需要将最优化问题改写成标准形式，即只含有一个决策变量向量 $\boldsymbol{x}$、目标函数为最小值、约束条件为 $\leqslant$ 不等式的形式。对线性约束条件而言，还需要正确地提取 $\boldsymbol{A}$、$\boldsymbol{B}$、$\boldsymbol{A}_{\mathrm{eq}}$ 和 $\boldsymbol{B}_{\mathrm{eq}}$。此外，采用常规方法求解非线性规划问题时，还需要建立一些 MATLAB 函数。

除了常规的描述与求解方法之外,MATLAB还支持基于问题(problem-based)的描述方法。该方法的一般步骤为:

(1)**创建问题**。可以由下面语句创建一个空白的最优化问题prob:

```
prob=optimproblem('ObjectiveSense','max')
```

如果不给出'ObjectiveSense',则prob问题为默认的最小值问题。

(2)**声明决策变量**。调用optimvar()函数可以声明决策变量:

```
x=optimvar('x',n,m,k,'Lower',xm,'Upper',xM)
```

其中,n、m和k最多可以声明三维决策变量数组。如果不提供k,则可以定义一个$n\times m$的决策变量矩阵;如果m为1,则声明一个$n\times 1$向量。还可以定义决策变量的上下界,如果$\boldsymbol{x}_{\mathrm{m}}$为标量,则将所有决策变量的下界设置成相同的值。

(3)**描述目标函数和约束条件**。可以对问题变量的Objective和Constraints属性进行设置,以定义整个最优化问题。具体的描述方法将通过例子演示。

(4)**求解问题**。可以用sol=solve(prob,$\boldsymbol{x}_0$)命令直接求解最优化问题。对凸问题而言,初值变量$\boldsymbol{x}_0$可以略去。得出的解sol为结构体变量。

例3-21 试用基于问题的描述方法重新求解例3-7中的非线性规划问题。

解 由原始问题可见,该模型具有3个决策变量$\boldsymbol{q}$、w和k,若使用传统的模式方法,则需要将其统一成单一的决策变量向量。使用基于问题的描述方法则可以直接声明这3个决策变量,也可以直接描述线性不等式约束,没有必要提取矩阵。另外,$2.25-0.25k\leqslant q_1\leqslant 2.25+0.25k$这样的序贯不等式可以直接表示成向量,例如,使用下面语句中的c3成员。利用基于问题的描述方法,可以构造变量P,然后调用solve()函数直接求解。

```
>> P=optimproblem;          %创建最优化问题,并定义决策变量
   q=optimvar('q',3,1); w=optimvar('w',1); k=optimvar('k',1);
   P.Objective=k;           %目标函数
   P.Constraints.c1=q(3)+9.625*q(1)*w+16*q(2)*w+16*w^2+12-...
                    4*q(1)-q(2)-78*w==0;  %描述约束条件
   P.Constraints.c2=16*q(1)*w+44-19*q(1)-8*q(2)-q(3)-24*w==0;
   P.Constraints.c3=[2.25-0.25*k<=q(1), q(1)<=2.25+0.25*k];
   P.Constraints.c4=[1.5-0.5*k<=q(2), q(2)<=1.5+0.5*k];
   P.Constraints.c5=[1.5-1.5*k<=q(3), q(3)<=1.5+1.5*k];
   clear x0; x0.q=3*rand(3,1); x0.w=rand(1); x0.k=rand(1);
   sol=solve(P,x0), sol.q, sol.w, sol.k   %提取最优解
```

可见,这样的描述方式与原始的数学表达式很相似,利于求解过程的检验。

如果创建了最优化问题P,则可以由p=prob2struct(P)命令将其转换成结构体格式p。配合选定的x0成员变量,则可以由$\boldsymbol{x}$=fmincon(p)命令直接求解最优化问题。自动生成的非线性目标函数与约束条件分别存于generatedObjective.m和generatedConstraints.m文件中,无须另外创建文件。

3.4.5　最优曲线拟合方法

假设有一组数据 $x_i, y_i, i = 1, 2, \cdots, N$，且已知这组数据满足某一函数原型 $\hat{y}(x) = f(\boldsymbol{a}, x)$，其中 $\boldsymbol{a}$ 待定系数向量，则最小二乘曲线拟合的目标就是求出这一组待定系数的值，使得目标函数（即拟合的总误差）

$$J = \min_{\boldsymbol{a}} \sum_{i=1}^{N} \left[y_i - \hat{y}(x_i) \right]^2 = \min_{\boldsymbol{a}} \sum_{i=1}^{N} \left[y_i - f(\boldsymbol{a}, x_i) \right]^2 \tag{3-4-3}$$

为最小。在 MATLAB 的最优化工具箱中提供了 `lsqcurvefit()` 函数，可以解决最小二乘曲线拟合的问题，该函数的调用格式为：

$[\boldsymbol{a}, J_{\mathrm{m}}]$=`lsqcurvefit(Fun,`$\boldsymbol{a}_0$`,`$\boldsymbol{x}$`,`$\boldsymbol{y}$`,`$\boldsymbol{x}_{\mathrm{m}}$`,`$\boldsymbol{x}_{\mathrm{M}}$`,options)`

其中，$\boldsymbol{a}_0$ 为最优化的初值，$\boldsymbol{x}$、$\boldsymbol{y}$ 为原始输入输出数据向量，`Fun` 为原型函数的 MATLAB 表示，可以用匿名函数描述，也可以用 M-函数表示，该函数还允许指定待定向量的最小值 $\boldsymbol{x}_{\mathrm{m}}$ 和最大值 $\boldsymbol{x}_{\mathrm{M}}$，也可以设置搜索控制参数 `options`。调用该函数则将返回待定系数向量 $\boldsymbol{a}$，以及在此待定系数下的目标函数的值 J_{m}。

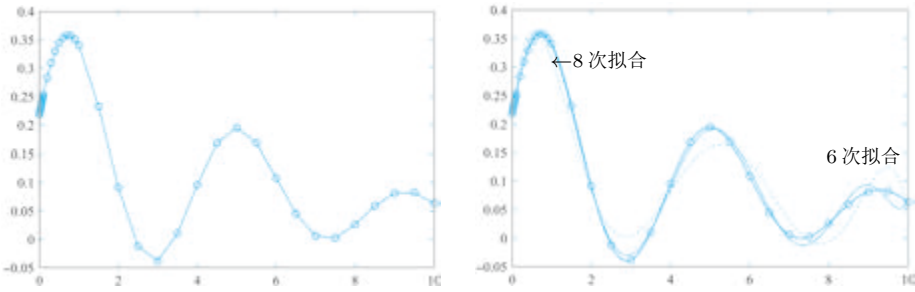
例 3-22　假设在实验中测出一组数据，且已知其可能满足的函数为

$$y(x) = a_1 \mathrm{e}^{a_2 x} \sin(a_3 x + a_4) \cos(a_5 x)$$

其中 a_i 为待定系数，试用最小二乘方法拟合该函数。

解　可以通过最小二乘拟合的方法拟合出函数的待定系数。假设通过数据生成的方法产生这组“实验数据”，下面将演示曲线的最小二乘拟合方法。首先由下面语句生成“实验数据”，这些生成的坐标点可以用二维图形绘制出来，如图 3-5(a)所示。

```
>> x=[0:0.01:0.1, 0.2:0.1:1,1.5:0.5:10]; %生成不等间距的横坐标
   y=0.56*exp(-0.2*x).*sin(0.8*x+0.4).*cos(-0.65*x); %实验数据
   plot(x,y,'o',x,y)                          %绘制实验点坐标图形
```



（a）给定的数据曲线　　（b）多项式拟合效果

图 3-5　给定数据点的曲线拟合

由原型函数可以通过最小二乘进行最优拟合，这样可以通过 MATLAB 语言编写匿名函数，然后由最小二乘法求取待定系数。

```
>> F=@(a,x)a(1)*exp(-a(2)*x).*sin(a(3)*x+a(4)).*cos(-a(5)*x);
   f=optimset; f.RelX=1e-10; f.TolFun=1e-15;  %指定较高的拟合精度
   a=lsqcurvefit(F,[1;1;1;1;1],x,y,[0,0,0,0,0],[],f) %参数拟合
   a0=[0.56;0.2;0.8;0.4;0.65]; norm(a-a0)     %和真值比较的误差
   x0=0:0.01:10; y0=F(a0,x0);                 %设置拟合点
   y1=F(a,x0); plot(x0,y0,x0,y1,'--',x,y,'o') %绘制拟合曲线
```

得出的拟合参数为 $\boldsymbol{a}=[0.56,0.2,0.8,0.4,0.65]^{\mathrm{T}}$，与给定的真值完全一致，拟合误差为 4.4177×10^{-7}，拟合结果与图3-5(a)相似，但插值结果光滑得多。

MATLAB提供的多项式拟合函数 $\boldsymbol{a}$=polyfit($\boldsymbol{x}$,$\boldsymbol{y}$,n) 也可以用于曲线拟合，其中 $\boldsymbol{x}$ 和 $\boldsymbol{y}$ 为数据向量，n 为拟合系数的阶次，通过该函数的调用将得出拟合多项式系数向量 $\boldsymbol{a}$，该向量是多项式系数按降幂排列构成的向量，前面已经介绍过，用polyval()函数可以对多项式求值。下面的语句将比较6次和8次多项式拟合的效果，如图3-5(b)所示，可见多项式拟合效果难以保证，故曲线拟合时如果已知原型时不必采用多项式拟合，若不知原型时应该选择插值。

```
>> p=polyfit(x,y,6), y2=polyval(p,x0); %6次多项式拟合结果
   p=polyfit(x,y,8); y3=polyval(p,x0); %8次多项式拟合
   plot(x0,y0,x,y,'o',x0,y2,x0,y3)     %拟合结果比较
```

其中，拟合的6次多项式为

$$P_6(x)=-0.0002x^6+0.0054x^5-0.0632x^4+0.3430x^3-0.8346x^2+0.6621x+0.2017$$

3.5 Laplace变换与z变换问题的MATLAB求解

在早期连续控制系统的研究中，常微分方程是最主要的建模工具，然而，由于微分方程相对于代数方程要复杂得多，所以应该利用一种积分变换——Laplace变换，将其映射成代数方程，从而引入了传递函数模型，该模型奠定了经典控制理论的基础，线性系统时域响应解析解方法利用了Laplace反变换的功能，而求解反变换需要复变函数中的留数方法。同样，离散系统也可以利用 z 变换构建离散传递函数模型，求解传递函数又需要 z 反变换。

3.5.1 Laplace变换

一个时域函数 $f(t)$ 的Laplace变换可以定义为

$$\mathscr{L}[f(t)]=\int_0^\infty f(t)\mathrm{e}^{-st}\mathrm{d}t=F(s) \tag{3-5-1}$$

其中 $\mathscr{L}[f(t)]$ 为Laplace变换的简单记号。

若已知函数的Laplace变换式 $F(s)$，则可以通过下面的反变换公式直接求出其Laplace反变换

$$f(t)=\mathscr{L}^{-1}[F(s)]=\frac{1}{2\pi\mathrm{j}}\int_{\sigma-\mathrm{j}\infty}^{\sigma+\mathrm{j}\infty}F(s)\mathrm{e}^{st}\mathrm{d}s \tag{3-5-2}$$

其中 σ 大于 $F(s)$ 奇异点的实部。

MATLAB 的符号运算工具箱提供的 `laplace()` 函数及 `ilaplace()` 函数直接求取给定函数的 Laplace 变换和反变换。求解积分变换问题的步骤为：

（1）**声明符号变量**。用 `syms` 命令声明符号变量。

（2）**描述原函数**。直接用 MATLAB 格式描述出原函数。

（3）**求取积分变换**。Laplace 变换和反变换可以由 `laplace()` 和 `ilaplace()` 直接变换。其实，`fourier()`、`ifourier()` 函数可以求解 Fourier 变换与反变换。

例 3-23　试求时域函数 $f(t)=1-(1+at)\mathrm{e}^{-at}$ 的 Laplace 变换。

解　下面通过计算机工具直接求取这个函数的 Laplace 变换。

```
>> syms a t                          %声明所需的变量为符号变量
   f=1-(1+a*t)*exp(-a*t);            %表示时域函数公式
   F=laplace(f), %求取函数的Laplace变换,注意得出的结果不一定最简
```

可以得出如下的结果

$$F=\frac{1}{s}-\frac{1}{s+a}-\frac{a}{(s+a)^2}$$

利用 `ilaplace()` 对上述结果进行 Laplace 反变换，则可以还原成原来的时域函数，可以采用下面命令来完成这样的反变换。由 `ilaplace(F)` 可以得出反变换的结果为 $1-\mathrm{e}^{-at}-at\mathrm{e}^{-at}$。

例 3-24　已知某 Laplace 表达式如下，试求 Laplace 反变换。

$$G(s)=\frac{s+3}{s(s^4+2s^3+11s^2+18s+18)}$$

解　用下面的 MATLAB 语句就可以求出信号 Laplace 反变换的解析解。

```
>> syms s, G=(s+3)/s/(s^4+2*s^3+11*s^2+18*s+18); y=ilaplace(G)
```

可以得出解析解为

$$y(t)=\frac{1}{255}\cos 3t-\frac{13}{255}\sin 3t-\frac{29}{170}\mathrm{e}^{-t}\cos t-\frac{3}{170}\mathrm{e}^{-t}\sin t+\frac{1}{6}$$

例 3-25　求出 Laplace 变换式 $\mathscr{L}^{-1}\left[\dfrac{3a^2}{s^3+a^3}\right]$，$a>0$。

解　如果想证明该式子，可以首先对给出的式子进行 Laplace 反变换。

```
>> syms s t; syms a positive; F=3*a^2/(s^3+a^3);
   f=simplify(ilaplace(F))
```

可以求出 $\mathscr{L}^{-1}\left[\dfrac{3a^2}{s^3+a^3}\right]=\mathrm{e}^{-at}+\mathrm{e}^{at/2}\left(-\cos\dfrac{\sqrt{3}}{2}at+\sqrt{3}\sin\dfrac{\sqrt{3}}{2}at\right)$。

例 3-26　试求 Laplace 函数 $F(s)=\dfrac{\mathrm{e}^{-\sqrt{s}}}{\sqrt{s}(\sqrt{s}-1)}$ 的反变换。

解　反变换可以通过下面函数直接求出

```
>> syms s; z=sqrt(s); f=ilaplace(exp(-z)/z/(z-1))
```

在新版本下这个问题是不能求解的,如果使用早期版本,例如,MATLAB 2008a及更早的版本,则得出的结果为 $f(t)=\mathrm{e}^{t-1}\mathrm{erfc}\big(-(2t-1)/(2\sqrt{t})\big)$,原函数是分数阶导数的一个例子。

3.5.2 数值Laplace变换

前面给出了Laplace变换的求解函数`laplace()`,该函数可以推导出很多时域函数的Laplace变换的解析解,同时也有很多函数Laplace变换的解析解不存在或不适合用解析解方法求解,所以应该考虑数值方法求解Laplace变换问题。

Juraj Valsa开发了基于数值方法的Laplace反变换的函数`INVLAP()`[10,11],该函数的调用格式为`[t,y]=INVLAP(f,t0,tn,N,其他参数)`,其中,原函数由含有字符s的字符串表示,(t_0,t_n)为感兴趣的区间且$t_0\neq 0$,N为用户选择的计算点数,用户可以选择不同的N值来检验运算的结果。"其他参数"的选取可以参考原函数的联机帮助,不过这里建议:除非特别需要没有必要去人为修改这些默认参数。

文献[1]对`INVLAP()`函数进行了扩展,给出了基于数值Laplace变换的反馈控制系统输出响应求解函数`INVLAP_new()`,其调用格式如下:

```
[t,y]=INVLAP_new(G,t0,tn,N)              %G的Laplace反变换
[t,y]=INVLAP_new(G,t0,tn,N,H)            %G,H闭环系统的冲激响应
[t,y]=INVLAP_new(G,t0,tn,N,H,u)          %u用于描述输入信号
[t,y]=INVLAP_new(G,t0,tn,N,H,tx,ux)      %tx,ux为时间、输入采样点
```

该函数支持多种调用格式,其中,G为Laplace变换表达式的字符串,如果同时提供了H,则G为前向通路的传递函数模型字符串,H为负反馈回路传递函数的字符串;如果需要描述输入信号,则u可以为输入信号的Laplace变换字符串,或输入时域信号的匿名函数句柄;输入信号还可以由采样点$(\boldsymbol{t}_x,\boldsymbol{u}_x)$表示;如果只考虑$G$模型的响应,可以将$H$设置为0。

例3-27 假设复杂开环无理模型如下[12],试绘制单位负反馈统的阶跃响应曲线。

$$G(s)=\left[\frac{\sinh(0.1\sqrt{s})}{0.1\sqrt{s}}\right]^2\frac{1}{\sqrt{s}\sinh(\sqrt{s})}$$

解 开环无理传递函数可以由字符串表示,这样由下面语句直接绘制出系统的闭环阶跃响应曲线,如图3-6所示。

```
>> G='(sinh(0.1*sqrt(s))/0.1/sqrt(s))^2/sqrt(s)/sinh(sqrt(s))';
   [t,y]=INVLAP_new(G,0,10,1000,1,'1/s');
   plot(t,y)      %闭环系统阶跃响应
```

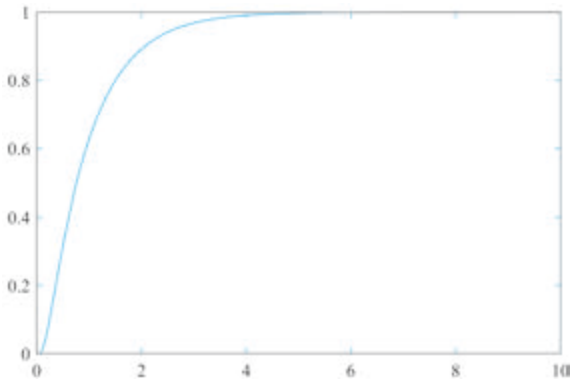


图 3-6　闭环系统的阶跃响应曲线

3.5.3　z 变换

离散序列信号 $f(n)$ 的 z 变换可以定义为

$$\mathscr{Z}[f(n)] = \sum_{i=0}^{\infty} f(n)z^{-n} = F(z) \tag{3-5-3}$$

给定 z 变换式子 $F(z)$，则其 z 反变换的数学表示为

$$f(n) = \mathscr{Z}^{-1}[F(z)] = \frac{1}{2\pi \mathrm{j}} \oint F(z)z^{n-1}\mathrm{d}z \tag{3-5-4}$$

可以调用 MATLAB 符号运算工具箱中的 `ztrans()` 函数和 `iztrans()` 函数对给定的函数进行 z 变换和反变换。

例 3-28　一般介绍 z 变换的书中不介绍 $q/(z^{-1}-p)^m\,(p \neq 0)$ 函数的 z 反变换，而该函数是求取有重根离散系统解析解的基础。试推导该函数的 z 反变换一般表达式。

解　如果 m 选作符号变量，则 `ztrans()` 函数并不能得出有益的结果。这里尝试对不同的 m 值进行反变换，并总结出一般规律。根据要求，可以用符号运算工具箱求出 $m = 1, 2, \cdots, 5$ 的 z 反变换。

```
>> syms p q z n; assume(p~=0); assume(n~=0)
   for i=1:5 %尝试不同阶次,以便总结规律
      F=iztrans(q/(1/z-p)^i); %求反变换
      F=subs(F,{nchoosek(n-1,2),nchoosek(n-1,3),nchoosek(n-1,4)},...
         {(n-1)*(n-2)/2,(n-1)*(n-2)*(n-3)/6,...
         (n-1)*(n-2)*(n-3)*(n-4)/24});
      F=prod(factor(F)) %对得出的结果直接作因式分解
   end
```

上述语句可以直接得出如下结果。

$$F_1 = -\frac{q}{p^{1+n}},\quad F_2 = \frac{q}{p^{2+n}}(1+n),\quad F_3 = -\frac{q}{2p^{3+n}}(1+n)(2+n)$$

$$F_4=\frac{q}{6p^{4+n}}(3+n)(2+n)(1+n),\quad F_5=-\frac{q}{24p^{5+n}}(4+n)(3+n)(2+n)(1+n)$$

总结上述结果的规律,可以写出一般的z反变换结果

$$\mathscr{Z}^{-1}\left[\frac{q}{(z^{-1}-p)^m}\right]=\frac{(-1)^m q}{(m-1)!\,p^{n+m}}(n+1)(n+2)\cdots(n+m-1)$$

注意,求解这个问题在新版本下使用的语句比较烦琐,因为,默认得出的结果含有nchoosek()函数,即组合函数,所以,需要手工替换成相应的多项式形式。此外,直接采用simplify()函数对结果化简,格式可能不统一,所以这里采用因式分解的方法化简结果。早期版本中,对本问题只需给出命令simplify()即可。

例3-29 已知某信号的z变换表达式如下,试求其z反变换。

$$H(z)=\frac{z(5z-2)}{(z-1)(z-1/2)^3(z-1/3)}$$

解 可以通过下面的MATLAB的符号运算工具箱直接求取该信号的z反变换。

```
>> syms z; H=z*(5*z-2)/((z-1)*(z-1/2)^3*(z-1/3)); iztrans(H)
```

得出的结果为

$$\mathscr{Z}^{-1}[H(z)]=36+(72-60n-12n^2)\left(\frac{1}{2}\right)^n-108\left(\frac{1}{3}\right)^n$$

3.6 习题

(1) 对下面给出的各个矩阵求取各种参数,如矩阵的行列式、迹、秩、特征多项式、范数等,试分别求出它们的解析解。

$$\boldsymbol{A}=\begin{bmatrix}7.5 & 3.5 & 0 & 0\\ 8 & 33 & 4.1 & 0\\ 0 & 9 & 103 & -1.5\\ 0 & 0 & 3.7 & 19.3\end{bmatrix},\quad \boldsymbol{B}=\begin{bmatrix}5 & 7 & 6 & 5\\ 7 & 10 & 8 & 7\\ 6 & 8 & 10 & 9\\ 5 & 7 & 9 & 10\end{bmatrix}$$

(2) 求出下面给出的矩阵的秩和Moore–Penrose广义逆矩阵,并验证它们是否满足Moore–Penrose逆矩阵的条件。

$$\boldsymbol{A}=\begin{bmatrix}2 & 2 & 3 & 1\\ 2 & 2 & 3 & 1\\ 4 & 4 & 6 & 2\\ 1 & 1 & 1 & 1\\ -1 & -1 & -1 & 3\end{bmatrix},\quad \boldsymbol{B}=\begin{bmatrix}4 & 1 & 2 & 0\\ 1 & 1 & 5 & 15\\ 3 & 1 & 3 & 5\end{bmatrix}$$

(3) 试求解下面的线性代数方程组,并检验解的正确性。

$$\boldsymbol{X}\begin{bmatrix}7 & 6 & 9 & 7\\ 7 & 1 & 3 & 2\\ 2 & 1 & 5 & 5\\ 6 & 4 & 2 & 6\end{bmatrix}=\begin{bmatrix}2 & 1 & 0 & 1\\ 0 & 3 & 1 & 2\end{bmatrix}$$

（4）给定下面特殊矩阵 $\boldsymbol{A}$，试利用符号运算工具箱求出其逆矩阵、特征值，并求出状态转移矩阵 $\mathrm{e}^{\boldsymbol{A}t}$ 的解析解。

$$\boldsymbol{A}=\begin{bmatrix}-4 & -11 & -2 & 4 & -2\\ -1 & -5 & -1 & 0 & -1\\ 2 & 6 & -1 & -2 & 2\\ -1 & -4 & -1 & -1 & -1\\ 2 & 12 & 2 & -5 & 0\end{bmatrix}$$

（5）试求下面齐次方程的基础解系。

$$\begin{cases}6x_1+x_2+4x_3-7x_4-3x_5=0\\ -2x_1-7x_2-8x_3+6x_4=0\\ -4x_1+5x_2+x_3-6x_4+8x_5=0\\ -34x_1+36x_2+9x_3-21x_4+49x_5=0\\ -26x_1-12x_2-27x_3+27x_4+17x_5=0\end{cases}$$

（6）求解下面的 Lyapunov 方程，并检验所得解的精度。

$$\begin{bmatrix}1 & 2 & 3\\ 4 & 5 & 6\\ 7 & 8 & 0\end{bmatrix}\boldsymbol{X}+\boldsymbol{X}\begin{bmatrix}2 & 3 & 6\\ 3 & 5 & 2\\ 3 & 2 & 2\end{bmatrix}=\begin{bmatrix}1 & 3 & 2\\ 3 & 4 & 1\\ 5 & 2 & 1\end{bmatrix}$$

（7）试用数值方法和解析方法求取下面的 Sylvester 方程，并验证得出的结果。

$$\begin{bmatrix}3 & -6 & -4 & 0 & 5\\ 1 & 4 & 2 & -2 & 4\\ -6 & 3 & -6 & 7 & 3\\ -13 & 10 & 0 & -11 & 0\\ 0 & 4 & 0 & 3 & 4\end{bmatrix}\boldsymbol{X}+\boldsymbol{X}\begin{bmatrix}3 & -2 & 1\\ -2 & -9 & 2\\ -2 & -1 & 9\end{bmatrix}=\begin{bmatrix}-2 & 1 & -1\\ 4 & 1 & 2\\ 5 & -6 & 1\\ 6 & -4 & -4\\ -6 & 6 & -3\end{bmatrix}$$

（8）试求出下面的代数方程的全部的根

$$\begin{cases}x^2y^2-zxy-4x^2yz^2=xz^2\\ xy^3-2yz^2=3x^3z^2+4xzy^2\\ y^2x-7xy^2+3xz^2=x^4zy\end{cases}$$

（9）采用适当的方法求解下面的非线性方程 [13]

① $\begin{cases}xyz=1\\ x^2+2y^2+4z^2=7\\ 2x^2+y^3+6z=7\end{cases}$　② $\begin{cases}x^2+2\sin(y\pi/2)+z^2=0\\ -2xy+z=3\\ x^2z-y=7\end{cases}$

（10）试找出下面非线性矩阵方程所有可能的解

$$\begin{bmatrix}9 & 1 & 0\\ 8 & 7 & 3\\ 3 & 0 & 6\end{bmatrix}\boldsymbol{X}^2+\boldsymbol{X}\begin{bmatrix}5 & 6 & 6\\ 9 & 2 & 2\\ 6 & 9 & 5\end{bmatrix}\boldsymbol{X}+\boldsymbol{X}\begin{bmatrix}7 & 9 & 4\\ 6 & 7 & 1\\ 4 & 6 & 4\end{bmatrix}+\boldsymbol{I}_{3\times3}=\boldsymbol{0}$$

（11）若 $\boldsymbol{A}$、$\boldsymbol{B}$、$\boldsymbol{C}$ 和 $\boldsymbol{D}$ 矩阵由例 3-13 给出，试求解矩阵方程

$$\boldsymbol{X}^3+\boldsymbol{X}^4\boldsymbol{D}-\boldsymbol{X}^2\boldsymbol{B}\boldsymbol{X}+\boldsymbol{C}\boldsymbol{X}-\boldsymbol{I}=\boldsymbol{0}$$

假设已经求出了方程的 77 个实根，总共 3351 个复数根，在 `data3ex1.mat` 文件给出，试接着求解该方程，看看能不能找到新的解。

(12) 试求出伪多项式方程 $x^{\sqrt{7}}+2x^{\sqrt{3}}+3x^{\sqrt{2}-1}+4=0$ 所有的根,并检验结果。

(13) 某 Riccati 方程的数学表达式为 $\boldsymbol{PA}+\boldsymbol{A}^{\mathrm{T}}\boldsymbol{P}-\boldsymbol{PBR}^{-1}\boldsymbol{B}^{\mathrm{T}}\boldsymbol{P}+\boldsymbol{Q}=\boldsymbol{0}$,且已知

$$\boldsymbol{A}=\begin{bmatrix}-27 & 6 & -3 & 9\\ 2 & -6 & -2 & -6\\ -5 & 0 & -5 & -2\\ 10 & 3 & 4 & -11\end{bmatrix},\quad \boldsymbol{B}=\begin{bmatrix}0 & 3\\ 16 & 4\\ -7 & 4\\ 9 & 6\end{bmatrix}$$

$$\boldsymbol{Q}=\begin{bmatrix}6 & 5 & 3 & 4\\ 5 & 6 & 3 & 4\\ 3 & 3 & 6 & 2\\ 4 & 4 & 2 & 6\end{bmatrix},\quad \boldsymbol{R}=\begin{bmatrix}4 & 1\\ 1 & 5\end{bmatrix}$$

试求解该方程,得出 $\boldsymbol{P}$ 矩阵,并检验得出解的精度。试求出并验证方程全部的根。

(14) 试求解非线性矩阵方程 $\boldsymbol{AX}\sin(\boldsymbol{B}^2\boldsymbol{X}+\boldsymbol{C})\boldsymbol{X}+\boldsymbol{X}\mathrm{e}^{-\boldsymbol{B}}+\boldsymbol{C}=\boldsymbol{0}$,其中

$$\boldsymbol{A}=\begin{bmatrix}4 & 4 & 4\\ 0 & 3 & 9\\ 4 & 7 & 9\end{bmatrix},\quad \boldsymbol{B}=\begin{bmatrix}1 & 1 & 7\\ 3 & 5 & 5\\ 8 & 1 & 0\end{bmatrix},\quad \boldsymbol{C}=\begin{bmatrix}6 & 4 & 7\\ 1 & 9 & 4\\ 1 & 3 & 0\end{bmatrix}$$

(15) Lorenz 方程是研究混沌问题的著名的非线性微分方程,其数学形式为

$$\begin{cases}\dot{x}_1(t)=-\beta x_1(t)+x_2(t)x_3(t)\\ \dot{x}_2(t)=-\sigma x_2(t)+\sigma x_3(t)\\ \dot{x}_3(t)=-x_1(t)x_2(t)+\gamma x_2(t)-x_3(t)\end{cases}$$

其中,$\beta=8/3,\sigma=10,\gamma=28$,且其初值为 $x_1(0)=x_2(0)=0,x_3(0)=10^{-3}$。试求出其数值解,绘制三维空间的相轨迹,并绘制出 Lorenz 方程解在两两平面上的投影。

(16) 请给出求解下面微分方程的 MATLAB 命令

$$\dddot{y}+ty\ddot{y}+t^2\dot{y}y^2=\mathrm{e}^{-ty},\quad y(0)=2,\quad \dot{y}(0)=\ddot{y}(0)=0$$

并绘制出 $y(t)$ 曲线,试问该方程存在解析解吗?

(17) Lotka–Volterra 扑食模型方程如下,且初值为 $x(0)=2,\ y(0)=3$,试求解该微分方程,并绘制相应的曲线。

$$\begin{cases}\dot{x}(t)=4x(t)-2x(t)y(t)\\ \dot{y}(t)=x(t)y(t)-3y(t)\end{cases}$$

(18) 试求解下面的零初值微分方程

① $\begin{cases}\dot{x}(t)=\sqrt{x^2(t)-y(t)+3}-3\\ \dot{y}(t)=\arctan(x^2(t)+2x(t)y(t))\end{cases}$ ② $\begin{cases}\dot{x}(t)=\ln(2-y(t)+2y^2(t))\\ \dot{y}(t)=4-\sqrt{x(t)+4x^2(t)}\end{cases}$

(19) 试求解边值问题 $\ddot{y}(x)=\lambda^2(y^2(x)+\cos^2\pi x)+2\pi^2\cos 2\pi x$,其中,$y(0)=y(1)=0$,$\dot{y}(0)=1$。提示:这个问题不是真正的边值问题,可以考虑由 $y(1)=0$ 求出未知参数 λ,再求解微分方程。试学习与使用 `bvp5c()` 函数直接求解这个边值问题。

（20）试求出下面微分方程组的解析解，并和数值解比较。

① $\begin{cases} \ddot{x}(t)=-2x(t)-3\dot{x}(t)+\mathrm{e}^{-5t}, & x(0)=1,\ \dot{x}(0)=2 \\ \ddot{y}(t)=2x(t)-3y(t)-4\dot{x}(t)-4\dot{y}(t)-\sin t, & y(0)=3,\ \dot{y}(0)=4 \end{cases}$

② $\begin{cases} \ddot{x}(t)+\ddot{y}(t)+x(t)+y(t)=0, & x(0)=2,\ y(0)=1 \\ 2\ddot{x}(t)-\ddot{y}(t)-x(t)+y(t)=\sin t, & \dot{x}(0)=\dot{y}(0)=-1 \end{cases}$

（21）一级倒立摆模型的数学描述为

$$\begin{cases} \ddot{x}=\dfrac{u+ml\sin\theta\dot{\theta}^2-mg\cos\theta\sin\theta}{M+m-m\cos^2\theta} \\ \ddot{\theta}=\dfrac{u\cos\theta-(M+m)g\sin\theta+ml\sin\theta\cos\theta\dot{\theta}}{ml\cos^2\theta-(M+m)l} \end{cases}$$

已知 $m=M=0.5\,\mathrm{kg}, l=0.3\,\mathrm{m}, g=9.81\,\mathrm{m/s^2}$。试求解该系统在单位阶跃信号 u 作用下的零初始状态时间响应（注意：该系统为自然不稳定系统，如果需要使之稳定则应使用特殊的控制方法）。

（22）假设方程为零初值问题，试求解下面的常微分方程

$$\begin{cases} \cos\ddot{x}(t)\,\ddot{y}(t)-\cos\ddot{x}(t)-\ddot{y}(t)-x(t)\dot{y}(t)+\mathrm{e}^{-x(t)}y(t)=2 \\ \sin\ddot{x}(t)\cos\ddot{y}(t)-x(t)\dot{y}(t)+\ddot{x}(t)y(t)-y^2(t)\dot{y}(t)=5 \end{cases}$$

（23）求解下面的最优化问题。

① $\min\left(x_1^2-2x_1+x_2\right)$

$$\boldsymbol{x}\ \text{s.t.}\begin{cases} 4x_1^2+x_2^2\leqslant 4 \\ x_1,x_2\geqslant 0 \end{cases}$$

② $\max\left[-(x_1-1)^2-(x_2-1)^2\right]$

$$\boldsymbol{x}\ \text{s.t.}\ x_1+x_2+5=0$$

（24）考虑下面二元最优化问题的求解，还可以用图解方法验证你得出的解。

$$\max\ (-x_1^2-x_2)$$
$$\boldsymbol{x}\ \text{s.t.}\begin{cases} 9\geqslant x_1^2+x_2^2 \\ x_1+x_2\leqslant 1 \end{cases}$$

（25）试求解下面的非线性规划问题。

$$\min\ \frac{1}{2\cos x_6}\left[x_1x_2(1+x_5)+x_3x_4\left(1+\frac{31.5}{x_5}\right)\right]$$
$$\boldsymbol{x}\ \text{s.t.}\begin{cases} 0.003079x_1^3x_2^3x_5-\cos^3 x_6\geqslant 0 \\ 0.1017x_3^3x_4^3-x_5^2\cos^3 x_6\geqslant 0 \\ 0.09939(1+x_5)x_1^3x_2^2-\cos^2 x_6\geqslant 0 \\ 0.1076(31.5+x_5)x_3^3x_4^2-x_5^2\cos^2 x_6\geqslant 0 \\ x_3x_4(x_5+31.5)-x_5[2(x_1+5)\cos x_6+x_1x_2x_5]\geqslant 0 \\ 0.2\leqslant x_1\leqslant 0.5, 14\leqslant x_2\leqslant 22, 0.35\leqslant x_3\leqslant 0.6 \\ 16\leqslant x_4\leqslant 22, 5.8\leqslant x_5\leqslant 6.5, 0.14\leqslant x_6\leqslant 0.2618 \end{cases}$$

（26）试求解下面的最优化问题 [14]。

$$\min\ k$$
$$\boldsymbol{q},w,k\ \ \text{s.t.}\begin{cases} q_3+9.625q_1w+16q_2w+16w^2+12-4q_1-q_2-78w=0 \\ 16q_1w+44-19q_1-8q_2-q_3-24w=0 \\ 2.25-0.25k\leqslant q_1\leqslant 2.25+0.25k \\ 1.5-0.5k\leqslant q_2\leqslant 1.5+0.5k \\ 1.5-1.5k\leqslant q_3\leqslant 1.5+1.5k \end{cases}$$

(27) 试求解下面的最优化问题。

$$\min \quad 0.6224x_1x_2x_3x_4+1.7781x_2x_3^2+3.1661x_1^2x_4+19.84x_1^2x_3$$

$$\boldsymbol{x}\ \text{s.t.}\begin{cases}0.0193x_3-x_1\leqslant 0\\ 0.00954x_3-x_2\leqslant 0\\ 750\times 1728-\pi x_3^2x_4-4\pi x_3^3/3\leqslant 0\\ x_4-240\leqslant 0\\ 0.0625\leqslant x_1,x_2\leqslant 6.1875,10\leqslant x_3,x_4\leqslant 200\end{cases}$$

(28) 试用一般的有约束最优化方法求解下面的线性规划问题，试学习 `linprog()` 函数的使用方法重新求解下面的问题。

① $\min \quad -3x_1+4x_2-2x_3+5x_4$

$$\boldsymbol{x}\ \text{s.t.}\begin{cases}4x_1-x_2+2x_3-x_4=-2\\ x_1+x_2-x_3+2x_4\leqslant 14\\ 2x_1-3x_2-x_3-x_4\geqslant -2\\ x_{1,2,3}\geqslant -1, x_4\text{无约束}\end{cases}$$

② $\min \quad x_6+x_7$

$$\boldsymbol{x}\ \text{s.t.}\begin{cases}x_1+x_2+x_3+x_4=4\\ -2x_1+x_2-x_3-x_6+x_7=1\\ 3x_2+x_3+x_5+x_7=9\\ x_{1,2,\cdots,7}\geqslant 0\end{cases}$$

(29) 考虑一个简单的一元函数最优化问题求解，$f(x)=x\sin 10\pi x+2, x\in(-1,2)$，试求出 $f(x)$ 取最大值时 x 的值。已知，该函数曲线有很强振荡，所以采用常规最优化方法时，若初值选择不当往往会得出局部最小值。要求在本题求解中，在 $x\in(-1,2)$ 区间内随机选择 40 个初始点，按照图 3-7 中给出的流程编程，用循环的方式从每个初始点出发进行搜索，得出全局最优解。如有可能，将该方法和函数扩展成求取多元函数 $y=f(\boldsymbol{x})$ 全局最优解的通用程序。

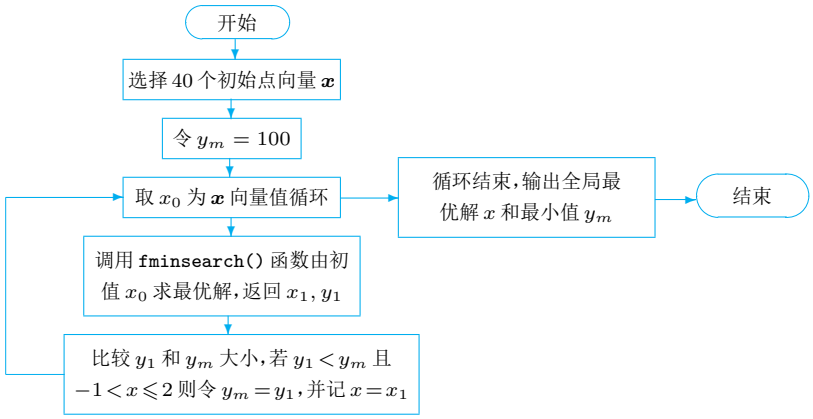


图 3-7　随机初值的全局最优化求解框图

(30) 假设有一组实测数据由表 3-1 给出，且已知该数据可能满足的原型函数为 $y(x)=ax+bx^2\mathrm{e}^{-cx}+d$，试求出满足下面数据的最小二乘解 a,b,c,d 的值。

表 3-1　习题(30)实测数据

x_i	0.1	0.2	0.3	0.4	0.5	0.6	0.7	0.8	0.9	1
y_i	2.3201	2.6470	2.9707	3.2885	3.6008	3.9090	4.2147	4.5191	4.8232	5.1275

（31）假设某日气温的实测值由表 3-2 给出，试用各种方法对之进行平滑插值，并得出 3 次、4 次插值多项式，并用曲线绘制的方法观察拟合效果。如果想获得很好的拟合效果，至少应该用多少阶多项式去拟合。

表 3-2　习题（31）实测数据

时间	1	2	3	4	5	6	7	8	9	10	11	12
温度	14	14	14	14	15	16	18	20	22	23	25	28
时间	13	14	15	16	17	18	19	20	21	22	23	24
温度	31	32	31	29	27	25	24	22	20	18	17	16

（32）已知某连续系统的阶跃响应数据由表 3-3 给出，且已知系统为二阶系统，其阶跃响应的曲线原型为 $y(t) = x_1 + x_2\mathrm{e}^{-x_4 t} + x_3\mathrm{e}^{-x_5 t}$，试用曲线最小二乘拟合算法拟合出 x_i 参数，从而拟合出系统的传递函数模型。

表 3-3　习题（32）实测数据

t	$y(t)$	t	$y(t)$	t	$y(t)$	t	$y(t)$	t	$y(t)$	t	$y(t)$
0	0	1.6	0.2822	3.2	0.3024	4.8	0.3145	6.4	0.3218	8	0.3263
0.1	0.08324	1.7	0.2839	3.3	0.3034	4.9	0.315	6.5	0.3222	8.1	0.3265
0.2	0.1404	1.8	0.2855	3.4	0.3043	5	0.3156	6.6	0.3225	8.2	0.3267
0.3	0.1798	1.9	0.287	3.5	0.3051	5.1	0.3161	6.7	0.3228	8.3	0.3269
0.4	0.2072	2	0.2885	3.6	0.306	5.2	0.3166	6.8	0.3231	8.4	0.3271
0.5	0.2265	2.1	0.2899	3.7	0.3068	5.3	0.3172	6.9	0.3235	8.5	0.3273
0.6	0.2402	2.2	0.2912	3.8	0.3076	5.4	0.3176	7	0.3238	8.6	0.3275
0.7	0.2501	2.3	0.2925	3.9	0.3084	5.5	0.3181	7.1	0.324	8.7	0.3277
0.8	0.2574	2.4	0.2937	4	0.3092	5.6	0.3186	7.2	0.3243	8.8	0.3278
0.9	0.2629	2.5	0.2949	4.1	0.3099	5.7	0.319	7.3	0.3246	8.9	0.328
1	0.2673	2.6	0.2961	4.2	0.3106	5.8	0.3195	7.4	0.3249	9	0.3282
1.1	0.2708	2.7	0.2973	4.3	0.3113	5.9	0.3199	7.5	0.3251	9.1	0.3283
1.2	0.2737	2.8	0.2983	4.4	0.312	6	0.3203	7.6	0.3254	9.2	0.3285
1.3	0.2762	2.9	0.2994	4.5	0.3126	6.1	0.3207	7.7	0.3256	9.3	0.3286
1.4	0.2784	3	0.3004	4.6	0.3133	6.2	0.3211	7.8	0.3258	9.4	0.3288
1.5	0.2804	3.1	0.3014	4.7	0.3139	6.3	0.3214	7.9	0.3261	9.5	0.3289

（33）神经网络是拟合曲线的一种有效方法，虽然本书未详细介绍神经网络理论，但可以试用 MATLAB 神经网络工具箱带的现成程序 `nntool`，由给出的程序界面对上述的数据进行曲线拟合，并与多项式拟合的结果进行比较。

（34）对下列函数 $f(t)$ 进行 Laplace 变换

① $f_1(t) = \dfrac{\sin \alpha t}{t}$　② $f_2(t) = t^5 \sin \alpha t$　③ $f_3(t) = t^8 \cos \alpha t$　④ $f_4(t) = t^6 \mathrm{e}^{\alpha t}$

⑤ $f_e(t) = \dfrac{\cos \alpha t}{t}$　⑥ $f_f(t) = \mathrm{e}^{\beta t} \sin(\alpha t + \theta)$　⑦ $f_g(t) = \mathrm{e}^{-12t} + 6\mathrm{e}^{9t}$

（35）对上面的结果作 Laplace 反变换，看看能不能还原给定的函数。

(36) 对下面的 $F(s)$ 式进行 Laplace 反变换,并对得出的结果作 Laplace 变换,看看能否还原原函数。

① $F_1(s)=\dfrac{1}{\sqrt{s}(s^2-a^2)(\sqrt{s}+b)}$ ② $F_2(s)=\sqrt{s-a}-\sqrt{s-b}$,

③ $F_3(s)=\ln\dfrac{s-a}{s-b}$ ④ $F_4(s)=\dfrac{s-a}{\sqrt{s}(s^2-a^2)(\sqrt{s}+b)}$ ⑤ $F_5(s)=\dfrac{3a^2}{s^3+a^3}$,

⑥ $F_6(s)=\dfrac{(s-1)^8}{s^7}$ ⑦ $F_7(s)=\ln\dfrac{s^2+a^2}{s^2+b^2}$,

⑧ $F_h(s)=\dfrac{s^2+3s+8}{\prod\limits_{i=1}^{8}(s+i)}$ ⑨ $F_i(s)=\dfrac{1}{2}\dfrac{s+\alpha}{s-\alpha}$

(37) 假设某分数阶系统是由两个子模型 $G_1(s)$ 和 $G_2(s)$ 并联而成,则系统的总模型可以由 $G(s)=G_1(s)+G_2(s)$ 计算出来。试对下面的两个子模型并联的总系统绘制出阶跃响应曲线。如果 $G_1(s)$、$G_2(s)$ 串联连接,试求出总模型的阶跃响应。

$$G_1(s)=\frac{(s^{0.4}+2)^{0.8}}{\sqrt{s}(s^2+3s^{0.9}+4)^{0.3}},\quad G_2(s)=\frac{s^{0.4}+0.6s+3}{(s^{0.5}+3s^{0.4}+5)^{0.7}}$$

(38) 已知下述各个 z 变换表达式 $F(z)$,试对它们分别进行 z 反变换,并对得出的结果作 z 变换,看看能否还原原函数。

① $F_1(z)=\dfrac{10z}{(z-1)(z-2)}$ ② $F_2(z)=\dfrac{z^2}{(z-0.8)(z-0.1)}$,

③ $F_3(z)=\dfrac{z}{(z-a)(z-1)^2}$ ④ $F_4(z)=\dfrac{z^{-1}(1-\mathrm{e}^{-aT})}{(1-z^{-1})(1-z^{-1}\mathrm{e}^{-aT})}$,

⑤ $F_e(z)=\dfrac{Az[z\cos\beta-\cos(\alpha T-\beta)]}{z^2-2z\cos\alpha T+1}$

(39) 已知某信号的 Laplace 变换为 $\dfrac{b}{s^2(s+a)}$,试求其 z 变换,并验证结果。

(40) 用计算机证明

$$\mathscr{Z}\left\{1-\mathrm{e}^{-akT}\left[\cos bkT+\frac{a}{b}\sin bkT\right]\right\}=\frac{z(Az+B)}{(z-1)(z^2-2\mathrm{e}^{-aT}\cos bTz+\mathrm{e}^{-2aT})}$$

式中

$$A=1-\mathrm{e}^{-aT}\cos bT-\frac{a}{b}\mathrm{e}^{-aT}\sin bT,B=\mathrm{e}^{-2aT}+\frac{a}{b}\mathrm{e}^{-aT}\sin bT-\mathrm{e}^{-aT}\cos bT$$

参考文献

[1] 薛定宇. 高等应用数学问题的 MATLAB 求解 [M]. 4 版. 北京:清华大学出版社,2018.

[2] 薛定宇. 薛定宇教授大讲堂(卷Ⅱ):MATLAB 微积分运算 [M]. 北京:清华大学出版社,2019.

[3] 薛定宇. 薛定宇教授大讲堂(卷Ⅲ):MATLAB 线性代数运算 [M]. 北京:清华大学出版社,2019.

[4] 薛定宇. 薛定宇教授大讲堂（卷 IV）：MATLAB 最优化计算 [M]. 北京：清华大学出版社，2020.

[5] 薛定宇. 薛定宇教授大讲堂（卷 V）：MATLAB 微分方程求解 [M]. 北京：清华大学出版社，2020.

[6] 薛定宇，陈阳泉. 基于 MATLAB/Simulink 的系统仿真技术与应用 [M]. 2 版. 北京：清华大学出版社，2011.

[7] Forsythe G E，Malcolm M A，Moler C B. Computer methods for mathematical computations [M]. Englewood Cliffs：Prentice-Hall，1977.

[8] Nelder J A，Mead R. A simplex method for function minimization [J]. Computer Journal，1965，7（4）：308–313.

[9] D'Errico J. Bound constrained optimization fminsearchbnd [OL]，2005. http://www.mathworks.cn/matlabcentral/fileexchange/8277-fminsearchbnd.

[10] Valsa J，Brančik L. Approximate formulae for numerical inversion of Laplace transforms [J]. International Journal of Numerical Modelling：Electronic Networks，Devices and Fields，1998，11（3）：153–166.

[11] Valsa J. Numerical inversion of Laplace transforms in MATLAB [R/OL]. MATLAB Central. https://ww2.mathworks.cn/matlabcentral/fileexchange/32824-numerical-inversion-of-laplace-transforms-in-matlab，2011.

[12] Callier F M，Winkin J. Infinite dimensional system transfer functions [M]// Curtain R F，Bensoussan A and Lions J L eds. Analysis and optimization of systems：State and frequency domain approaches for infinite-dimensional systems. Berlin：Springer-Verlag，1993：72–101.

[13] Yang W Y，Cao W，Chung T S，et al. Applied numerical methods using MATLAB [M]. Hoboken：John Wiley & Sons，Inc.，2005.

[14] Henrion D. A review of the global optimization toolbox for Maple [R/OL]. https://homepages.laas.fr/henrion/Papers/mapleglobopt.pdf，2006.