第1章　系统仿真技术与应用

1.1　系统仿真技术概述

系统是由客观世界中实体与实体间的相互作用和相互依赖关系构成的具有某种特定功能的有机整体。系统的分类方法是多种多样的，习惯上依照其应用范围可以分为工程系统和非工程系统。工程系统是指由相互关联的部件组成一个整体，实现特定的目标，例如，电机驱动自动控制系统是由执行部件、功率转换部件和检测部件所组成，用来完成电机的转速、位置和其他参数控制的某个特定目标。

非工程系统涵盖的范围更加广泛，大至宇宙，小至微观世界都存在着相互关联、相互制约的关系，形成一个整体，实现某种目的，所以均可以认为是系统。

如果想定量地研究系统的行为，可以将其本身的特性及内部的相互关系抽象出来，构造出系统的模型。系统的模型分为物理模型和数学模型。由于计算机技术的迅速发展和广泛应用，数学模型的应用越来越普遍。

系统的数学模型是描述系统动态特性的数学表达式，用来表示系统运动过程中各个量的关系，是分析、设计系统的依据。根据数学模型所描述的系统的运动性质和数学工具来划分，又可以分为连续系统、离散时间系统、离散事件系统和混杂系统等；还可以细分为线性、非线性、定常、时变、集中参数、分布参数、确定性和随机等子类。

系统仿真是根据被研究的真实系统的数学模型研究系统性能的一门学科，现在尤指利用计算机去研究数学模型行为的方法。计算机仿真的基本内容包括系统、模型、算法、计算机程序设计与仿真结果显示、分析与验证等环节。

在系统仿真技术的诸多环节中，算法和计算机程序设计是很重要的一个环节，它直接决定原来的问题是否能够正确地求解。基于国际上仿真领域最权威、最实用的计算机工具——MATLAB®语言介绍仿真问题的编程与求解方法将是本书最显著的特点。在第1.2节中将介绍数学软件、仿真软件的仿真概况和现状；第1.3节着重介绍MATLAB/Simulink语言的发展状况、语言特色和在系统仿真领域的应用举例，通过学习可以初步领略MATLAB的强大功能；第1.4节中将介绍本书的结构和有关内容。

1.2　仿真软件的发展概况

早期的计算机仿真技术大致经历了几个阶段：20世纪40年代模拟计算机仿真；50年代初数字仿真；60年代早期仿真语言的出现等。20世纪80年代出现的面向对象仿真技术为系统仿真方法注入了活力。我国早在20世纪50年代就开始研究仿真技术了，当时主要用于国防领域，以模拟计算机的仿真为主。20世纪70年代初开始应用数字计算机进行仿真[1]。随着数字计算机的普及，近20年以来，国际、国内出现了许多专门用于计算机数字仿真的

仿真语言与工具，如CSMP、ACSL、SIMNON、MATLAB/Simulink、MatrixX/System Build和CSMP-C等，随着MATLAB/Simulink等仿真工具的日益强大，前面列出的很多仿真语言已经退出了历史舞台，而MATLAB/Simulink已经成为仿真领域事实上的首选计算机语言和工具。

1.2.1 早期数学软件包的发展概况

数字计算机的出现给数值计算技术的研究注入了新的活力。在现代计算技术的早期发展中，出现了一些著名的数学软件包，如美国的基于特征值的软件包EISPACK[2,3]和线性代数软件包LINPACK[4]、英国牛津数值算法研究组（Numerical Algorithm Group）开发的NAG软件包[5]及参考文献[6]中给出的Numerical Recipes程序集等，都是在国际上广泛流行的、有着较高声望的软件包。

美国的EISPACK和LINPACK都是基于矩阵特征值和奇异值解决线性代数问题的专用软件包。限于当时的计算机发展状况，这些软件包大都是由Fortran语言编写的源程序组成的。

例如，若想求出N阶实矩阵$\boldsymbol{A}$的全部特征值（用$\boldsymbol{W}_{\mathrm{R}}$、$\boldsymbol{W}_{\mathrm{I}}$数组分别表示其实虚部）和对应的特征向量矩阵$\boldsymbol{Z}$，则EISPACK软件包给出的子程序建议调用路径为：

```
CALL BALANC(NM,N,A,IS1,IS2,FV1)
CALL ELMHES(NM,N,IS1,IS2,A,IV1)
CALL ELTRAN(NM,N,IS1,IS2,A,IV1,Z)
CALL HQR2(NM,N,IS1,IS2,A,WR,WI,Z,IERR)
IF (IERR.EQ.0) GOTO 99999
CALL BALBAK(NM,N,IS1,IS2,FV1,N,Z)
```

由上面的叙述可以看出，要求取矩阵的特征值和特征向量，首先要给一些数组和变量，然后依据EISPACK的格式做出定义和赋值，并编写出主程序，再经过编译和连接过程，形成可执行文件，最后才能得出所需的结果。

英国的NAG和美国学者的Numerical Recipes工具包则包括了各种各样数学问题的数值解法，两者中NAG的功能尤其强大。NAG的子程序都是以字母加数字编号的形式命名的，非专业人员很难找到适合自己问题的子程序，更不用说能保证以正确的格式去调用这些子程序了。这些程序包使用起来极其复杂，谁也不能保证不发生错误，NAG数百页的使用手册就十几本！

Numerical Recipes一书[6]中给出的一系列算法语言源程序也是一个在国际上广泛应用的软件包。该书中的子程序有C、Fortran和Pascal等版本，适合于科学研究者和工程技术人员直接应用。该书的程序包由200多个高效、实用的子程序构成，这些子程序一般有较好的数值特性，比较可靠，被各国的研究者所信赖。

具有Fortran和C等高级计算机语言知识的读者可能已经注意到，如果用它们去进行程序设计，尤其当涉及矩阵运算或画图时，则编程会很麻烦。例如，若想求解一个线性代数方程，用户需要首先编写一个主程序，然后编写一个子程序去读入各个矩阵的元素，之后再编

写一个子程序，求解相应的方程（如使用Gauss消去法），最后输出计算结果。如果选择的计算子程序不是很可靠，则所得的计算结果往往可能会出现问题。如果没有标准的子程序可以调用，则用户往往要将自己编好的子程序逐条地输入计算机，然后进行调试，最后进行计算。这样一个简单的问题往往需要用户编写100条左右的源程序，输入与调试程序也是很麻烦的，并无法保证所输入的程序100%可靠。求解线性方程组这样一个简单的功能需要100条源程序，其他复杂的功能往往要求有更多条语句，如采用双步QR法求取矩阵特征值的子程序则需要500多条源程序，其中任何一条语句有毛病，甚至调用不当（如数组维数不匹配），都可能导致错误结果的出现。

用软件包的形式编写程序有如下的缺点：

- **使用不方便**。对不是很熟悉所使用软件包的用户来说，直接利用软件包编写程序是相当困难的，也容易出错。如果其中一个子程序调用发生微小的错误，则可能导致最终得出错误的结果。
- **调用过程繁琐**。首先需要编写主程序，确定对软件包的调用过程，再经过必要的编译和连接过程，有时还要花大量的时间去调试程序以保证其正确性，而不是想得出什么马上就可以得出的。
- **执行程序过多**。想求解一个特定的问题就需要编写一个专门的程序，并形成一个可执行文件，如果需要求解的问题很多，那么就需要在计算机硬盘上同时保留很多这样的可执行文件，这样，计算机磁盘空间的利用不是很经济，管理起来也将十分困难。
- **不利于传递数据**。通过软件包调用方式会针对每个具体问题形成一个孤立的可执行文件，因而在一个程序中产生的数据无法传入另一个程序，更无法使几个程序同时执行以解决所关心的问题。
- **维数指定困难**。在很多数学问题中最重要的变量是矩阵，如果要求解的问题维数较低，则形成的程序就不能用于求解高阶问题，如参考文献[7]中的程序维数均定为10阶。所以，有时为使程序通用，往往将维数设置得很大，这样在解小规模问题时会出现空间的浪费，而更大规模的问题仍然求解不了。在优秀的软件中往往需要动态地定维矩阵。

此外，这里介绍的大多数早期软件包都是由Fortran语言编写的，由于众所周知的原因，以前使用Fortran语言绘图并不是轻而易举的事情，它需要调用相应的软件包做进一步处理，在绘图方面比较实用和流行的软件包是GINO-F[8]，但这种软件包只给出绘图的基本子程序，如果要绘制较满意的图形则需要用户自己用这些低级命令编写出合适的绘图子程序。

除了上面指出的缺点以外，用Fortran和C等程序设计语言编程还有一个致命的弱点，那就是因为C语言本身的原因，致使在不同的机器平台上，扩展的C源程序代码是不兼容的，尤其在绘图及界面设计方面更是如此。例如，在PC机的Microsoft Windows操作系统下编写的C语言程序不能立即在Linux操作系统上直接运行，而需要在该操作系统上对源程序进行修改、重新编译后才可以执行。

尽管如此，数学软件包仍在继续发展，其发展方向是采用国际上最先进的数值算法，提供更高效的、更稳定的、更快速、更可靠的数学软件包。例如，在线性代数计算领域，全新的LAPACK已经成为当前最有影响的软件包[9]，但它们的目的似乎已经不再是为一般用户提

供解决问题的方法，而是为数学软件提供底层的支持。在新版的MATLAB中已经抛弃了一直使用的LINPACK和EISPACK，采用LAPACK为其底层支持软件包。

1.2.2 仿真软件的发展概况

从前面提及的软件包的局限性看，直接调用它们进行系统仿真将有较大的困难，因为要掌握这些函数的接口是一件相当复杂的事，准确调用它们将更难；此外，软件包函数调用直接得出的结果可信度也不是很高，因为软件包的质量参差不齐。

抛弃成型的软件包，另起炉灶自己编写程序也不是很现实的事，毕竟在成型软件包中包含有很多同行专家的心血，有时自己从头编写程序很难达到这样的效果，所以必须采用经验证的信誉好的高水平软件包或计算机语言来进行仿真研究。

仿真技术引起该领域各国学者、专家们的重视，建立起国际的仿真委员会（Simulation Councils Inc，SCi），该公司于1967年通过了仿真语言规范。仿真语言CSMP（Computer Simulation Modelling Language）应该属于建立在该标准上的最早的专用仿真语言。中科院沈阳自动化研究所在1988年推出了该语言的推广版本CSMP-C。

20世纪80年代初期，美国Mitchell and Gauthier Associate公司推出了依照该标准的著名仿真语言ACSL（Advanced Continuous Simulation Language）[10]。该语言出现后，由于其功能较强大，并有一些系统分析的功能，很快就在仿真领域占据了主导地位。

ACSL首先要求用户依照其语言规则建立一个模型文件，然后可以通过ACSL本身提供的命令对其进行仿真及辅助分析。ACSL与Fortran语言的主要区别在于：ACSL的语句更简练，内容更丰富，可以直接调用由Fortran编写的子程序；ACSL编程的结构比相应的Fortran语言更严格，程序的基本结构必须严格按照规定的格式来编写，否则所得出的仿真结果可能出现意想不到的错误。ACSL提供了几十个系统子模块（macros），其中包括很常用的线性和非线性子模块，如传递函数模块TRAN、积分器模块INTEG、超前滞后环节LEDLAG、延迟模块DELAY、死区非线性模块DEAD、磁滞回环BAKLSH和限幅积分器LIMINT等，用户可以利用这些子模块简单地编写出描述给定系统的仿真模型，然后采用ACSL提供的功能来对系统进行仿真分析，并绘制出结果的曲线表示。

编写完ACSL源程序后，可以采用ACSL的编译命令来编译并将此模型和ACSL库连接起来，形成一个可执行文件，这一过程完成之后，ACSL将自动给出提示符ACSL>，在这个提示符下用户可以输入相应命令。

例 1-1 著名的Van der Pol方程由下式给出，$\ddot{y}+\mu(y^2-1)\dot{y}+y=0$，若取$\mu=1$，并取状态变量为$y_1=y, y_2=\dot{y}$，则Van der Pol方程可以写成$\dot{y}_1=y_1(1-y_2^2)-y_2$，$\dot{y}_2=y_1$，这时可以由ACSL所定义的语言写出此系统的模型如下：

```
PROGRAM VAN DER POL EQUATION
CINTERVAL CINT=0.01
CONSTANT  Y1C=3.0, Y2C=2.5, TSTP=15.0
   Y1=INTEG(Y1*(1-Y2**2)-Y2, Y1C)
   Y2=INTEG(Y1, Y2C)
```

```
TERMT (T.GE.TSTP)
END
```

在此程序中把显示步长CINT设置为0.01,状态变量的初值由Y1C和Y2C表示,而终止仿真时间TSTP设置为15,该程序中变量T为实际仿真时间。编写了ACSL源程序后,可以用ACSL编译器对其编译、连接,生成可执行文件。运行该文件可以得出提示符ACSL>,在提示符下可以输入如下的语句:

```
ACSL> PREPAR T, Y1, Y2
ACSL> START
ACSL> PLOT Y1, Y2
```

用来通知ACSL模型在仿真时需要保留T、Y1、Y2三个参数,开始数字仿真,并绘制出系统的相平面图(Y1和Y2之间的关系曲线)。还可以用下面的命令修改系统内部的参数:

```
ACSL> SET Y1C=-1, Y2C=-3
```

和ACSL大致同时出现的还有瑞典Lund工学院Karl Åström教授主持开发的SIMNON[11]、英国Salford大学的ESL[12]等,这些语言的编程语句结构也很类似,因为它们所依据的标准都是相同的。

MATLAB语言的出现及普及将数值计算技术与应用带入了一个新的阶段,与之配套的Simulink仿真环境又为系统仿真技术提供了新的解决方案。MATLAB语言发行后,国际上又出现了很多仿照其思想的软件,如稍后出现的美国的商品软件Ctrl-C、Matrix-X、O-Matrix和韩国汉城国立大学权旭铉教授主持开发的CemTool,以及现在仍作为免费软件的Octave[13]、Scilab[14,15]等。本书将以当前最新的MATLAB版本为主要对象,系统地介绍MATLAB语言的编程技术及其在系统仿真领域的应用。

计算机代数系统或符号运算是本领域中又一个吸引人的主题,而解决数学问题解析计算又是C这样的计算机语言直接应用的难点。于是国际上很多学者在研究、开发高质量的计算机代数系统。早期IBM公司开发的mumath和Reduce等软件为解决这样的问题提出了新的思路。后来出现的Maple[16]和Mathematica[17]逐渐占领了计算机代数系统的市场,成为比较成功的实用工具。

早期的Mathematica可以和MATLAB语言交互信息,例如,通过一个称为MathLink的软件接口就可以很容易地完成这样的任务。为了解决计算机代数问题,MATLAB语言的开发者 —— 美国MathWorks公司也研制开发了符号运算工具箱(Symbolic Toolbox®),该工具箱将Maple及现在的muPad语言的内核作为MATLAB符号运算的引擎,使得两者能更好地结合起来。

这些软件和语言还是很昂贵的,所以有人更倾向于采用免费的、但编程结构类似于MATLAB的计算机语言,如Octave和Scilab,这些软件的全部源程序也是公开的,有较高的透明度,但目前它们的功能已经无法与越来越强大的MATLAB语言相比。

系统仿真领域有很多自己的特性,如果能选择一种能反映当今系统仿真领域最高水平、也是最实用的软件或语言介绍仿真技术,使读者能直接采用该语言解决自己的问题,将是很有意义的。实践证明,MATLAB语言和Simulink程序就是这样的仿真软件,由于它本身卓

越的功能，已经使得它成为自动控制、航空航天、汽车工程等诸多领域仿真的首选语言，并在其他科学与工程的研究中起着重要的作用，并且将在长时间内保持其独一无二的地位。所以在本书中将介绍基于MATLAB/Simulink的系统仿真理论与应用。

1.3 MATLAB语言简介

1.3.1 MATLAB语言发展简史

MATLAB语言的首创者Cleve Moler教授在数值分析，特别是在数值线性代数领域中很有影响[2~4,18~20]，他曾在密西根大学、斯坦福大学和新墨西哥大学任数学与计算机科学教授。1980年前后，时任新墨西哥大学计算机系主任的Moler教授在讲授线性代数课程时，发现了用其他高级语言编程极为不便，便构思并开发了MATLAB（MATrix LABoratory，即矩阵实验室），这一软件利用了他参与研制的、在国际上颇有影响的、基于特征值计算的软件包EISPACK[2]和线性代数软件包LINPACK[4]两大软件包中可靠的子程序，用Fortran语言编写了集命令翻译、科学计算于一身的一套交互式软件系统。所谓交互式语言，是指用户给出一条命令，立即就可以得出该命令的结果。该语言无须像C和Fortran语言那样，首先要求使用者去编写源程序，然后对其进行编译、连接，最终形成可执行文件。这无疑会给使用者带来极大的方便。在MATLAB下，矩阵的运算变得异常容易，所以它一出现就广受欢迎。

早期的MATLAB只能做矩阵运算，绘图也只能用极其原始的方法，即用星号描点的形式画图，内部函数也只提供了几十个。但即使其当时的功能十分简单，但当它作为免费软件出现时还是吸引了大批的使用者。

Cleve Moler和Jack Little等人成立了一个名叫The MathWorks的公司，Cleve Moler一直任该公司的首席科学家。该公司于1984年推出了第一个MATLAB的商品版本。当时的MATLAB版本已经用C语言做了完全的改写，其后又增添了丰富多彩的图形图像处理、多媒体功能、符号运算和它与其他流行软件的接口功能，使得MATLAB的功能越来越强大。最早的PC机版又称为PC-MATLAB，其工作站版本又称为Pro MATLAB。1990年推出的MATLAB 3.5i版是第一个可以运行于Microsoft Windows下的版本，它可以在两个窗口上分别显示命令行计算结果和图形结果。稍后推出的SimuLAB环境首次引入了基于框图的仿真功能，其模型输入的方式令人耳目一新，该环境就是人们现在熟知的Simulink。

MathWorks公司于1992年推出了具有划时代意义的MATLAB 4.0版本，并于1993年推出了其微机版，充分支持在Microsoft Windows下进行界面编程。1994年推出的4.2版本扩充了4.0版本的功能，尤其在图形界面设计方面更提供了新的方法。

1997年推出的MATLAB 5.0版本支持了更多的数据结构，如单元数组、数据结构体、多维数组、对象与类等，使其成为一种更方便、完美的编程语言。2000年10月，MATLAB 6.0问世，其在操作界面上有了很大改观，同时还给出了程序发布窗口、历史信息窗口和变量管理窗口等，为用户的使用提供了很大的方便；在计算内核上抛弃了其一直使用的LINPACK和EISPACK，而采用了更具优势的LAPACK软件包[9]和FFTW系统[21]，速度变得更快，数值性能也更好；在用户图形界面设计上也更趋合理；与C语言接口及转换的兼容性也更强；与之配套的Simulink 4.0版本的新功能也特别引人注目。2004年9月推出了MATLAB 7.0版

本，陆续提出了物理建模与仿真的理念与新方法。

MathWorks公司目前正在致力于新版本的开发、测试，现在每年推出两个新版本，已于2010年9月正式推出MATLAB R2010b版。本书以MATLAB R2010b版本为主介绍MATLAB/Simulink及其应用。

目前，MATLAB已经成为国际上最流行的科学与工程计算的软件工具，现在的MATLAB已经不仅仅是一个“矩阵实验室”了，它已经成为一种具有广泛应用前景的、全新的计算机高级编程语言，有人称它为“第四代”计算机语言，它在国内外高校和研究部门正扮演着重要的角色。MATLAB语言的功能也越来越强大，不断适应新的要求提出新的解决方法，越来越多的计算机语言都将提供和MATLAB/Simulink的直接接口。所以可以预见，在科学运算与系统仿真领域，MATLAB语言将长期保持其独一无二的地位。

1.3.2 MATLAB语言的特色

除了MATLAB语言的强大数值计算和图形功能外，它还有其他语言难以比拟的功能，此外，它和其他语言的接口能够保证它可以和各种各样的强大计算机软件相结合，发挥更大的作用。

MATLAB目前可以在各种类型的常用操作系统和计算机上运行，如在Windows系统、Linux系统、Mac OS X系统和其他一些机器上完全兼容。如果单纯地使用MATLAB语言进行编程而不采用其他外部语言，则用MATLAB语言编写出来的程序不做丝毫的修改便可以直接移植到其他机型上使用，所以说与其他语言不同，MATLAB与机器类型和操作系统基本上无关。

依作者的观点，MATLAB和其他高级语言之间的关系就像该高级语言和汇编语言的关系一样，因为虽然高级语言的执行效率要低于汇编语言，但其编程效率与可读性、可移植性要远远高于汇编语言。同样，MATLAB比一般高级语言的执行效率要低，而其编程效率与可读性、可移植性要远远高于其他高级语言，所以，在科学运算中较适合于从像MATLAB这样的专用高级语言入手，这不但可以大大地提高编程的效率，而且可以大大地提高编程的质量与可靠性。对于专门从事科学运算与系统仿真研究的人员来说，因为MATLAB语言可以轻易地再现C或Fortran语言几乎全部的功能，所以，即使用户不懂C或Fortran这样的程序设计语言，也照样可以设计出功能强大、界面优美、稳定可靠的高质量程序，且开发周期会大大地缩短，可靠性与可信度也会大大提高。

MATLAB语言具有较高的运算精度。由于采用了双精度数据结构，一般情况下在矩阵类运算中往往可达到10^{-15}数量级的精度，这当然符合一般科学与工程运算的要求。此外，MATLAB的符号运算还提供了强大的公式推导能力。

MATLAB是以复数矩阵作为基本编程单元的一种高级程序设计语言，它提供了各种矩阵的运算与操作，并有较强的绘图功能，所以得到广泛流传，成为当今国际科学与工程领域中应用最广、最受人们喜爱的一种软件环境。MATLAB是一个高度集成的软件系统，它集科学与工程计算、计算机仿真、图形可视化、图像处理和多媒体处理于一身，并提供了实用的Windows图形界面设计方法，使用户能设计出友好的图形界面。

Simulink是MATLAB语言应用的另一个亮点，它倡导基于框图的可视化建模与仿真

方法，当今，其领先于其他软件的多领域物理建模方法，更为复杂工程系统的建模与仿真提供了崭新的思路和方法。Stateflow支持的有限自动机也为离散事件系统和混杂系统的建模与仿真提供了实用工具，而Simulink和外界硬件环境的直接接口更搭建起纯数字仿真与半实物仿真及实时控制的桥梁，使MATLAB/Simulink的工程应用上了一个新的台阶。

MATLAB/Simulink在自动控制、航天工业、汽车工业、生物医学工程、语音处理、图像信号处理、雷达工程、信号分析和计算机技术等各行各业中都有极广泛的应用，在很多领域中，MATLAB/Simulink已经成为事实上的首选计算机语言。

1.3.3 MATLAB版本选择和建议

最新的MATLAB的符号运算工具箱放弃了Maple内核的支持，改用功能差很多的MuPAD，使得其符号运算能力大大降低，所以从符号运算和公式推导角度看，建议使用MATLAB R2008a（7.6版）或以前的版本，而不是最新的版本。另外，很多功能不能在64位MATLAB下运行，所以即使读者安装了64位操作系统，也建议安装32位的MATLAB。从仿真角度看，尤其是从后面介绍的多领域物理建模与仿真的研究看，建议安装最新的版本，如R2010b（7.11）版。如果想兼顾这两类要求，则建议同时安装这两个版本。

类似地，如果使用Linux系统或Mac OS X操作系统，也可以参考上述的版本建议。

如果想在R2008b及以后版本下使用符号运算工具箱，则应该注意以下几点：

（1）本书可能有若干符号运算命令不能正常执行，例如，MuPad的微积分推导功能被弱化了很多，求100阶导数的演示语句不能执行。

（2）早期版本的`maple()`函数不再支持，不能再直接调用Maple函数。

（3）本书编写了一些关于符号变量的重载函数，如`lyap()`函数等，早期版本下这些文件置于`@sym`路径下即可，但新版本不允许这样做，所以只能将这些文件复制到MATLAB根目录的`toolbox/symbolic/symbolic`下，然后运行命令`rehash toolboxcache`。

1.4 本书的结构和代码

1.4.1 本书的结构

学好MATLAB语言，可以将30字的学习准则作为座右铭，即**“带着问题学，活学活用，学用结合，急用先学，立竿见影，在用字上狠下工夫”**。和学习其他计算机语言一样，要很好地掌握MATLAB语言和Simulink工具一定要多实践，通过实践提高真正的运用水平。对本书的学生读者来说，至少在学习本书理论的同时要亲自将程序、模型在自己的计算机上实践一番，得出第一手经验。

本书第1章（本章）中将首先介绍系统仿真的概念，并介绍数学软件包、仿真语言的发展概况。第2章将介绍MATLAB语言程序设计的基础知识，包括一般的数据结构、语句结构、函数编写、图形绘制和用户图形界面的设计，并介绍提高MATLAB程序执行效率的技巧，旨在为读者提供关于MATLAB语言编程的必备知识；第3章将系统介绍MATLAB语言在现代科学运算中的应用，包括数值线性代数问题及求解、微积分问题的MATLAB求解、常微

分方程的数值解法、非线性方程与最优化问题的求解、动态规划及其在路径规划中的应用、数据插值与统计分析等内容，为后面主要介绍的系统仿真技术与方法奠定数学理论基础；第4章将初步介绍各种Simulink模块组，并介绍Simulink环境的基本使用方法，然后举例演示Simulink在数学建模中的应用，该章还将简要介绍线性系统的建模、仿真与分析，最后还将介绍随机激励下连续系统的仿真方法等；第5章将深入介绍Simulink仿真知识，如常用的模块应用技巧、非线性模块的搭建、代数环及避免、过零点检测、各类微分方程的Simulink框图求解、各种模块输出方法、仿真结果的三维动画显示、子系统和模块的封装技术，并将介绍开发自己的模块库的方法，并通过复杂系统的例子演示Simulink的建模与仿真方法；第6章将介绍Simulink仿真的高级技术，如利用MATLAB语句的Simulink建模与仿真方法、非线性模型的线性化技术、S-函数的编写及应用，最后将介绍基于MATLAB/Simulink和最优化问题求解方法的伺服系统最优控制器的设计方法；第7章侧重于介绍基于Simulink及下属模块集的工程系统建模与仿真方法，首先介绍多领域物理建模方法和必要性，然后介绍Simscape模块集的基础模块库及SimPowerSystems、SimElectronics和SimMechanics等实用模块集，分别介绍它们在电气系统、电子系统、电机拖动系统和机械系统中的建模与仿真实例；第8章将介绍非工程系统的建模与仿真方法，首先介绍药物动力学系统建模、仿真与控制问题的求解，然后介绍图像与影像处理问题的MATLAB/Simulink求解方法，最后介绍离散事件系统的仿真方法，包括Stateflow和SimEvents模块集的使用方法；第9章将介绍半实物仿真与实时控制技术。

1.4.2 代码下载和网上资源

本书中给出了大量作者编写的MATLAB程序和Simulink框图，读者可以从

`http://mechatronics.ece.usu.edu/simubook2ed/index.html`

直接下载。不过，还是建议读者将学习到的程序和框图亲自输入到计算机中，因为这也是一个有意义的学习和实践环节。如果读者得到的结果和书中给出的有差异，或遇到因为书中篇幅限制没有充分介绍的内容，再使用下载的程序。

MATLAB的全套PDF版手册可以从MathWorks网站上直接下载，此外，互联网有大量的资源可以利用，也有大量的论坛和活跃在各个论坛的热心人士为使用者解决所遇到的各种各样的问题，其中，比较有影响的网站如下。

- MathWorks官方网站：`http://www.mathworks.com`、`http://www.mathworks.cn`。
- 用户讨论区：`http://www.mathworks.com/matlabcentral/newsreader/`。
- MATLAB中文论坛：`http://www.ilovematlab.cn/`。
- 研学论坛：`http://bbs.matwav.com/`。

此外，众多高校BBS中有MATLAB专版，可以直接访问。作者有两个关于论坛利用的观点：第一，遇到问题时应该自己主动思考，不要直接发问，如果能自己找出答案将对提高水平大有裨益，过分依赖论坛并不是好的学习方式；第二，尽量积极回答论坛中你了解的问题，或积极参与探讨，共同提高水平。

1.4.3 书中英文字体说明

在这里我们对书中所用的字体作整体的说明，相信这些字体的使用对读者正确理解本书的内容有所帮助。

- 英文文体与公式中常量采用正体 Times-Roman 字体，如MATLAB、e、x 轴。
- 公式中变量字体采用 Times-Roman 斜体字，如 x、t 等；矩阵、向量名用黑斜体，如 $\boldsymbol{A}$、$\boldsymbol{x}$、$\boldsymbol{f}(t,\boldsymbol{x})$ 等。
- 程序清单、函数、命令语句及函数中变量名等采用打字机字体，如 `eig()`、`tic`、`tspan`、`stateflow` 等。
- 界面中的文字标识、Simulink 模块的名称等采用 Sans Serif 正体，如 File 菜单，OK 按钮、Step 模块等。

1.5 习 题

(1) MATLAB 提供了较好的演示程序，在 MATLAB 的命令窗口中输入 `demo` 命令，则可以直接运行演示程序。试运行 MATLAB 的演示程序，初步领略 MATLAB 的强大功能。

(2) 本书中给出了大量作者编写的程序和模型，在深入学习本书的内容前应该从互联网上下载这些文件，本书介绍的全部程序都是可重复的，一般程序都是几个语句，所以建议用户执行输入程序。边学习边阅读与运行这些程序将有助于更好地理解本书介绍的内容。

(3) MATLAB 提供了丰富的联机帮助系统，可以用 `lookfor` 命令查询关键词，用 `help` 或 `doc` 可以查询某个具体函数的调用格式，已知某 Riccati 矩阵方程如下：

$$\boldsymbol{P}\boldsymbol{A}+\boldsymbol{A}^{\mathrm{T}}\boldsymbol{P}-\boldsymbol{P}\boldsymbol{B}\boldsymbol{R}^{-1}\boldsymbol{B}^{\mathrm{T}}\boldsymbol{P}+\boldsymbol{Q}=\boldsymbol{0}$$

且已知

$$\boldsymbol{A}=\begin{bmatrix}-27 & 6 & -3 & 9\\ 2 & -6 & -2 & -6\\ -5 & 0 & -5 & -2\\ 10 & 3 & 4 & -11\end{bmatrix},\ \boldsymbol{B}=\begin{bmatrix}0 & 3\\ 16 & 4\\ -7 & 4\\ 9 & 6\end{bmatrix},\ \boldsymbol{Q}=\begin{bmatrix}6 & 5 & 3 & 4\\ 5 & 6 & 3 & 4\\ 3 & 3 & 6 & 2\\ 4 & 4 & 2 & 6\end{bmatrix},\ \boldsymbol{R}=\begin{bmatrix}4 & 1\\ 1 & 5\end{bmatrix}$$

试用 `lookfor riccati` 找出求解函数，再用 `help` 命令查询出该函数的具体调用格式，最后得出该方程的解。