Kalman 滤波器理论与应用
——基于 MATLAB 实现

金学波　著

科 学 出 版 社

北　京

内 容 简 介

本书以 Kalman 滤波器为主要介绍对象，包含基本原理、推导方法及其在跟踪系统中的应用，同时配套 MATLAB 源程序。具体内容包括 Kalman 滤波器、扩展 Kalman 滤波器、不敏 Kalman 滤波器及其在 RFID 系统的跟踪应用研究。

本书凝练了作者二十余年来对 Kalman 滤波器基础理论及在目标跟踪应用的研究成果，具体内容包括：根据目标运动特征进行自调整参数的“自适应动力学模型”、不敏变换的性能分析、RFID 跟踪系统的测量方程及其仿真平台等。

本书可作为自动化、电子信息、计算机应用、控制科学与工程、信号处理、导航与制导等相关专业高年级本科生和研究生的教材，也可供相关领域的工程技术人员和研究人员参考。

图书在版编目 (CIP) 数据

Kalman 滤波器理论与应用：基于 MATLAB 实现/金学波著．—北京：科学出版社，2016

ISBN 978-7-03-049655-3

Ⅰ．①K… Ⅱ．①金… Ⅲ．①卡尔曼滤波器－滤波理论 Ⅳ．①TN713

中国版本图书馆 CIP 数据核字(2016)第 201409 号

责任编辑：王 哲 赵微微 / 责任校对：郭瑞芝

责任印制：张 倩 / 封面设计：迷底书装

科 学 出 版 社 出版

北京东黄城根北街 16 号

邮政编码：100717

http://www. sciencep. com

中国科学院印刷厂 印刷

科学出版社发行 各地新华书店经销

*

2016 年 8 月第 一 版 开本：720×1 000 1/16

2016 年 8 月第一次印刷 印张：12

字数：237 000

定价：72.00 元

（如有印装质量问题，我社负责调换）

前　言

Kalman 滤波器是目前应用最广泛的估计方法，在实时跟踪领域具有不可替代的学术价值和应用价值。尤其是在今天，信息技术已经跨入“互联网+”的时代，基于各种传感器的处理是物联网、信息物理系统（Cyber Physical System，CPS）的关键技术，研究者及应用者对实时估计方法的认知度具有巨大的需求。

本书以 Kalman 滤波器为主要内容，较全面地介绍了 Kalman 滤波器的基本原理、推导过程以及在跟踪领域中的应用。全书分两个部分，第一部分为基础知识，共包括 5 章。第 1 章介绍了 Kalman 滤波器的基本特点、应用领域等，并简单介绍了本书的特色。第 2 章简要介绍了 MATLAB 语言的使用方法和编程技巧。第 3 章介绍了最小二乘估计方法的基本原理和推导过程，并给出了几种不同的最小二乘估计方法表达形式，它们是 Kalman 滤波器的理论基础。第 4 章在第 3 章的基础上，给出了基于线性系统的 Kalman 滤波器的递推算法。第 5 章首先给出了非线性系统模型，然后给出了能够应用于非线性系统的 Kalman 滤波器算法，包括扩展 Kalman 滤波器、不敏 Kalman 滤波器。

第二部分为应用部分，共包括 3 章，是前面基础知识的必要补充，分别是第 6～8 章。第 6 章介绍了跟踪系统的测量模型，并给出了跟踪系统仿真研究中经常用到的几种仿真轨迹。第 7 章给出了机动目标的建模方法，并在第 6 章给出的仿真轨迹的基础上，使用不同的运动模型进行了仿真研究。第 8 章介绍了基于 RFID 室内跟踪系统的仿真研究，包括 RFID 跟踪系统的特点、不规则采样系统模型及跟踪方法。

同时，本书还配有 MATLAB 程序源程序，其中包括 1 个已经取得软件著作权专利的软件平台——“基于 RFID 系统的移动目标轨迹数据软件平台”。本书的源程序可在科学出版社的网站（www.sciencep.com）下载，同时利用微信公众号（智能感知数据研究中心，微信号 datamole）对书中的知识点进行说明及解答疑问。

本书以 Kalman 滤波器为主要介绍对象，从基本原理、推导方法及其在跟踪系统中的应用完整勾勒出来。同其他相关著作比较，本书的学术价值在于：

（1）脉络更清晰、体系更完整。本书对 Kalman 滤波器理论的讲述更加完整，更加注重 Kalman 滤波器基础原理、推导过程，加之配套的 MATLAB 程序，能够使读者更容易理解和掌握 Kalman 滤波器理论体系。

（2）包含创新研究成果、具有较大学术价值。本书凝练了作者二十余年关于 Kalman 滤波器基础理论及在目标跟踪应用中的研究成果，包括已授权国家发明专利 4 项，EI、SCI 检索论文 40 余篇，多项软件著作权登记等。涉及的研究成果包括能够根据目标运动特征进行自调整参数的“自适应动力学模型”、不敏变换的性能分析、RFID 传感器

测量方程及其仿真平台等。其中，基于自适应动力学模型的 Kalman 滤波方法已经获得了国家发明专利授权，利用该方法的应用成果发表了 SCI 检索论文 *Closed-Loop Estimation for Randomly Sampled Measurements in Target Tracking System*。

感谢国家自然科学基金（No.61273002）、北京市教委科研计划重点项目（No.KZ201510011012）以及北京工商大学学术出版项目（No.ZZCB2015-15）对本书出版提供的经费资助。

本书的撰写工作历时近 3 年时间，作者编写、调试并运行了所有的 MATLAB 程序，对公式推导及文字描述进行了多次校正，希望做到深入浅出地叙述 Kalman 滤波器理论、方法及其应用。同时，也很希望和广大读者进行交流，恳请专家同仁批评指正（datamole@126.com），作者将不胜感激。

作　者

2016 年 8 月

目　　录

第 1 章　绪　　论

1.1　Kalman 滤波器简介

Rudolph Emil Kalman 在 1960 年给出了离散时间系统的 Kalman 滤波器，也就是现在多数教材可以看到的方程形式。目前，Kalman 滤波器有多种实现形式，通常，其最初提出的形式称为标准 Kalman 滤波器。除此以外，还有施密特扩展滤波器、信息滤波器以及平方根滤波器等变换形式。

先通过下面关于温度问题的例子来说明一下 Kalman 滤波器的原理。假设研究的是某个房间内的温度。这里假设该房间内各处的温度是相等的，房间内不存在空调、风扇、暖气等局部发热或散热的设备，根据生活经验，认为这个房间内的温度是恒定的。也就是说，下一时刻的温度等于当前时刻的温度。但是，这也并不是绝对的准确，所以，房间内温度的估计存在上下几摄氏度的偏差，用公式表示就是 $x(k+1)=A(k)x(k)+w(k)$，其中 $A(k)=1$。偏差 $w(k)$ 在这里假设是满足高斯白噪声分布的，即偏差跟时间没有关系而且是服从高斯分布的，已知噪声的方差用 $Q(k)$ 表示。另外，在房间内放置一个温度计，用于测量房间内的温度。但是，由于制造工艺等因素，温度计的测量值也不是绝对准确的，即温度计的测量值与真实值之间也存在一定的偏差，用公式表示是 $z(k)=C(k)x(k)+v(k)$，其中 $C(k)=1$。这个偏差 $v(k)$ 也假设是高斯白噪声的，方差为 $R(k)$。那么，对于任意时刻，可以得到该房间的两个温度值：根据生活经验得到的估计值（系统的预测值）和温度计的测量值（系统的测量值）。下面用这两个值结合他们各自的噪声来估算出房间的实际温度值。假如要估算 k 时刻的实际温度值，首先要根据 $k-1$ 时刻的温度值，来预测 k 时刻的温度。

（1）假设知道前一时刻的温度是 23℃，$Q(k)=9$。因为相信温度是恒定的，所以会得到 k 时刻的温度预测值是跟 $k-1$ 时刻一样的结论，也是 23℃，考虑噪声则该值偏差是 3℃。

（2）从温度计那里得到了 k 时刻的温度值，假设是 25℃，同时该值的偏差是 2℃，即 $R(k)=4$。

由于用于估算 k 时刻的实际温度有两个温度值，分别是 23℃和 25℃。究竟实际温度是多少呢？相信自己还是相信温度计呢？究竟相信谁多一点，Kalman 滤波器给出了这样的判断 $23+K(25-23)$，式中 K 称做 Kalman 滤波增益。其实 K 的取法有很多种，不同的取法给出的估计结果也不一样，估计的准确度也是不一样的。Kalman 滤波器的

优势在于，如果系统满足 $A(k)$、$C(k)$ 已知，$w(k)$ 及 $v(k)$ 为高斯白噪声且方差 $Q(k)$、$R(k)$ 已知的条件，Kalman 给出的 K 能给出最优的估计结果。

那么到底房间的温度是多少呢？Kalman 估计方法给出的 K 值为 0.75，即房间的温度估计值为 $23+0.75\times(25-23)=23.5$（具体计算方法见第 4 章例题 4.3）。可以看到，估计的温度是在预测温度的基础上进行了一个修正，修正的大小由测量值和滤波增益的乘积决定的，而滤波增益是由两个噪声方差决定的（具体计算方法见第 4 章）。

通过上面的例子可以看到，在估计的时候需要知道两个模型：一个是描述待估计状态变化规律的数学表达式（过程模型）；另一个是描述传感器与待估计状态之间测量关系的数学表达式（测量模型）。估计方法可以对变化规律和测量关系的偏差进行修正，而这些偏差如果满足高斯白噪声分布的话，Kalman 滤波器就会给出误差的方差最小的估计结果。

下面说明一下使用 Kalman 滤波器必需的模型，先来看过程模型。前面已经提到，它是待估计量 $x(k)$ 随着时间变化的一种关系，对于 Kalman 滤波器来说，它必须是已知的。那么，如何才能知道这个模型呢？这需要考虑需要估计的对象。

再来看看从牛顿惯性运动定律到 Kalman 滤波器，这是 Kalman 滤波器的一个很典型的应用，就是估计机动目标的运动问题。和温度估计问题不同的是，现在假设目标是在移动的，而不像刚才假设温度是不变的，因此就需要考虑目标的运动规律，假设目标运动符合牛顿惯性定律，小车做匀速直线运动，有一辆质量为 m 的小车，受恒定的力 F，沿着 r 方向做匀速直线运动。已知小车在 $t-\Delta T$ 时刻的位移是 $s(t-1)$，此时的速度为 $v(t-1)$。求 t 时刻的位移 $s(t)$是多少？

先来看看如图 1-1 所示的目标运动示意图。假设目标沿着虚线运动，每个空心圆圈是目标在某一时刻的位置，现在来判断一下，下一时刻目标会在 A、B、C 哪一个点？

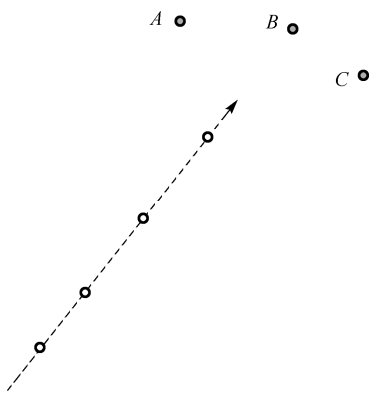


图 1-1　目标运动示意图

从图上可以看出，目标做的是匀速直线运动，再继续向前走，如果还是做这样的运动，根据惯性定律目标应该会在 B 点，而不是在 A 和 C 点。这里注意到，假设目标

还是做这样的运动，完全符合惯性定律。不考虑目标由于某种原因突然停下来、突然转弯的情况，因此，给出的基于牛顿惯性定律的运动目标模型只能描述这类目标的运动。

那么这类目标模型岂不是不十分准确？回答是：任何一个模型都只是某类实际目标的简单数学表达式，是有一定的偏差的。为了弥补这类偏差，过程模型往往会含有一个噪声项（如前面提到的 $w(k)$），来适当地扩大模型所能描述的实际情况。

使用 Kalman 滤波器必需的另一个模型是测量模型。通常，测量模型也是有噪声的，这比较好理解，因此测量时候的不确定因素造成不准确，如读数有一定的偏差、传感器的内部转换。测量矩阵是测量模型很重要的参数，它描述待估计量与可测量量之间的关系。例如，待估计状态为运动目标的位移 $s(k)$、速度 $v(k)$、加速度 $a(k)$，但我们只能通过传感器得到目标的位移量 $s(k)$，则测量方程为 $\boldsymbol{z}(k)=[1 \quad 0 \quad 0]\boldsymbol{x}(k)$，其中测量量用 $\boldsymbol{z}(k)$ 表示，待估计变量 $\boldsymbol{x}(k)=\begin{bmatrix} s(k) \\ v(k) \\ a(k) \end{bmatrix}$。如果能得到二维测量量，包含位移 $s(k)$ 和加速度 $a(k)$，则测量方程就变为 $\boldsymbol{z}(k)=\begin{bmatrix} 1 & 0 & 0 \\ 0 & 0 & 1 \end{bmatrix}\boldsymbol{x}(k)$，待估计变量仍为 $\boldsymbol{x}(k)=\begin{bmatrix} s(k) \\ v(k) \\ a(k) \end{bmatrix}$。

1.2　Kalman 滤波器的历史与研究现状

1809 年，高斯提出了一种称为最小二乘法的最优滤波方法，用来从已知的量测数量中确定天体所运行的轨道。由于最小二乘法在估计的时候不需要知道信号的先验统计知识，因此，最小二乘法在很多领域都有广泛的应用。

在 20 世纪 40 年代，为了对火力进行控制，美国学者维纳（Wiener）等在频域设计了 Wiener 滤波器，在平稳随机过程系统中实现了线性最优动态估计。通过维纳-霍普方程的计算，Wiener 滤波器获得了滤波器最优传递函数的解析解，从而可以实现抑制或选通包含多种信息的信号。但是，由于 Wiener 滤波器要求被估计信号和量测信号都必须符合平稳随机过程，并且在滤波过程中需要求解维纳-霍普方程，这使其计算量变大，需要的存储空间也很大，工程实现困难，因此，限制了 Wiener 滤波器的实用和推广。

为了克服 Wiener 滤波器的缺点，1960 年，匈牙利数学家 Rudolf Emil Kalman 提出现代滤波理论。他在随机估计理论中引入了状态空间的概念，并将信号过程看作白噪声作用下线性系统的输出，从而用状态方程来描述输入与输出之间的关系，从观测

量中估计出所需要的信号。Kalman 滤波不需要存储所有的历史数据，根据前一时刻的状态估计值和当前量测的信息，根据递推方式就可以计算出新的估计值，这大大降低了计算机的存储量和计算量，提高了实时处理的能力。与此同时，由于 Kalman 滤波器在时域内对信息进行分析，因此其不但可以对一维、平稳的随机过程进行估计，还可以对多维、非平稳的随机过程进行估计。

Kalman 滤波器由于涉及方法简单易行而得到了广泛应用，但是必须通过计算机才能执行。由于计算机在计算的过程中会不断累积、传递舍入误差和截断误差，而使误差协方差矩阵失去正定性，造成滤波估计不稳定。因此，研究者又陆续提出了一系列的数值鲁棒滤波算法，如平方根滤波、UD 分解滤波、奇异值分解滤波等。这些方法在有效改善 Kalman 滤波数值稳定性的同时，也提高了计算效率。

另一方面，标准的 Kalman 滤波器要求所建立的数学模型是精确的，随机干扰信号的统计特性是已知的，并且其对适用条件的要求也比较苛刻。例如，系统噪声是白色噪声，模型噪声和量测噪声相互独立，等等，但在实际系统中，系统由于一些不确定性导致不能满足这些条件，从而使 Kalman 滤波器失去了最优性，这大大降低了估计精度，甚至导致发散。为了解决这个问题，近年来研究者在滤波中引入了鲁棒控制思想，从而形成了鲁棒估计。

在实际的工程实践中，由于系统存在高斯或非高斯随机噪声干扰等不确定性，会导致系统存在不同程度的非线性滤波问题。但是，Kalman 最初提出的滤波理论只适用于线性系统，且要求观测方程也是线性的，因此，20 世纪 70 年代，Bucy 和 Sunahara 等提出了扩展 Kalman 滤波器（Extended Kalman Filter，EKF）。EKF 首先将非线性系统进行线性化处理，然后再利用广义 Kalman 滤波器进行状态估计。EKF 由于滤波思路简单明了而广泛应用于各个领域。但是，由于非线性系统在线性化过程中会引入线性化误差，导致最终的状态估计精度变差。由于需要计算雅克比矩阵，EKF 的计算量也很大，并且当雅克比矩阵计算不准确时，也会产生滤波发散问题。

Sigma 点 Kalman 滤波（SPKF）方法是一类基于高斯分布的近似非线性滤波方法，包括不敏 Kalman 滤波（Unscented Kalman Filter，UKF）、中心分布 Kalman 滤波（Central Difference Kalman Filter，CDKF）、平方根无迹 Kalman 滤波（Square-root Unscented Kalman Filter，SRUKF）、平方根中心分布 Kalman 滤波（Square-root Central Difference Kalman Filter，SRCDKF）等。这些方法的基本滤波思想大致相同，其中以 UKF 最为著名。对于非线性系统来说，EKF 是基于非线性函数的一阶泰勒公式展开实现的，因而其近似精度仅能达到一阶，而 UKF 采用若干服从高斯分布的函数组合来驱动，其精度至少可达二阶。并且，UKF 不需要计算雅克比矩阵，实时性更好。

随着研究的不断深入，针对非线性特性的逼近问题，在高斯域滤波体系中，对原有的方法进行了改进，并提出了一些新的方法。例如，Haykin 等于 2009 年提出了一种独立于 EKF、UKF 算法体系的新的滤波策略容积 Kalman 滤波器（Cubature Kalman Filter，CKF），该滤波器由于具有更加优良的特性而被广泛应用。

总的来说，对于很多问题，Kalman 滤波器及其衍生算法都可以得到较好的估计结果，效率最高甚至是最有用的。

1.3　本书的基本结构

本书配有与理论研究、仿真研究配套的近百个 MATLAB 源程序，包括 1 个已经取得软件著作权专利的软件平台“基于 RFID 系统的移动目标轨迹数据软件平台”。在与本书配套的网站上可以下载源程序及解答疑问，同时，读者也可以通过微信平台向作者提问或探讨。

全书主要分成两部分，第一部分为基础知识，共包括 5 章。

第 1 章介绍了 Kalman 滤波器的基本特点，并介绍了一个应用例子。

第 2 章对 MATLAB 语言的使用方法、必要函数和一些编程技巧进行了简介。本书假设读者已经熟悉 MATLAB 语言，并能够利用该语言进行编程，所以本章并没有对 MATLAB 进行深入介绍，有需要的读者可以参考 MATLAB 相关书籍。

第 3 章介绍了最小二乘估计方法的基本原理和推导过程，并给出了几种不同的表达形式，包括基本最小二乘估计方法、最小二乘加权估计和线性最小二乘递推估计方法，并对最小二乘方法的估计方差进行了深入研究，该方差是评价估计性能的定量指标。本章是 Kalman 滤波器的理论基础，尤其是递推最小二乘估计方法。值得一提的是，本章的估计方法都假设待估计量是恒定不变的。

第 4 章在第 3 章的基础上，假设待估计量是变化的，进而基于线性系统介绍了 Kalman 滤波器的递推算法。先对待估计量如何变化进行研究，前一时刻的待估计量 $x(k-1)$ 如何变为 $x(k)$？这需要系统模型之一——过程模型来描述。由于待估计量的变化，前一时刻的估计量在 k 时刻已经变为 $\hat{x}(k|k-1)$，将其称为“向前一步预测估计”，在 k 时刻的测量 $z(k)$ 到来时，向前一步预测估计 $\hat{x}(k|k-1)$ 又被进一步估计，得到了与真值更接近的当前步的估计结果 $\hat{x}(k|k)$。本章详细推导了 Kalman 滤波器，这其实是贝叶斯估计的过程。

第 5 章研究了非线性系统的 Kalman 估计方法，第 4 章的系统模型——过程模型和测量模型，都是具有线性关系的，如果系统模型包括非线性关系，就不能使用基本的 Kalman 滤波器。因此，研究者推导了几种能够估计非线性系统状态的方法，本章给出其中的两个：扩展 Kalman 滤波器、不敏 Kalman 滤波器。书中给出了详细的推导过程，并利用 MATLAB 进行仿真研究，如不敏变换和泰勒级数展开方法线性化的区别等。

作为前面基础知识的必要补充，本书的第二部分为应用部分，包括第 6～8 章。

第 6 章介绍了跟踪系统的测量模型，并给出了很多跟踪系统研究中经常用到的仿真轨迹，如蛇形机动、圆形机动、匀速直线运动、在 2 维平面内任意机动的仿真程序和数据，数据包括传感器测量模拟数据及运动目标的参考轨迹。另外本章还给出了

RFID 系统的测量数据仿真平台，可以任意设置 RFID 阅读器的个数和位置，模拟目标在 2 维平面内任意机动的采集数据。

第 7 章给出了机动目标的模型建模方法，包括常速度模型、常加速度模型、Singer 模型以及当前统计模型、Jerk 模型等，并介绍了基于数据驱动思想——状态估计闭环方法，给出了基于目标动力学统计特征的自适应动力学模型，大量仿真研究表明该模型可以很好地适应目标的运动特征。本章还包括对前一章给出的移动目标轨迹在不同运动模型下，对移动目标进行跟踪的仿真结果。

第 8 章介绍了基于 RFID 室内跟踪系统的仿真研究，包括 RFID 跟踪系统的特点，不规则采样测量下的系统模型以及不规则采样跟踪理论。其中，本章对基于多 RFID 阅读器的 EKF 跟踪，以及基于可变数量 RFID 阅读器的 UKF 跟踪，进行了仿真，结果表明即使在阅读器数量已经发生很大变化的情况下，通过本章给出的估计方法仍然可以得到平滑的估计结果。

第 2 章　MATLAB 基础知识介绍

本书分享了 Kalman 滤波器的 MATLAB 源程序，为了方便读者理解，本章对 MATLAB 基础知识进行简要介绍。

2.1　MATLAB 语言的主要特点

MATLAB 语言广泛应用于各种仿真研究中，尤其是自动控制领域。与 Python、R 等具有计算功能的语言相比，MATLAB 具有更为专业的工具包。MATLAB 语言由 Mathworks 公司开发，主体框架非常统一，因此也比其他语言更稳定。与此同时，由于售价太贵，MATLAB 也受到批评，在亲民方面与 Python、R 语言相差甚远，因此，很多人积极推崇后两者语言。

由于 MATLAB 语言稳定，并且在控制系统及相关领域具有非常丰富的工具包可供调用，研究人员可以非常方便地试验自己的算法思想，因此，本书沿用 MATLAB 进行讲解。虽然不能将算法直接写成执行文件，如果想开发真正的软件系统还需要使用其他的语言重写算法，但是对于算法研究人员说，MATLAB 仍然是首选。另外，作为入门方面的书籍，MATLAB 具有门槛低的优势，对一个大学本科生来说，几个学时之后就可以进行独立编程。特别地，MATLAB 语言在国内非常具有市场，很多高校仍然采用 MATLAB 作为主流的仿真工具。

MATLAB 参考书非常多，本书不再进行推荐列举，默认读者已经掌握了一定的 MATLAB 语言基本知识，并且曾经使用该语言编写过简单的小程序，因此，在 MATLAB 操作方面也不做过多的赘述。不过，2.2 节和 2.3 节简单回顾一下本书会用到的一些 MATLAB 编写程序的基本方法，以减轻读者重新翻阅其他参考书的负担。

2.2　MATLAB 编程介绍

MATLAB 程序通常包括四部分，下面来详细说明。

第一部分为清除主界面和内存空间。MATLAB 几十年来一直沿用主界面风格，可以在命令窗口（Command Window）上调试某一条语句是否正确，还可以执行输入命令。当程序出现错误时，命令窗口也会出现红色报错提示，想要输出某个变量的结果时，命令窗口也会非常方便地显示运行结果。可见主界面的内容是相当丰富的，但主界面保存的变量并不会自己清空，其历史命令（Command History）及工作空间（Workspace）

的变量自动保存着。这会使用户弄不清楚哪个是当前程序的执行结果，当程序报错时，也给错误排查带来困难。因此在程序执行的开始，最好清除主界面现有的内容，这样在本次程序执行之后，方便分析本次程序执行的结果。

在 MATLAB 中，清除主界面的语句如下。

clc：清空命令窗口的内容；

clear：清空内存变量；

close all：关闭所有图形窗口。

建议使用者在程序的开始就使用以上命令，这个习惯对管理程序数据空间很有帮助。MATLAB 的“管辖”的数据空间范围包括内存空间、路径，输入命令：

```
a=b+1
```

MATLAB 首先寻找变量 *b* 的值，如果内存空间已经有该变量的值，该语句在执行的时候就不会出错。但如果变量 *b* 在之前的程序中已被赋值，那么，在内存空间中已经存在 *b* 值，当前执行的程序当中并没有对 *b* 进行赋值，但由于 MATLAB 在内存空间中有 *b* 值就导致 *a* 被赋值。然而，此时程序是错的，因为没有在使用变量之前进行赋值。人们会大呼“程序通过了!”然后，很开心地关机，当下一次打开 MATLAB 时，MATLAB 会再次运行原程序，会发现系统出现错误提示“b 没有赋值”，这对一个初学者来讲是相当费解的一件事：程序昨天还好好的，为什么又出现了错误？现在明白了吧：由于没有使用 clear 语句清除内存空间，导致最初运行程序时，*b* 变量没有初始化的错误“漏”报。

这样的错误不是每次都会有，但是在算法研究的过程中错误率的确非常高。初学者由于缺乏专业的编程经验，在变量命名时比较随意，常常使用“c、A、PP、n”等简单的字母变量，在内存空间中留下痕迹。因此，养成一个好习惯至关重要，即在程序的开始使用清除主界面的 clc 语句，在“视觉”上有一个良好的开始，迎接“干干净净”的主界面；使用 clear 清除内存空间，使内存空间“干干净净”，程序每次都重新开始。

MATLAB 语言编写程序的第二部分是设置问题中交代的因变量，一般是一些已知的数据。第三部分是需要解决计算问题的结果变量，需要依据题意进行计算。第四部分是结果输出，有的只需要输出数据，而大多数情况下都需要用图形来表达。这几部分很多 MATLAB 书籍进行了详细介绍，本书不再赘述。下面使用一段具体的实例进行说明。

例 2.1　假设温度传感器 *A* 具有均匀分布测量噪声，噪声的均值为 0，方差为 2，请模拟当室内温度是 20℃的时候，传感器的输出数据是多少？给出 1000 个数。

```
% 清除主界面和内存空间
clc;
clear;
% 设置变量
m=20;
v=2;
```

在 MATALB 中，通过使用 randn 函数产生随机数。这里，测量值是真实值与测量噪声的叠加。根据测量方差和均值可以模拟出噪声，也就是将 randn 产生的结果乘以标准差，然后加上期望均值并将其与真实值叠加：

```
% 室内温度
    s1=m+sqrt(v)*randn(1,1000); %这里期望均值为 0，所以语句中没有显示均值。
```

最后，模拟出的测量噪声如图 2-1 所示，测量噪声是一个离散序列，所以形式非常简单，使用 plot 函数就可以实现：

```
plot(s1)
xlabel('采样时刻')
ylabel('测量数据')
```

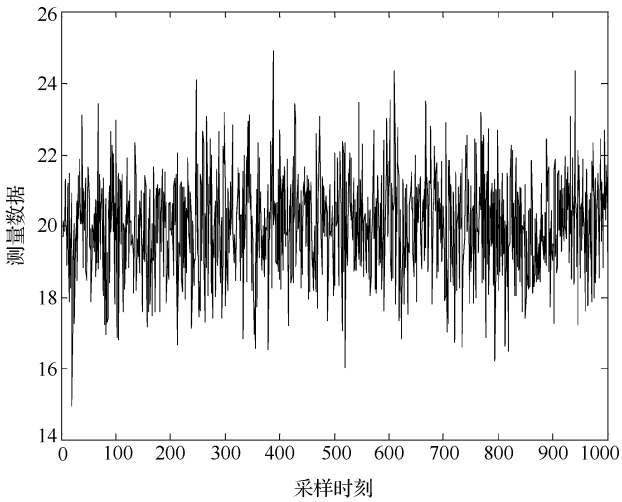


图 2-1　模拟传感器 *A* 的测量数据输出

MATLAB 画图函数非常丰富，还可以对图设定适当的标注来提高效果图的表达方式和内容，增强可视化效果，如横、纵坐标的设置，用不同线型来区别不同变量的曲线，或在一张图上画出多个图进行对比分析。详情可查阅 MATLAB 相关书籍，本书不再详述，在以后的应用中，会直接给出程序及对应的函数。

下面再看几个类似的例子。

例 2.2　若传感器 *B* 的噪声更大些，其噪声的均值为 0，方差为 8，请模拟当室内温度是 20℃的时候，传感器的输出数据是多少？给出 1000 个数。

同理，MATLAB 程序应该有四大部分，可以给出下面这段程序：

```
%C2_1
clc;
clear;
m=20;
```

```
v=8;
s2=m+sqrt(v)*randn(1,1000);
plot(s2)
xlabel('采样时刻')
ylabel('测量数据')
```

相应的结果如图 2-2 所示。

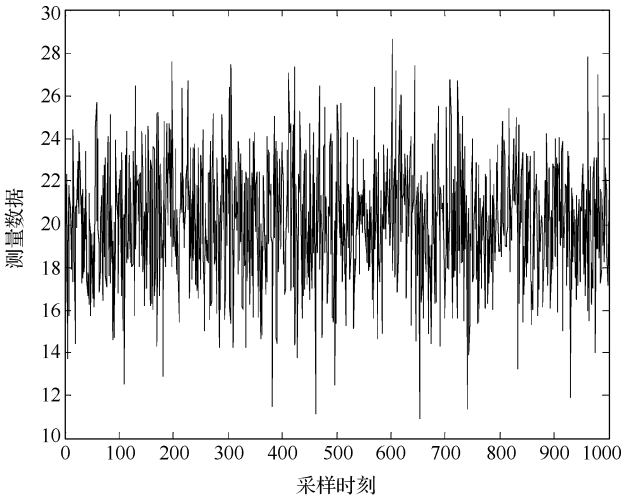


图 2-2　模拟传感器 *B* 的测量数据输出

例 2.3　如果室温在上升，实际的上升曲线是 2*t*+20，传感器 *A* 的输出数据是多少？给出 1000 个数。

与例 2.1 和例 2.2 不同，本例的实际室温随着时间的变化而变化，为了描述这个关系，需要给出时间 *t* 的变化，然后再考虑测量噪声与真值的关系。在实际的仿真中，需要指定时间 *t* 的仿真范围，也就是开始时间和结束时间，一般情况下开始时间可以设为从 0 开始，而结束的时间则可以根据实际情况进行设置。但是题中要求给出 1000 个数据，也就是 *t* 要包含 1000 个数据点。

可以用语句 t=0:0.01:9.99 来表示从 0 时刻到 9.99s 的采样时间，每 0.01s 采集一个数据共获得 1000 个测量数据。或者使用另外一个语句 t=linspace(0,9.99,1000)，也能输出 1000 个结果值。若使用传感器 *A* 进行测量，在真值之上需要叠加测量噪声获得测量数据，程序如下，图形结果如图 2-3 所示。

```
%C2-2
clc;
clear;
t=0:0.01:9.99;
m=2*t+20;
v=2;
```

```
s3=m+sqrt(v)*randn(1,1000);
plot(t,s3)
xlabel('采样时刻')
ylabel('测量数据')
```

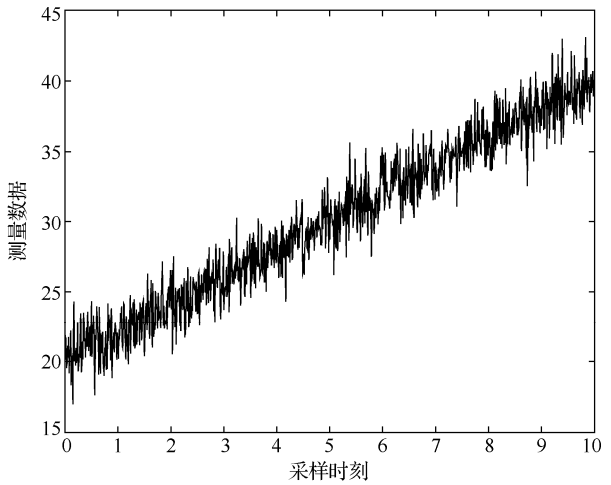


图 2-3　室内温度变化时模拟传感器 *A* 测量数据的输出

2.3　MATLAB 函数文件

本节对 MATLAB 函数的知识进行简单介绍。MATLAB 语言除了主程序（如上述编写的这些小程序）之外，还可以编写函数文件。以下面的小例子来说明函数文件的格式。

（1）在 MATLAB 的新建 function 文件中，编写如下程序并进行保存：

```
function s = sumfunc (a,b)
s = a+b;
```

（2）然后在 MATLAB 的工作空间中调用 sumfunc 函数，便可完成 MATLAB 函数的编写目的。在上述程序中，文件需要有关键字，即文件中第一行。函数文件必须由 function 开头，然后标明函数的输入输出变量。在函数体中需要表明函数输入输出变量之间的关系，如在上述小例子中，函数的输入输出是简单的求和关系。

在调用函数时可以使用 y=sumfunc (3,4)语句，会得出 y=7 的计算结果。

2.4　本 章 小 结

本章介绍了仿真语言的基础知识，以编写仿真程序的四部分步骤为例，简单介绍了 MATLAB 语言的基本使用方法，以及函数的编写。给出了几个小实例，来说明仿真过程产生的数据方法。如果读者想要更深入地了解 MATLAB 语言，可以参阅其他相关书籍。

第 3 章　最小二乘估计

最小二乘估计方法是一种利用观测数据估计线性模型中未知参数的方法，其基本思路是，选择合适的估计参数使模型输出与传感器实测输出数据之差的平方和最小。这种求误差平方和的方式可以避免正负误差相抵，便于数学求导计算（用误差的绝对值计算不便）。最小二乘方法是应用最广泛参数估计方法之一，在理论研究和工程应用中具有重要的作用，也是许多其他更复杂估计方法的基础。

先介绍一下最小二乘方法试图解决的问题。在实际问题中，经常会有这样的情况：有些量不能观测，或不容易观测，将其用$(O_1,\cdots,O_m)$表示，而另外的一些量$(\Omega_0,\Omega_1,\cdots,\Omega_m)$容易观测，它们之间经常有线性关系：

$$\Omega_0+\Omega_1O_1+\Omega_2O_2+\cdots+\Omega_mO_m=0 \tag{3-1}$$

对可观测量$(\Omega_0,\Omega_1,\cdots,\Omega_m)$观测若干次，得观测数据$(\Omega_{i0},\Omega_{i1},\cdots,\Omega_{im})$，其中$i=1,\cdots,n$。现在的问题是要根据这 n 次观测数据去估计$(O_1,\cdots,O_m)$。考虑到在观测过程中存在的影响与干扰，使得$(\Omega_0,\Omega_1,\cdots,\Omega_m)$的每次观测总会产生误差，同时模型的选择形式并非完全确切。所以合理的模型为

$$\Omega_{i0}+\Omega_{i1}O_1+\Omega_{i2}O_2+\cdots+\Omega_{im}O_m=\varepsilon_i \tag{3-2}$$

式中，$\varepsilon_1,\cdots,\varepsilon_n$是 n 次观测中的随机误差，这里要求$n\geqslant m$，即观测的次数不少于未知参数，否则无法估计。

作为一种应用非常广泛的估计方法，它的归属权还在科学界引起过争议。法国数学家兼天文学家勒让德 19 世纪初（1805 年）在研究天文学计算彗星的轨道时设法构造 m 个方程去求其解$(O_1,\cdots,O_m)$，他打破了前人在思想上囿于解方程这一思维定势，他认识到寻找的$(\hat{O}_1,\hat{O}_2,\cdots,\hat{O}_m)$应该使各次观测所产生的误差$\varepsilon_i(i=1,\cdots,n)$的平方和最小，即$\sum_{i=1}^{n}\varepsilon_i^2=\sum_{i=1}^{n}(\Omega_{i0}+\Omega_{i1}\hat{O}_1+\cdots+\Omega_{im}\hat{O}_m)^2$达到最小。他在估计巴黎子午线的时候就使用过这种方法。

几乎是同时，德国数学家高斯在解决行星轨道猜测问题时也使用了这个方法。当时在 1801 年，意大利天文学家皮亚齐发现了第一颗小行星——谷神星。经过 40 天的跟踪观测后，由于谷神星运行至太阳背后，皮亚齐失去了谷神星的位置。随后全世界的科学家利用皮亚齐的观测数据开始寻找谷神星，但是根据大多数人计算的结果来寻找谷神星都没有结果。时年 24 岁的高斯也计算了谷神星的轨道，奥地利天文学家奥尔

伯斯根据高斯计算出来的轨道重新发现了谷神星。高斯使用的最小二乘方法于 1809 年发表在他的著作《天体运动论》中。现在科学界普遍认可的是两位科学家独立发明的最小二乘方法。

在介绍最小二乘方法之前先明确一个现象，即传感器的测量数据和实际的真实值通常不是完全一致的。那么什么是传感器？广义上讲，传感器是能够感觉外界信息，并按一定规律将这些信息转换成可用输出信号的器件或装置。这一概念具有以下 3 个含义。

（1）传感器能够提取外界信息。

（2）传感器的输入量通常是非电量，如物理量、化学量、生物量等；而输出量是便于传输、转换、处理、显示的物理量，主要是电信号。例如，电容传感器的输入量可以是压力、位移、速度等非电物理量，输出则是电压信号。

（3）传感器的输出量与输入量之间满足一种测量关系。

传感器一般由敏感元件、转换元件和转换电路 3 部分组成，如图 3-1 所示。

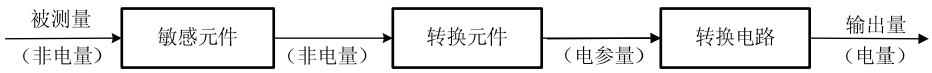


图 3-1　传感器的组成

其中敏感元件是传感器中能够感受被测量物体物理量的部分，例如，弹力敏感器件可以将压力转换为位移，并且压力与位移之间保持一定的函数关系。转换元件将敏感元件输出量转换为合适的电信号。例如，压变式压力传感器将应变力转换成变化的电阻。转换电路则将电量参数转换成电压、电流、频率等电量信号。

测量时，由于各种因素的影响，会对测量造成误差，导致物理量的测量数据和真实值不一样，这就是通常所说的“测得不准”。这种误差主要是系统误差和随机误差，系统误差包括一系列的错误：误读、误算、视差、刻度误差、磨耗误差、接触力误差等。系统误差在测量过程中是不变的，可以用计算或实验的方法求得，这表明系统误差是可以预测的，并且可以通过修正或调整的方式变小。

随机误差和系统误差的情况刚好相反，随机误差在测量过程中是改变的，因此不能预测某一时刻随机误差的值到底是多少，也无法进行修正和调整。随机误差主要表现为误差数据符合某一随机分布，如均匀分布、高斯分布或瑞利分布等。这种带有随机分布的测量误差，通常称为测量噪声。

现在总结一下传感器测量数据的特点，由于传感器的系统组成、人员操作等原因，传感器的测量数据通常存在一定的误差，其中一部分误差称为随机噪声（或测量噪声），这部分误差无法从根本上消除，只能在研究传感器测量数据时充分考虑。幸运的是，研究者给出了很多关于随机噪声的理论和方法，例如，用均值和方差描述其大小，采用这些方法可以分析传感器的随机误差，在利用传感器数据时，要充分考虑这些随机误差的影响，利用带有随机误差的测量数据尽量准确地估计实际物理量。

回顾一下图 2-1～图 2-3 关于传感器测量室温的数据。假设并不清楚实际温度，但可以通过温度传感器进行测量，现在的任务是利用这些测量数据来估计室内的实际温度，并且预测以后某个时间点的温度值。由于传感器随机测量噪声的存在，导致图 2-1 和图 2-2 每个数据点的值并不相同，但从整体而言，数据会在其均值上下波动。若实际的真实温度用 θ 表示，在每一个采样点的测量噪声用 $N(k)$ 表示，则每一个采样点测量的温度数据 $Z(k)$ 就是实际的真实温度 θ 与测量噪声 $N(k)$ 之和，即 $Z(k)=\theta+N(k)$。对于图 2-1 和图 2-2 中的数据，首先会想到可以利用求取均值的方式进行代替，从而消除噪声的影响。MATLAB 中求取均值的函数是 mean，可以使用语句 mean（s1）获得室内温度数据的平均值，20.0387（根据图 2-1 数据的均值得到）。同样，图 2-2 数据获得的室内平均温度为 20.0864℃。可以发现噪声越大，平均值的结果与真实室内温度相差越大。

但是对于图 2-3 的数据，不能再使用求平均值的方法获得室内温度了，因为室内温度不再是一个恒值，而是一个变量，仔细观察发现，图 2-3 中数据变化的方式可以描述为一条具有一定斜率的直线，如 $\theta_1 t+\theta_2$，当然测量数据中还包括测量噪声 $N(k)$，也就是图 2-3 的测量数据可以表示为 $\theta_1 t+\theta_2+N(k)$,如果能够估计该直线的斜率 θ_1 及其与纵轴的交点 θ_2，就可以获得温度的变化规律。接下来要介绍的最小二乘估计方法就可以完成这个任务。

3.1　最小二乘估计方法描述

用 $\boldsymbol{\theta}$ 表示待估计量，$Z(k)$、$N(k)$ 分别表示第 k 次观测数据和观测噪声，$\boldsymbol{H}(k)$ 为观测矩阵，线性观测方程可以用如下方程描述：

$$Z(k)=\boldsymbol{H}(k)\boldsymbol{\theta}+N(k) \tag{3-3}$$

再来看一下数据变化符合一次曲线的温度测量问题，将测量方差设为 $Z(k)=\theta_1 t(k)+\theta_2+N(k)$，写成状态方程的形式：

$$Z(k)=[t(k)\quad 1]\begin{bmatrix}\theta_1\\ \theta_2\end{bmatrix}+N(k) \tag{3-4}$$

要想得到温度的变化规律，需要很多数据，对于 k 个观测数据的向量，将这 k 个观测数据写成如下向量形式：

$$\boldsymbol{Z}_k=\begin{pmatrix}Z(1)\\ Z(2)\\ \vdots\\ Z(k)\end{pmatrix},\quad \boldsymbol{H}_k=\begin{pmatrix}\boldsymbol{H}(1)\\ \boldsymbol{H}(2)\\ \vdots\\ \boldsymbol{H}(k)\end{pmatrix},\quad \boldsymbol{N}_k=\begin{pmatrix}N(1)\\ N(2)\\ \vdots\\ N(k)\end{pmatrix} \tag{3-5}$$

则 k 个观测数据满足如下方程：

$$\boldsymbol{Z}_k = \boldsymbol{H}_k\boldsymbol{\theta} + \boldsymbol{N}_k \tag{3-6}$$

例 3.1　讨论先前提到的温度测量问题。回顾例题 2.3 中，室温在上升，变化曲线是 2*t*+20，利用模拟得到的 1000 个室温数据，如图 2-3 所示。现在想要得到温度的变化规律，需要估计直线的斜率 θ_1 及其截距 θ_2 两个参数。

根据式（3-6），$\boldsymbol{Z}_{1000}$ 为 1000 个测量数据排成一个列向量，$\boldsymbol{N}_{1000}$ 为每个测量数据的测量噪声，是未知量。观测矩阵是这样得到的：式（3-4）决定了测量矩阵每一行数据的组成，$\boldsymbol{H}_{1000}$ 的第一列是从 0 到 9.99s 每隔 0.01s 的测量时刻，第二列是全 1 的数据，它是 1000 行 2 列的矩阵，即

$$\boldsymbol{H}_{1000} = \begin{bmatrix} 0 & 1 \\ 0.01 & 1 \\ \vdots & \vdots \\ 9.99 & 1 \end{bmatrix} \tag{3-7}$$

下面讨论一下，如何根据测量得到的 1000 个数据估计描述温度变化规律，即 θ_1、θ_2 的值。前面已经说过，最小二乘估计方法要求测量模型输出 $\boldsymbol{H}(k)\hat{\boldsymbol{\theta}}$ 与实际测量数据输出之差 $Z(k)$ 的平方和达到最小，并且，所有的测量都满足这个要求，即 $\sum\left(Z(k)-\boldsymbol{H}(k)\hat{\boldsymbol{\theta}}\right)^2$ 达到最小。将求和性能指标写成下面矩阵形式，以便于求导运算：

$$\boldsymbol{J}(\hat{\boldsymbol{\theta}}) = (\boldsymbol{Z}_k - \boldsymbol{H}_k\hat{\boldsymbol{\theta}})^{\mathrm{T}}(\boldsymbol{Z}_k - \boldsymbol{H}_k\hat{\boldsymbol{\theta}}) \tag{3-8}$$

式中，上标 T 表示矩阵转置，先来看一下式（3-8）中估计量 $\hat{\boldsymbol{\theta}}$ 与实际真实值 $\boldsymbol{\theta}$ 之间的关系。将式（3-6）代入式（3-8）后，可以得到

$$\boldsymbol{J}(\hat{\boldsymbol{\theta}}) = (\boldsymbol{H}_k\boldsymbol{\theta} + \boldsymbol{N}_k - \boldsymbol{H}_k\hat{\boldsymbol{\theta}})^{\mathrm{T}}(\boldsymbol{H}_k\boldsymbol{\theta} + \boldsymbol{N}_k - \boldsymbol{H}_k\hat{\boldsymbol{\theta}})$$

如果真实的参数 $\boldsymbol{\theta}$ 和估计 $\hat{\boldsymbol{\theta}}$ 与测量噪声 $\boldsymbol{V}_k$ 无关，上式两边取期望可以得到

$$\boldsymbol{J}(\hat{\boldsymbol{\theta}}) = \boldsymbol{E}[(\boldsymbol{\theta}-\hat{\boldsymbol{\theta}})^{\mathrm{T}}\boldsymbol{H}_k^{\mathrm{T}}\boldsymbol{H}_k(\boldsymbol{\theta}-\hat{\boldsymbol{\theta}})] + \boldsymbol{E}(\boldsymbol{V}_k^{\mathrm{T}}\boldsymbol{V}_k) \tag{3-9}$$

仔细观察式（3-9），发现该式由两项为正的子式加和组成，后一项表明测量噪声的方差和估计参数无关。对于前一项来说，真实的参数 $\boldsymbol{\theta}$ 和估计参数 $\hat{\boldsymbol{\theta}}$ 越接近，该项的数值就会越小，因此，只要对性能指标函数 $\boldsymbol{J}(\hat{\boldsymbol{\theta}})$ 求极小值，找到的 $\hat{\boldsymbol{\theta}}$ 就是所求的估计参数。

求极值问题可以利用微分公式，令 $\dfrac{\partial \boldsymbol{J}(\hat{\boldsymbol{\theta}})}{\partial \hat{\boldsymbol{\theta}}} = 0$，再通过求解方程得到 $\hat{\boldsymbol{\theta}}$。将式（3-8）代入微分公式 $\boldsymbol{H}_{k-1}$，利用补充知识 1 中的矩阵求导相关知识可得

$$\frac{\partial}{\partial \hat{\boldsymbol{\theta}}}[(\boldsymbol{Z}_k - \boldsymbol{H}_k\hat{\boldsymbol{\theta}})^{\mathrm{T}}(\boldsymbol{Z}_k - \boldsymbol{H}_k\hat{\boldsymbol{\theta}})] = -2\boldsymbol{H}_k^{\mathrm{T}}(\boldsymbol{Z}_k - \boldsymbol{H}_k\hat{\boldsymbol{\theta}}) \tag{3-10}$$

令式（3-10）等于 0，可得

$$\boldsymbol{H}_k^{\mathrm{T}}\boldsymbol{Z}_k - \boldsymbol{H}_k^{\mathrm{T}}\boldsymbol{H}_k\hat{\boldsymbol{\theta}} = 0 \tag{3-11}$$

进一步推导得到

$$\boldsymbol{H}_k^{\mathrm{T}}\boldsymbol{Z}_k = \boldsymbol{H}_k^{\mathrm{T}}\boldsymbol{H}_k\hat{\boldsymbol{\theta}} \tag{3-12}$$

假设 $\boldsymbol{H}_k^{\mathrm{T}}\boldsymbol{H}_k$ 是满秩的，式（3-12）两边同时左乘 $\boldsymbol{H}_k^{\mathrm{T}}\boldsymbol{H}_k$ 的逆，可以获得估计参数 $\hat{\boldsymbol{\theta}}$ 的计算方法如下：

$$\hat{\boldsymbol{\theta}} = (\boldsymbol{H}_k^{\mathrm{T}}\boldsymbol{H}_k)^{-1}\boldsymbol{H}_k^{\mathrm{T}}\boldsymbol{Z}_k \tag{3-13}$$

下面利用上述估计方法估计图 2-3 所示室内温度的变化。已知 1000 个测量数据如图 2-3 所示，$\boldsymbol{H}_k$ 如式（3-7）所示，在例 2.3 程序 C2-2 的后面增加如下语句：

```
Hk=[t;ones（1,1000）]';
estim=inv（Hk'*Hk）*Hk'*s3';
```

程序中上标“'”表示矩阵的转置，ones（1,1000）产生 1 行 1000 列的全 1 矩阵，t 是前面程序所产生的从 0 到 9.99s，并且间隔 0.01s 的测量时间向量 0,0.01,0.02,…共 1000 个，Hk 是测量矩阵 $\boldsymbol{H}_k$，inv（Hk'*Hk）表示 $(\boldsymbol{H}_k^{\mathrm{T}}\boldsymbol{H}_k)^{-1}$。

运行程序后，获得估计参数 $\hat{\boldsymbol{\theta}}$ 的结果为

```
estim =
    1.9808
    20.0125
```

看到这个结果，读者会发现结果和真实值（根据第 2 章的数据模拟真实值 $\theta_1 = 2,\ \theta_2 = 20$）并不相同，的确如此，估计理论会接受估计结果和实际真值略有偏差。此时，一方面承认估计结果可能会有一定偏差，另一方面要想办法使偏差尽量小。研究者给出了很多估计方法，有些方法的估计偏差可以随着数据量的增多而逐渐趋于零，有些则不会，而是收敛到某一恒值。前者通常称为无偏估计，后者称为有偏估计，即有偏差的意思。相关书籍还会介绍很多其他的估计方法，本书则继续介绍最小二乘估计，以期通过最小二乘估计来改进这种有偏估计方法，获得更小的偏差。为了更好地让读者掌握最小二乘估计方法的基本原理和应用，给出一道题目进行练习，并提供相关 MATLAB 程序及结果图，供读者参考。

例 3.2　美国在 1946～1956 年的钢产量分别为 66.6 百万吨、84.9 百万吨、88.6 百万吨、78.0 百万吨、96.8 百万吨、105.2 百万吨、93.2 百万吨、111.6 百万吨、88.3 百万吨、117.0 百万吨、115.2 百万吨，请分别用一次线性曲线、二次曲线、三次曲线以及四次曲线来拟合这些数据，并预测 1957～1960 年的钢产量。

解　如图 3-2 所示，展示了从 1946～1956 年的钢产量的离散数据，用“○”表示，为了说明每一年的前后关系，使用虚线把每年的数据按照前后关系连接起来。

```
clc
clear
z=[66.6,84.9,88.6,78.0,96.8,105.2,93.2,111.6,88.3,117.0,115.2]';
```

```
plot(z,'o--')
ylabel('钢产量/百万吨')
```

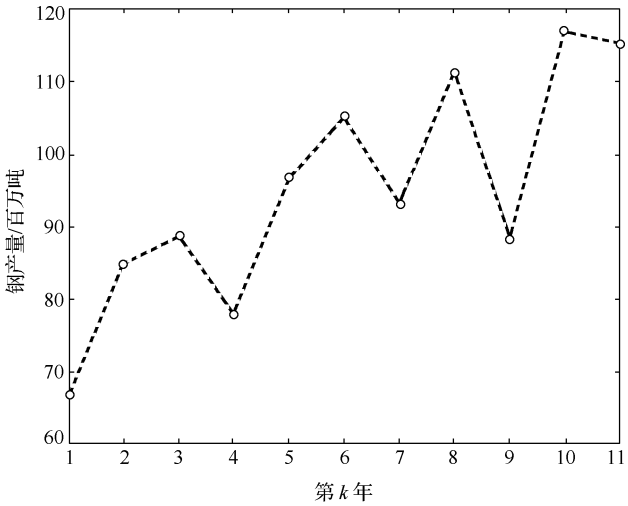


图 3-2　美国 1946～1956 年的钢产量

（1）先来考虑使用一次曲线拟合这些数据，一次曲线方程为

$$z(k)=a_1k+a_0+w(k) \tag{3-14}$$

写成具有矩阵形式的测量方程为

$$z(k)=[k \quad 1]\begin{bmatrix}a_1\\a_0\end{bmatrix}+w(k) \tag{3-15}$$

这里用 a_i 表示 N 次曲线的待估计参数将 1946 年取为第 1 个点，则 k 的取值为 1, 2…。$\boldsymbol{H}_k$ 和待估计参数 $\boldsymbol{\theta}$ 分别为

$$\boldsymbol{H}_k=\begin{bmatrix}1 & 1\\2 & 1\\3 & 1\\\vdots & \vdots\\11 & 1\end{bmatrix},\quad \boldsymbol{\theta}=\begin{bmatrix}a_1\\a_0\end{bmatrix} \tag{3-16}$$

利用最小二乘方法实现估计的程序如 C3-1 所示。程序中 Hk 表示式（3-16）的测量矩阵 $\boldsymbol{H}_k$，利用最小二乘方法得到的估计结果用 estim 表示，得到了 estim 后，设 $k=1,2,3,\cdots,11$，利用测量方程再次重构从 1946～1956 年的钢产量数据 ze(1:11)，又设 $k=12,13,14,15$ 使用循环语句预测后面 4 年，也就是从 1957～1960 年的数据，程序中也用 ze 表示。最后程序使用 hold on 语句及画图语句 plot，将拟合出的一次线性曲线画在刚才的结果图，也就是图 3-2 的上面。横坐标是从 1 到 15，其中包含了后面预测的四个数据点，如图 3-3 所示。

```
%C3-1
k=1:11;
Hk=[k;ones(1,11)]';          %计算式(3-16)中的 Hk
estim=inv(Hk'*Hk)*Hk'*z;     %计算最小二乘估计方法式(3-13)
ze=estim(1)*k+estim(2);      %利用估计的结果重新计算 1946～1956 年的钢产量
                             %来评价使用一次曲线的拟合结果

for i=1:4
ze(11+i)=estim(1)*(11+i)+estim(2); %计算 1957～1960 年的钢产量
end

hold on
plot(ze)
```

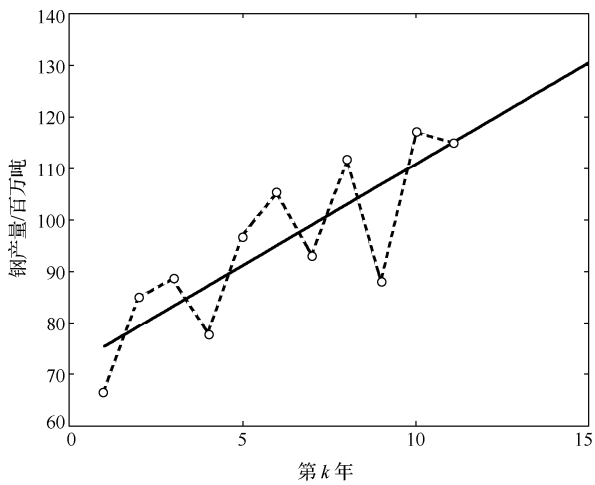


图 3-3　一次曲线拟合钢产量的结果

利用一次曲线拟合钢产量的结果得出估计参数 $a_1 = 3.9464$， $a_0 = 71.3582$。首先发现，从 $k = 1,2,3,\cdots,11$ 得到的曲线拟合结果和原来的数据是有差别的，这说明原来的数据在测量时存在噪声。如果不考虑噪声 $w(k)$ 的影响，利用估计出的线性一次曲线可以预测 1957～1960 年的钢产量分别为：118.7145 百万吨、122.6609 百万吨、126.6073 百万吨、130.5536 百万吨，这些数据点都落在图 3-3 的直线上。

（2）下面看一下若使用二次曲线进行拟合，结果会有什么不同，如果钢产量的数据满足二次曲线，则测量方程为

$$z(k) = a_2 k^2 + a_1 k + a_0 + w(k)$$

$$= [k^2 \quad k \quad 1]\begin{bmatrix} a_2 \\ a_1 \\ a_0 \end{bmatrix} + w(k)$$

和前面类似，将 1946 年取为第 1 个点，则 k 为 $1,2,\cdots$ $\boldsymbol{H}_k$ 和待估计参数 $\boldsymbol{\theta}$ 分别为

$$\boldsymbol{H}_k=\begin{bmatrix}1 & 1 & 1\\ 2^2 & 2 & 1\\ 3^2 & 3 & 1\\ \vdots & \vdots & \vdots\\ 11^2 & 11 & 1\end{bmatrix}$$

本例程序与 C3-1 程序相似 $\boldsymbol{\theta}=\begin{bmatrix}a_2\\ a_1\\ a_0\end{bmatrix}$，只是 $\boldsymbol{H}_k$ 有所不同。

```
k=1:11;
Hk=[k.^2;k;ones(1,11)]';
estim=inv(Hk'*Hk)*Hk'*z;
ze=estim(1)*k.^2+estim(2)*k+estim(3);

for i=1:4
ze(11+i)=estim(1)*((11+i).^2)+estim(2)*(11+i)+estim(3)
end

hold on
plot(ze)
```

运行该程序，则二次曲线拟合钢产量的结果如图 3-4 所示，可以推断出 1957～1960 年的钢产量分别为：114.5297 百万吨、116.3836 百万吨、117.9157 百万吨、119.1258 百万吨。

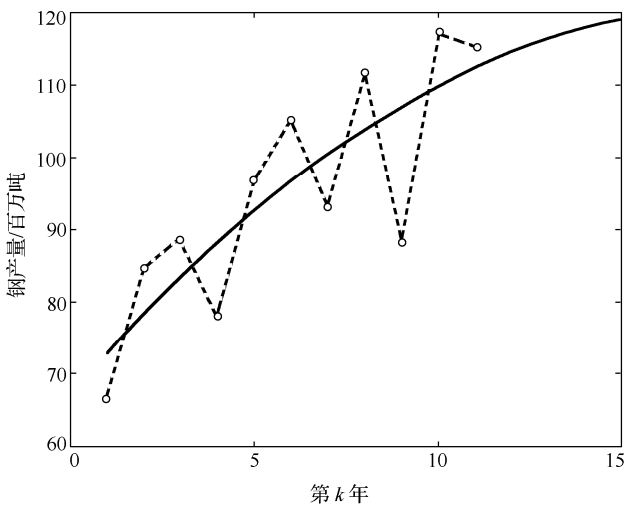


图 3-4　二次曲线拟合钢产量的结果

（3）若使用三次曲线进行拟合，则测量方程和相应参数为

$$\begin{aligned} z(k) &= a_3k^3 + a_2k^2 + a_1k + a_0 + w(k) \\ &= [k^3 \quad k^2 \quad k \quad 1]\begin{bmatrix} a_3 \\ a_2 \\ a_1 \\ a_0 \end{bmatrix} + w(k) \end{aligned} \tag{3-17}$$

$\boldsymbol{H}_k$ 和待估计参数 $\boldsymbol{\theta}$ 分别为

$$\boldsymbol{H}_k = \begin{bmatrix} 1^3 & 1^2 & 1 & 1 \\ 2^3 & 2^2 & 2 & 1 \\ 3^3 & 3^2 & 3 & 1 \\ \vdots & \vdots & \vdots & \vdots \\ 11^3 & 11^2 & 11 & 1 \end{bmatrix}, \quad \boldsymbol{\theta} = \begin{bmatrix} a_3 \\ a_2 \\ a_1 \\ a_0 \end{bmatrix} \tag{3-18}$$

对应程序如下，结果图如图 3-5 所示。

```
k=1:11;
Hk=[k.^3;k.^2;k;ones(1,11)]';
estim=inv(Hk'*Hk)*Hk'*z;
ze=estim(1)*k.^3+estim(2)*k.^2+estim(3)*k+estim(4);

for i=1:4
ze(11+i)=estim(1)*((11+i).^3)+estim(2)*((11+i).^2)+estim(3)*(11+i)+
estim(4);
end

hold on
plot(ze)
```

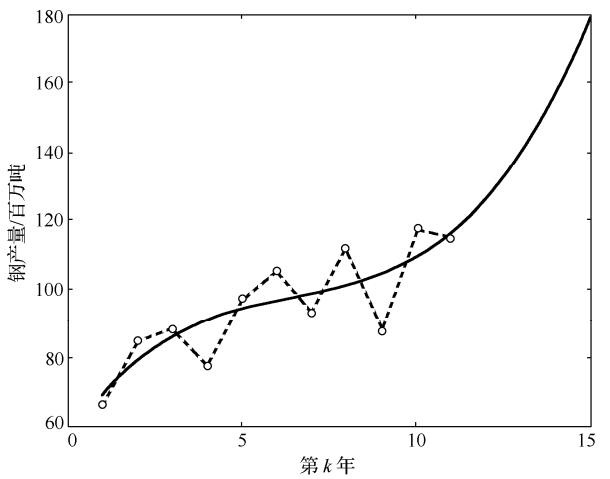


图 3-5　三次曲线拟合钢产量的结果

利用三次曲线拟合钢产量的结果可以推断出 1957～1960 年的钢产量分别为：126.1879 百万吨、139.7000 百万吨、157.3741 百万吨、179.8508 百万吨。

（4）若使用四次曲线进行拟合，则测量方程及相关参数为

$$\begin{aligned} z(k) &= a_4k^4 + a_3k^3 + a_2k^2 + a_1k + a_0 + w(k) \\ &= [k^4 \quad k^3 \quad k^2 \quad k \quad 1]\begin{bmatrix} a_4 \\ a_3 \\ a_2 \\ a_1 \\ a_0 \end{bmatrix} + w(k) \end{aligned} \tag{3-19}$$

$\boldsymbol{H}_k$ 和待估计参数 $\boldsymbol{\theta}$ 分别为

$$\boldsymbol{H}_k = \begin{bmatrix} 1^4 & 1^3 & 1^2 & 1 & 1 \\ 2^4 & 2^3 & 2^2 & 2 & 1 \\ 3^4 & 3^3 & 3^2 & 3 & 1 \\ \vdots & \vdots & \vdots & \vdots & \vdots \\ 11^4 & 11^3 & 11^2 & 11 & 1 \end{bmatrix}, \quad \boldsymbol{\theta} = \begin{bmatrix} a_4 \\ a_3 \\ a_2 \\ a_1 \\ a_0 \end{bmatrix} \tag{3-20}$$

对应程序如下，结果图如图 3-6 所示。

```
k=1:11;
Hk=[k.^4;k.^3;k.^2;k;ones(1,11)]';
estim=inv(Hk'*Hk)*Hk'*z;
ze=estim(1)*k.^4+estim(2)*k.^3+estim(3)*k.^2+estim(4)*k+estim(5);

for i=1:4
ze(11+i)=estim(1)*((11+i).^4)+estim(2)*((11+i).^3)+estim(3)*((11+i).
^2)+estim(4)*(11+i)+estim(5);
end

hold on
plot(ze)
```

可以推断出 1957～1960 年的钢产量分别为：128.8606 百万吨、146.8273 百万吨、172.0399 百万吨、206.1669 百万吨。依次运行上面四段程序，可以将 4 条拟合曲线画在一张图上，如图 3-7 所示。

可以看到，这几组曲线拟合的结果并不相同，在图上分别用箭头做了标注。结果表明，二次曲线拟合的结果偏低，而一、三、四次曲线拟合结果依次递增。

那么，这四种曲线哪种更为准确呢？当然不能凭感觉而论，这需要定量指标进行评价。可以使用曲线的拟合度计算哪一条曲线拟合得更准确来评价拟合效果。

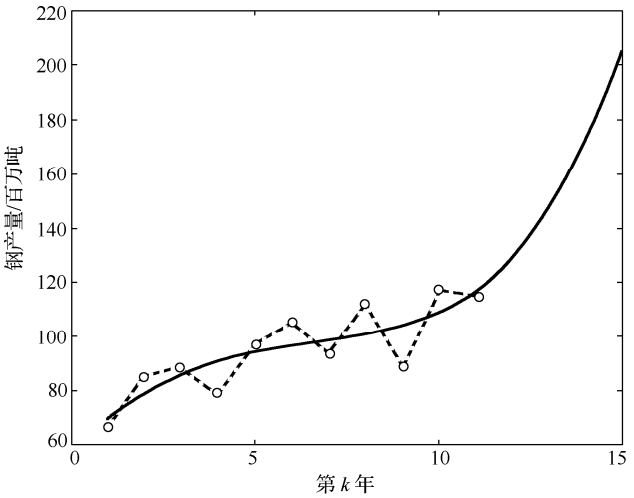


图 3-6　四次曲线拟合钢产量的结果

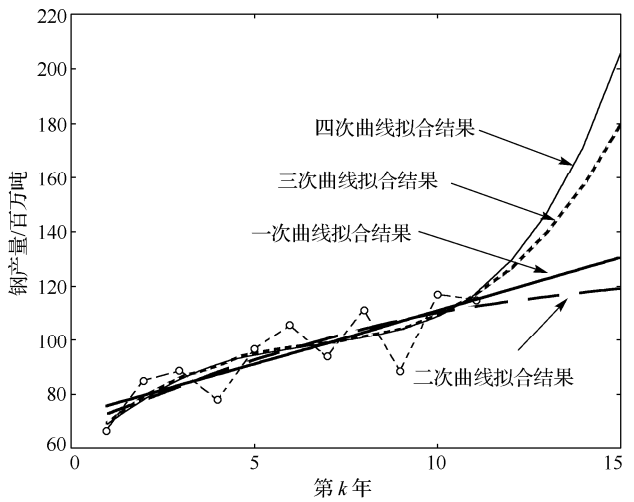


图 3-7　一、二、三、四次曲线拟合结果的比较

给出这道题主要是想让读者学会使用 MATLAB 语言和最小二乘方法进行曲线拟合。关于曲线拟合其实有很多软件，这些软件也基本采用刚才给出的方法，先假设拟合曲线的阶次，然后利用最小二乘方法估计待估计参数。

3.2　最小二乘加权估计

下面思考这样的问题，3.1 节给出的最小二乘方法并没有使用测量噪声方差。如果熟悉所使用的传感器，厂家也提供了传感器的测量性能，那么，经过实际测量能否

得到传感器的测量噪声方差，进而减小估计结果的偏差呢？从直觉上来讲应该是可以的。那么如何利用这个数值呢？可以将测量数据分别对待，如果测量传感器的噪声方差比较小，说明测量数据相对准确，可以利用这些数据来进行估计；如果测量传感器的噪声方差比较大，那么可以减小这些数据在估计中的作用。也就是对测量数据进行加权处理，而不再“一视同仁”。最小二乘加权估计方法就是从这个基本思想发展的。

加权的依据来源于测量的准确性。假设测量噪声为高斯白噪声 $\boldsymbol{N}_k$，其均值和协方差分别为 $E(\boldsymbol{N}_k)=0, E(\boldsymbol{N}_k\boldsymbol{N}_k^{\mathrm{T}})=\boldsymbol{R}_k$，若每次测量噪声不相关，则 $\boldsymbol{R}_k$ 为对角阵，即

$$\boldsymbol{R}_k=\begin{bmatrix} R(1) & & & \\ & R(2) & & \\ & & \ddots & \\ & & & R(k) \end{bmatrix} \tag{3-21}$$

式中，$R(1),R(2),\cdots,R(k)$ 是测量噪声在 $1,2,\cdots,k$ 时刻的方差。若 $\boldsymbol{N}_k$ 相关，则 $\boldsymbol{R}_k$ 为方阵，此时比较复杂。设权值用 $\boldsymbol{W}$ 表示，可取 $\boldsymbol{W}=\boldsymbol{R}_k^{-1}$。这样做的原因在于当数据测量方差大时，其权重应设置小些。当然，也可以使用其他函数实现这种关系，但是一般情况下都将测量噪声方差的“逆”取为权值。

考虑每个测量时刻的测量噪声方差，设线性最小二乘加权估计的性能指标为

$$\boldsymbol{J}_w(\hat{\boldsymbol{\theta}})=(\boldsymbol{Z}_k-\boldsymbol{H}_k\hat{\boldsymbol{\theta}})^{\mathrm{T}}\boldsymbol{W}(\boldsymbol{Z}_k-\boldsymbol{H}_k\hat{\boldsymbol{\theta}}) \tag{3-22}$$

接下来的任务是选择 $\hat{\boldsymbol{\theta}}$ 使之达到最小。

与 3.1 节相似，使用求极值的方式来解决问题。令 $\dfrac{\partial \boldsymbol{J}_w(\hat{\boldsymbol{\theta}})}{\partial \hat{\boldsymbol{\theta}}}=0$，则

$$\frac{\partial}{\partial \hat{\boldsymbol{\theta}}}[(\boldsymbol{Z}_k-\boldsymbol{H}_k\hat{\boldsymbol{\theta}})^{\mathrm{T}}\boldsymbol{W}(\boldsymbol{Z}_k-\boldsymbol{H}_k\hat{\boldsymbol{\theta}})]=-2\boldsymbol{H}_k^{\mathrm{T}}\boldsymbol{W}(\boldsymbol{Z}_k-\boldsymbol{H}_k\hat{\boldsymbol{\theta}})=0 \tag{3-23}$$

解上述方程得到

$$\hat{\boldsymbol{\theta}}=(\boldsymbol{H}_k^{\mathrm{T}}\boldsymbol{W}\boldsymbol{H}_k)^{-1}\boldsymbol{H}_k^{\mathrm{T}}\boldsymbol{W}\boldsymbol{Z}_k \tag{3-24}$$

例 3.3　继续例 3.1 的题目，现在假设知道测量噪声的方差为 2，且测量噪声为平稳过程，可以认为在每次测量中方差都相同。利用同样的测量数据，能否得到更为准确的估计？

可以试图利用测量噪声方差来构造具有加权最小二乘估计的方法，首先讨论一下权重的取法，如果测量噪声方差保持恒定不变，可以得到权值矩阵为

$$\boldsymbol{W}=\mathrm{diag}[1/2 \quad 1/2 \quad \cdots \quad 1/2] \tag{3-25}$$

这是一个 1000 行、1000 列的对角矩阵，可以用 MATLAB 语句 0.5*eye(1000)来实现。

在例2.3程序后面加上以下语句，利用MATLAB程序实现加权最小二乘估计方法如下：

```
Hk=[t;ones(1,1000)]';
```

```
W=0.5*eye(1000);
estim=inv(Hk'*W*Hk)*Hk'*W*s3'; %与前面的差别在于此处使用了权重 W
```

运行程序后得到了和原来相同的结果：

```
estim =
    1.9808
   20.0125
```

这是为什么呢？原来的设想是通过加权减小估计的偏差，而得到的这两个值却和原来完全相同，仔细分析发现：当权值为对角阵且所有的对角元素都相同时，矩阵运算中对角阵的作用与标量相同，用小写 $\boldsymbol{w}$ 表示式（3-24）中的标量权值，根据加权最小二乘估计方法可以得到

$$\begin{aligned}\hat{\boldsymbol{\theta}} &= (\boldsymbol{H}_k^{\mathrm{T}}\boldsymbol{w}\boldsymbol{H}_k)^{-1}\boldsymbol{H}_k^{\mathrm{T}}\boldsymbol{w}\boldsymbol{Z}_k \\ &= \boldsymbol{w}^{-1}(\boldsymbol{H}_k^{\mathrm{T}}\boldsymbol{H}_k)^{-1}\boldsymbol{H}_k^{\mathrm{T}}\boldsymbol{w}\boldsymbol{Z}_k \\ &= (\boldsymbol{H}_k^{\mathrm{T}}\boldsymbol{H}_k)^{-1}\boldsymbol{H}_k^{\mathrm{T}}\boldsymbol{Z}_k \end{aligned} \tag{3-26}$$

标量的计算使得求逆后运算 $\boldsymbol{w}^{-1}$ 与 $\boldsymbol{w}$ 抵消，所以最小二乘加权估计和最小二乘的结果是相同的。

但是，如果权值矩阵对角线上的数据不相同，也就是并不是每次测量的测量噪声方差都相同时，最小二乘加权估计就能够获得更准确的估计结果。从另一方面也可以看到，最小二乘估计在并不清楚测量噪声具体数值的时候，也能获得较好的估计结果，即如果传感器的测量噪声方差不随时间的变化而变化，考虑测量噪声方差的最小二乘加权方法和不考虑测量噪声方差的最小二乘方法可以获得相同的结果。但测量数据中的测量噪声方差不变是一种特殊情况，更普遍的情况是测量噪声方差 $R(k)$ 是一个时变量，这时候考虑噪声的方差，利用加权最小二乘法还是很有意义的。

3.3　线性最小二乘递推估计

本书的核心内容是 Kalman 滤波器，如前面所述，Kalman 滤波器的基本特点是可以随着时间向前递推处理测量数据，实现所谓的“实时处理测量数据”。但是目前讲述的最小二乘估计方法、加权最小二乘估计方法都必须要等到所有的数据全部收集好之后一次处理完成，在前面的例题中需要构建一个 1000×1 的矩阵 $\boldsymbol{Z}_{1000}$，在实际操作的时候也意味着，需要等待这 1000 个测量数据全部测完之后才能进行估计，这样就不能在线地处理测量数据。还可以这样考虑，如果想获得比较准确的估计，是不是一定要等到 1000 次测量结束之后才可以呢？还是获得几百个数据就可以得到比较好的估计结果呢？如果回答是后者，就不需要测 1000 个数据了。可见，在线处理会更高效，这就是接下来要介绍的递推估计方法。

递推估计的核心思想是，在获得测量数据以后及时地进行处理，而不必等到所有测量数据都获得后再进行估计，而且在估计当前时刻的参数时，可以利用前一次得到的结果，而不必将数据全部重新计算一遍。也就是说，线性最小二乘递推估计方法关注的是：如果已经得到第 $k-1$ 步的估计 $\hat{\boldsymbol{\theta}}(k-1)$，当第 k 步的测量 $Z(k)$ 到达后，在 $\boldsymbol{H}(k)$ 已知情况下，如何利用 $\hat{\boldsymbol{\theta}}(k-1)$、$Z(k)$ 和 $\boldsymbol{H}(k)$ 构造估计量 $\hat{\boldsymbol{\theta}}(k)$ 使估计结果越来越接近估计量的真实值。

根据前面的假设，前 $k-1$ 步的估计 $\hat{\boldsymbol{\theta}}(k-1)$ 已经得到，下面先看看这个估计是如何得到的。设加权矩阵为 $\boldsymbol{W}_{k-1}=\mathrm{diag}[\boldsymbol{W}(1),\boldsymbol{W}(2),\cdots,\boldsymbol{W}(k-1)]$，根据线性最小二乘加权估计方法式（3-24），可知估计矢量的计算方法 $\hat{\boldsymbol{\theta}}(k-1)$ 为

$$\hat{\boldsymbol{\theta}}(k-1)=[\boldsymbol{H}_{k-1}^{\mathrm{T}}\boldsymbol{W}_{k-1}\boldsymbol{H}_{k-1}]^{-1}\boldsymbol{H}_{k-1}^{\mathrm{T}}\boldsymbol{W}_{k-1}\boldsymbol{Z}_{k-1} \tag{3-27}$$

式中，测量数据 $\boldsymbol{Z}_{k-1}$、测量矩阵 $\boldsymbol{H}_{k-1}$ 以及相应地测量噪声 $\boldsymbol{N}_{k-1}$ 分别为

$$\boldsymbol{Z}_{k-1}=\begin{bmatrix} z(1)\\ z(2)\\ \vdots\\ z(k-1)\end{bmatrix},\quad \boldsymbol{H}_{k-1}=\begin{bmatrix} \boldsymbol{H}(1)\\ \boldsymbol{H}(2)\\ \vdots\\ \boldsymbol{H}(k-1)\end{bmatrix},\quad \boldsymbol{N}_{k-1}=\begin{bmatrix} N(1)\\ N(2)\\ \vdots\\ N(k-1)\end{bmatrix} \tag{3-28}$$

用 $\boldsymbol{M}_{k-1}$ 表示式（3-27）右边的前一部分，即 $\boldsymbol{M}_{k-1}=[\boldsymbol{H}_{k-1}^{\mathrm{T}}\boldsymbol{W}_{k-1}\boldsymbol{H}_{k-1}]^{-1}$，重写式（3-27）为

$$\hat{\boldsymbol{\theta}}(k-1)=\boldsymbol{M}_{k-1}\boldsymbol{H}_{k-1}^{\mathrm{T}}\boldsymbol{W}_{k-1}\boldsymbol{Z}_{k-1} \tag{3-29}$$

接下来就是推导的关键了，在 $\boldsymbol{H}(k)$ 已知的情况下，当得到第 $k-1$ 步的估计 $\hat{\boldsymbol{\theta}}(k-1)$，并且测量 $Z(k)$ 到达后，根据最小二乘加权估计方法式（3-24），可以得到 $\hat{\boldsymbol{\theta}}(k)$：

$$\hat{\boldsymbol{\theta}}(k)=[\boldsymbol{H}_k^{\mathrm{T}}\boldsymbol{W}_k\boldsymbol{H}_k]^{-1}\boldsymbol{H}_k^{\mathrm{T}}\boldsymbol{W}_k\boldsymbol{Z}_k \tag{3-30}$$

前面提到过，测量数据 $\boldsymbol{Z}_k$、测量矩阵 $\boldsymbol{H}_k$ 以及相应的测量噪声 $\boldsymbol{N}_k$ 分别为

$$\boldsymbol{Z}_k=\begin{bmatrix} z(1)\\ z(2)\\ \vdots\\ z(k)\end{bmatrix},\quad \boldsymbol{H}_k=\begin{bmatrix} \boldsymbol{H}(1)\\ \boldsymbol{H}(2)\\ \vdots\\ \boldsymbol{H}(k)\end{bmatrix},\quad \boldsymbol{N}_k=\begin{bmatrix} N(1)\\ N(2)\\ \vdots\\ N(k)\end{bmatrix} \tag{3-31}$$

同样设 $\boldsymbol{M}_k=[\boldsymbol{H}_k^{\mathrm{T}}\boldsymbol{W}_k\boldsymbol{H}_k]^{-1}$，将其代入式（3-30），得

$$\hat{\boldsymbol{\theta}}(k)=\boldsymbol{M}_k\boldsymbol{H}_k^{\mathrm{T}}\boldsymbol{W}_k\boldsymbol{Z}_k \tag{3-32}$$

因为式（3-27）和式（3-29）是待估计量 $\hat{\boldsymbol{\theta}}(k-1)$ 的求解方法，而式（3-30）和式（3-32）是 $\hat{\boldsymbol{\theta}}(k)$ 的求解方法，前面提到估计 $\hat{\boldsymbol{\theta}}(k)$ 时想利用 $\hat{\boldsymbol{\theta}}(k-1)$ 的值，那就需要先弄清楚两者之间的关系，要发现相同的部分，然后利用 $\hat{\boldsymbol{\theta}}(k-1)$ 替代这些相同部分，就可以建立 $\hat{\boldsymbol{\theta}}(k)$ 和 $\hat{\boldsymbol{\theta}}(k-1)$ 之间的关系了。

接下来看式（3-29）和式（3-32）中各个量之间的关系。将前面 $k-1$ 次的所有测量写成向量的形式为 $\boldsymbol{Z}_{k-1}=\begin{bmatrix} z(1) \\ z(2) \\ \vdots \\ z(k-1) \end{bmatrix}$，而前 k 次所有的测量数据也写成向量形式，$\boldsymbol{Z}_k$ 和 $\boldsymbol{Z}_{k-1}$ 具有关系 $\boldsymbol{Z}_k=\begin{bmatrix} \boldsymbol{Z}_{k-1} \\ Z(k) \end{bmatrix}$，同样道理，$\boldsymbol{H}_{k-1}$ 和 $\boldsymbol{H}_k$ 也具有类似的关系矩阵 $\boldsymbol{H}_k=\begin{bmatrix} \boldsymbol{H}_{k-1} \\ \boldsymbol{H}(k) \end{bmatrix}$。对于测量噪声来说，噪声的关系矩阵 $\boldsymbol{N}_k=\begin{bmatrix} \boldsymbol{N}_{k-1} \\ N(k) \end{bmatrix}$ 意味着噪声方差的关系满足 $\boldsymbol{W}_k=\begin{bmatrix} \boldsymbol{W}_{k-1} & 0 \\ 0 & \boldsymbol{W}(k) \end{bmatrix}$，其中 $\boldsymbol{W}(k)=\boldsymbol{R}^{-1}(k)$，$\boldsymbol{W}(k)$ 是第 k 次测量的噪声方差值。很显然，$\boldsymbol{M}_k$ 与 $\boldsymbol{M}_{k-1}$ 也是有关系的，将 $\boldsymbol{H}_k$ 和 $\boldsymbol{W}_k$ 代入 $\boldsymbol{M}_k$ 得

$$\boldsymbol{M}_k=\left\{[\boldsymbol{H}_{k-1}^{\mathrm{T}} \quad \boldsymbol{H}^{\mathrm{T}}(k)]\begin{bmatrix} \boldsymbol{W}_{k-1} & 0 \\ 0 & \boldsymbol{W}(k) \end{bmatrix}\begin{bmatrix} \boldsymbol{H}_{k-1} \\ \boldsymbol{H}(k) \end{bmatrix}\right\}^{-1} \tag{3-33}$$

进行矩阵相乘运算得到

$$\boldsymbol{M}_k=[\boldsymbol{H}_{k-1}^{\mathrm{T}}\boldsymbol{W}_{k-1}\boldsymbol{H}_{k-1}+\boldsymbol{H}^{\mathrm{T}}(k)\boldsymbol{W}(k)\boldsymbol{H}(k)]^{-1} \tag{3-34}$$

观察式（3-34）右边和式中的第一项，再考虑前面定义 $\boldsymbol{M}_{k-1}$ 时提到的 $\boldsymbol{M}_{k-1}=[\boldsymbol{H}_{k-1}^{\mathrm{T}}\boldsymbol{W}_{k-1}\boldsymbol{H}_{k-1}]^{-1}$，将式（3-34）中右边和式中的第一项用 $\boldsymbol{M}_{k-1}$ 代替后得到

$$\boldsymbol{M}_k=[\boldsymbol{M}_{k-1}^{-1}+\boldsymbol{H}^{\mathrm{T}}(k)\boldsymbol{W}(k)\boldsymbol{H}(k)]^{-1}$$

两边取逆后得到 $\boldsymbol{M}_{k-1}$ 和 $\boldsymbol{M}_k$ 的关系如下：

$$\boldsymbol{M}_{k-1}^{-1}=\boldsymbol{M}_k^{-1}-\boldsymbol{H}^{\mathrm{T}}(k)\boldsymbol{W}(k)\boldsymbol{H}(k) \tag{3-35}$$

可以看到，在已知 $\boldsymbol{M}_{k-1}$ 时，可以根据测量矩阵 $\boldsymbol{H}(k)$ 与权值矩阵 $\boldsymbol{W}(k)$ 可以求得 $\boldsymbol{M}_k$。到目前为止，确定了 $\boldsymbol{Z}_{k-1}$ 和 $\boldsymbol{Z}_k$、$\boldsymbol{H}_{k-1}$ 和 $\boldsymbol{H}_k$、$\boldsymbol{W}_{k-1}$ 和 $\boldsymbol{W}_k$、$\boldsymbol{M}_{k-1}$ 和 $\boldsymbol{M}_k$ 的关系，下面继续推导第 k 步估计 $\hat{\boldsymbol{\theta}}(k)$ 和前一步的估计 $\hat{\boldsymbol{\theta}}(k-1)$ 的关系，方法是将 $\boldsymbol{Z}_k$、$\boldsymbol{H}_k$、$\boldsymbol{W}_k$、$\boldsymbol{M}_k$ 代入式（3-32），然后再利用其中的 $\boldsymbol{Z}_{k-1}$、$\boldsymbol{H}_{k-1}$、$\boldsymbol{W}_{k-1}$、$\boldsymbol{M}_{k-1}$ 找出 $\hat{\boldsymbol{\theta}}(k-1)$ 的关系式，具体过程如下。

由式（3-32）得到

$$\begin{aligned}\hat{\boldsymbol{\theta}}(k)=\boldsymbol{M}_k\boldsymbol{H}_k^{\mathrm{T}}\boldsymbol{W}_k\boldsymbol{Z}_k&=\boldsymbol{M}_k[\boldsymbol{H}_{k-1}^{\mathrm{T}} \quad \boldsymbol{H}^{\mathrm{T}}(k)]\begin{bmatrix} \boldsymbol{W}_{k-1} & 0 \\ 0 & \boldsymbol{W}(k) \end{bmatrix}\begin{bmatrix} \boldsymbol{Z}_{k-1} \\ Z(k) \end{bmatrix} \\ &=\boldsymbol{M}_k[\boldsymbol{H}_{k-1}^{\mathrm{T}}\boldsymbol{W}_{k-1}\boldsymbol{Z}_{k-1}+\boldsymbol{H}^{\mathrm{T}}(k)\boldsymbol{W}(k)Z(k)]\end{aligned} \tag{3-36}$$

观察式（3-36），等式右边括号里 $\boldsymbol{H}_{k-1}^{\mathrm{T}}\boldsymbol{W}_{k-1}\boldsymbol{Z}_{k-1}$ 和 $\hat{\boldsymbol{\theta}}(k-1)$ 有关系。先由式（3-27）～式（3-29）可以得到 $\boldsymbol{H}_{k-1}^{\mathrm{T}}\boldsymbol{W}_{k-1}\boldsymbol{Z}_{k-1}$ 与 $\hat{\boldsymbol{\theta}}(k-1)$ 的关系如下：

$$\boldsymbol{H}_{k-1}^{\mathrm{T}}\boldsymbol{W}_{k-1}\boldsymbol{Z}_{k-1}=\boldsymbol{M}_{k-1}^{-1}\hat{\boldsymbol{\theta}}(k-1)$$

将上式代入式（3-36）可以得到

$$\hat{\boldsymbol{\theta}}(k)=\boldsymbol{M}_k[\boldsymbol{M}_{k-1}^{-1}\hat{\boldsymbol{\theta}}(k-1)+\boldsymbol{H}^{\mathrm{T}}(k)\boldsymbol{W}(k)Z(k)] \tag{3-37}$$

进一步需要保留 $\hat{\boldsymbol{\theta}}(k-1)$，但需消去 $\boldsymbol{M}_k$ 或 $\boldsymbol{M}_{k-1}^{-1}$。本书选择消去后者，将式（3-35）代入式（3-37）得到

$$\begin{aligned}\hat{\boldsymbol{\theta}}(k)&=\boldsymbol{M}_k[(\boldsymbol{M}_k^{-1}-\boldsymbol{H}^{\mathrm{T}}(k)\boldsymbol{W}(k)\boldsymbol{H}(k))\hat{\boldsymbol{\theta}}(k-1)+\boldsymbol{H}^{\mathrm{T}}(k)\boldsymbol{W}(k)Z(k)]\\&=\hat{\boldsymbol{\theta}}(k-1)-\boldsymbol{M}_k\boldsymbol{H}^{\mathrm{T}}(k)\boldsymbol{W}(k)\boldsymbol{H}(k)\hat{\boldsymbol{\theta}}(k-1)+\boldsymbol{M}_k\boldsymbol{H}^{\mathrm{T}}(k)\boldsymbol{W}(k)Z(k)\\&=\hat{\boldsymbol{\theta}}(k-1)+\boldsymbol{M}_k\boldsymbol{H}^{\mathrm{T}}(k)\boldsymbol{W}(k)[Z(k)-\boldsymbol{H}(k)\hat{\boldsymbol{\theta}}(k-1)]\end{aligned}$$

现将递推最小二乘估计总结如下。

设 $\boldsymbol{M}_0$、$\hat{\boldsymbol{\theta}}(0)$ 为迭代初始值，一般 $\hat{\boldsymbol{\theta}}(0)$ 为一合适的数，$\boldsymbol{M}_0$ 为一大的正数或正定矩阵，其维数与 $\boldsymbol{H}^{\mathrm{T}}(k)\boldsymbol{W}(k)\boldsymbol{H}(k)$ 相同，递推最小二乘方法由以下两个递推公式组成：

$$\boldsymbol{M}_k^{-1}=\boldsymbol{M}_{k-1}^{-1}+\boldsymbol{H}^{\mathrm{T}}(k)\boldsymbol{W}(k)\boldsymbol{H}(k) \tag{3-38}$$

$$\hat{\boldsymbol{\theta}}(k)=\hat{\boldsymbol{\theta}}(k-1)+\boldsymbol{M}_k\boldsymbol{H}^{\mathrm{T}}(k)\boldsymbol{W}(k)[Z(k)-\boldsymbol{H}(k)\hat{\boldsymbol{\theta}}(k-1)] \tag{3-39}$$

式中，$\boldsymbol{M}_k\boldsymbol{H}^{\mathrm{T}}(k)\boldsymbol{W}(k)$ 为滤波增益，描述的是在当前测量 $Z(k)$ 及前一个估计的基础上，修正估计误差的力度。$Z(k)-\boldsymbol{H}(k)\hat{\boldsymbol{\theta}}(k-1)$ 为新息，指的是“新”的信息，是指在当前测量数据到来后，给估计“引入”新的修正量。

下面用程序实现式（3-38）、式（3-39）的递推最小二乘估计方法。把这段程序加在前面产生图 2-3 的温度测量数据程序 C2-2 后面，就可以利用递推最小二乘方法得到估计结果图。为节约篇幅，这里将前一段程序省略。

用 ea 表示待估计变量，其初值设为全 0 矩阵，因为此时待估计变量为两个，将其写成一个 2×1 的向量，用 M 表示 $\boldsymbol{M}_k$，其初值 $\boldsymbol{M}_0$ 设为对角元素为 1000 的 2×2 维矩阵。使用 estimate 存储所有估计的结果，先设为空矩阵，然后使用式（3-38）、式（3-39）进行迭代递推，最后给出 θ_1、θ_2 的递推估计结果 $\theta_1(k)$、$\theta_2(k)$。

```
%C3-2
%复制前面 C2-3 程序到这里，如果没有前面的程序，就没有测量数据 s3，程序会提示有错误
ea=[0;0];          %设置初值 θ̂(0)=[0;0]
M=diag(1)*1000; %设 M0=[1000 0;0 1000]
estiamte=[];     %存储估计结果，将要保存为向量的形式
W=0.5;

for i=1:1000     %因为有 1000 个测量数据，所以循环设置 1000 次
h=[t(i) 1];
M=inv(inv(M)+h'*W*h);        %计算式(3-38)
ea=ea+M*h'*W*(s3(i)-h*ea);  %计算式(3-39),s3(i)为测量数据

estiamte=[estiamte,ea]; %保存估计结果
```

```
end

plot(estiamte(1,:));figure
plot(estiamte(2,:));
```

在递推估计的过程中，每一次的估计都用到前一次的估计结果，这样能够实现所谓的在线估计。程序的最后两个语句画出了两张结果图，分别如图 3-8 和图 3-9 所示。结果图的曲线具有递推算法结果的典型特征：从初值开始，经过一段振荡之后，逐渐收敛到一稳定值。可以看到，这个稳定值和前面介绍最小二乘法的结果其实是一样的，不过图上可以看出在递推的过程中估计值是在不断变化的。

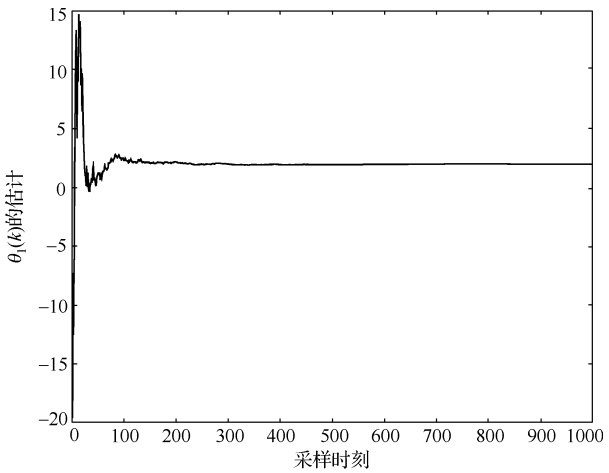


图 3-8　$\theta_1(k)$ 的估计结果

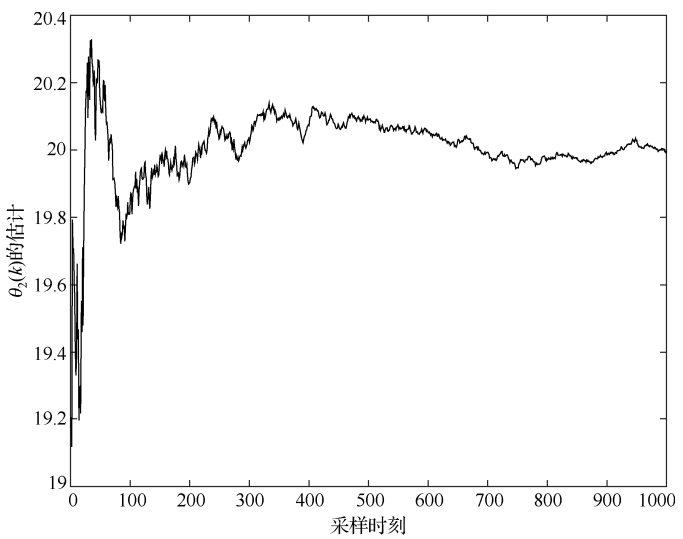


图 3-9　$\theta_2(k)$ 的估计结果

3.4　最小二乘的性能——估计方差

评价一种估计方法的好坏是很有必要的。估计结果和实际真实值不一样，无论是研究者还是应用者都能接受这个现象。只不过态度可能会不一样，研究者或心安理得、或摩拳擦掌，而应用者则处于不得不、很被动的状态，但不管怎样，估计的结果和实际真值不一样，估计误差的存在还是可以被接受的。定量描述估计的误差，在众多估计方法“大比拼”之中，估计方差是一个很关键的评价指标。本节给出几种最小二乘方法滤波增益和估计方差的求解方法，这些也是第 4 章 Kalman 滤波器的理论基础。

估计方差的定义为

$$\boldsymbol{P}(k)=E[(\boldsymbol{\theta}-\hat{\boldsymbol{\theta}}(k))(\boldsymbol{\theta}-\hat{\boldsymbol{\theta}}(k))^{\mathrm{T}}]$$

式中，$\boldsymbol{\theta}$ 为待估计参数的真值，$\hat{\boldsymbol{\theta}}(k)$ 为第 k 步的估计值，$E(\cdot)$ 表示求均值。以此类推前一步的估计方差为 $\boldsymbol{P}(k-1)=E[(\boldsymbol{\theta}-\hat{\boldsymbol{\theta}}(k-1))(\boldsymbol{\theta}-\hat{\boldsymbol{\theta}}(k-1))^{\mathrm{T}}]$。一般可以认为，所得的估计方差越小，估计方法就越好。估计方差最小的，估计方法就是最好的，这就是所谓的最优估计。

接下来以递推最小二乘方法为例，研究一下最小二乘方法的估计方差。前面已经讲过，权值 $\boldsymbol{W}(k)$ 一般取为测量噪声方差 $\boldsymbol{R}(k)$ 的倒数，如果权值 $\boldsymbol{W}(k)$ 为一个矩阵的话，则取为 $\boldsymbol{R}(k)$ 的逆。从现在开始都用 $\boldsymbol{R}^{-1}(k)$ 来代替 $\boldsymbol{W}(k)$，式（3-38）、式（3-39）中的滤波增益 $\boldsymbol{K}(k)$ 就变为 $\boldsymbol{K}(k)=\boldsymbol{M}_k\boldsymbol{H}^{\mathrm{T}}(k)\boldsymbol{R}^{-1}(k)$，将递推最小二乘估计方法估计 $\hat{\boldsymbol{\theta}}(k)=\hat{\boldsymbol{\theta}}(k-1)+\boldsymbol{K}(k)[\boldsymbol{Z}(k)-\boldsymbol{H}(k)\hat{\boldsymbol{\theta}}(k-1)]$ 代入估计方差的定义，可以得到

$$\begin{aligned}P(k)&=E[(\boldsymbol{\theta}-\hat{\boldsymbol{\theta}}(k))(\boldsymbol{\theta}-\hat{\boldsymbol{\theta}}(k))^{\mathrm{T}}]\\&=E[(\boldsymbol{\theta}-\hat{\boldsymbol{\theta}}(k-1)-\boldsymbol{K}(k)(Z(k)-\boldsymbol{H}(k)\hat{\boldsymbol{\theta}}(k-1)))\\&\quad(\boldsymbol{\theta}-\hat{\boldsymbol{\theta}}(k-1)-\boldsymbol{K}(k)(Z(k)-\boldsymbol{H}(k)\hat{\boldsymbol{\theta}}(k-1)))^{\mathrm{T}}]\end{aligned}$$

再将测量方程 $Z(k)=\boldsymbol{H}(k)\boldsymbol{\theta}+N(k)$ 代入上式，考虑到待估计值 $\boldsymbol{\theta}$ 与 $N(k)$ 不相关，经整理后可以得到

$$\boldsymbol{P}(k)=[\boldsymbol{I}-\boldsymbol{K}(k)\boldsymbol{H}(k)]\boldsymbol{P}(k-1)[\boldsymbol{I}-\boldsymbol{K}(k)\boldsymbol{H}(k)]^{\mathrm{T}}+\boldsymbol{K}(k)\boldsymbol{R}(k)\boldsymbol{K}^{\mathrm{T}}(k)\qquad(3\text{-}40)$$

接下来讨论递推最小二乘方法的估计性能问题，其估计结果是最优的吗？或者式（3-38）、式（3-39）给出的估计能够使估计方差式（3-40）最小吗？要是能使估计方差式（3-40）最小，那最小二乘方法就是最优的。

反过来，能使估计方差式（3-40）最小的滤波增益 $\boldsymbol{K}(k)$ 又是什么呢？是 $\boldsymbol{K}(k)=\boldsymbol{M}_k\boldsymbol{H}^{\mathrm{T}}(k)\boldsymbol{R}^{-1}(k)$ 吗？如果是的话，也说明最小二乘方法是最优的。下面来求解使式（3-40）中的 $\boldsymbol{P}(k)$ 得到最小值的滤波增益 $\boldsymbol{K}(k)$，如果与式（3-38）、式（3-39）中的 $\boldsymbol{K}(k)$ 相同就回答了这个问题。

令式（3-40） $\boldsymbol{P}(k)$ 求偏导得到的式子等于 0，即

$$\frac{\partial \boldsymbol{P}(k)}{\partial \boldsymbol{K}(k)} = 2[\boldsymbol{I} - \boldsymbol{K}(k)\boldsymbol{H}(k)]\boldsymbol{P}(k-1)[-\boldsymbol{H}^{\mathrm{T}}(k)] + 2\boldsymbol{K}(k)\boldsymbol{R}(k) = 0$$

经整理后得到关于 $\boldsymbol{K}(k)$ 的公式如下：

$$\boldsymbol{K}(k) = \boldsymbol{P}(k-1)\boldsymbol{H}^{\mathrm{T}}(k)[\boldsymbol{R}(k) + \boldsymbol{H}(k)\boldsymbol{P}(k-1)\boldsymbol{H}^{\mathrm{T}}(k)]^{-1} \tag{3-41}$$

这个答案初看起来令人惊讶，和已经得到的滤波增益 $\boldsymbol{K}(k) = \boldsymbol{M}_k\boldsymbol{H}^{\mathrm{T}}(k)\boldsymbol{R}^{-1}(k)$ 是不同的，仔细观察式（3-38）和式（3-41）就会发现，这两个公式看起来是不可能一样的，因为式（3-38）有 $\boldsymbol{M}_k$，但式（3-41）中包含了另一个量——前一步估计方差 $\boldsymbol{P}(k-1)$。但下面 2 个结论（证明方法见附录）却表明这两个公式是等价的。

（1）若考虑 $\boldsymbol{W}(k) = \boldsymbol{R}^{-1}(k)$，则式（3-38）中的 $\boldsymbol{M}_k$ 实际上是估计方差 $\boldsymbol{P}(k)$，按照式（3-38），估计方差也满足 $\boldsymbol{P}^{-1}(k) = \boldsymbol{P}^{-1}(k-1) + \boldsymbol{H}^{\mathrm{T}}(k)\boldsymbol{R}^{-1}(k)\boldsymbol{H}(k)$。

（2）若用 $\boldsymbol{P}(k)$ 来代替式（3-38）中的 $\boldsymbol{M}_k$，则与式（3-41）所表达的滤波增益是等价的。

已知式（3-41）所表达的滤波增益是能够使估计方差式（3-40）最小的滤波增益，也就是最优估计的滤波器增益。而 $\boldsymbol{K}(k) = \boldsymbol{P}(k)\boldsymbol{H}^{\mathrm{T}}(k)\boldsymbol{R}^{-1}(k)$ 是最小二乘方法的滤波器增益，若这两者等价，就可以得出这样的结论：最小二乘方法式（3-38）、式（3-39）是最优的。

到这里，可以得到如下结论，利用最小二乘性能指标得到的估计方法：

$$\hat{\boldsymbol{\theta}}(k) = \hat{\boldsymbol{\theta}}(k-1) + \boldsymbol{K}(k)[Z(k) - \boldsymbol{H}(k)\hat{\boldsymbol{\theta}}(k-1)] \tag{3-42}$$

可以得到最小的估计方差 $\boldsymbol{P}(k)$，其中滤波增益 $\boldsymbol{K}(k)$ 有以下两种不同的计算方法：

$$\boldsymbol{K}(k) = \boldsymbol{P}(k)\boldsymbol{H}^{\mathrm{T}}(k)\boldsymbol{R}^{-1}(k) \tag{3-43}$$

$$\boldsymbol{K}(k) = \boldsymbol{P}(k-1)\boldsymbol{H}^{\mathrm{T}}(k)[\boldsymbol{R}(k) + \boldsymbol{H}(k)\boldsymbol{P}(k-1)\boldsymbol{H}^{\mathrm{T}}(k)]^{-1} \tag{3-44}$$

估计方差 $\boldsymbol{P}(k)$ 可以使用如下两种求法来计算：

$$\boldsymbol{P}(k) = [\boldsymbol{P}^{-1}(k-1) + \boldsymbol{H}^{\mathrm{T}}(k)\boldsymbol{R}^{-1}(k)\boldsymbol{H}(k)]^{-1} \tag{3-45}$$

$$\boldsymbol{P}(k) = [\boldsymbol{I} - \boldsymbol{K}(k)\boldsymbol{H}(k)]\boldsymbol{P}(k-1)[\boldsymbol{I} - \boldsymbol{K}(k)\boldsymbol{H}(k)]^{\mathrm{T}} + \boldsymbol{K}(k)\boldsymbol{R}(k)\boldsymbol{K}^{\mathrm{T}}(k) \tag{3-46}$$

在附录中式（3-52）还提到下面的估计方差 $\boldsymbol{P}(k)$ 计算方法：

$$\boldsymbol{P}(k) = [\boldsymbol{I} - \boldsymbol{K}(k)\boldsymbol{H}(k)]\boldsymbol{P}(k-1) \tag{3-47}$$

在上面的三个计算 $\boldsymbol{P}(k)$ 的公式中，式（3-47）是一个比较简单的表达式，但是数值计算问题可能导致 $\boldsymbol{P}(k)$ 不是正定的，即使 $\boldsymbol{P}(k-1)$ 和 $\boldsymbol{R}(k)$ 都是正定的。式（3-46）虽然复杂，但是却可以保证 $\boldsymbol{P}(k)$ 是正定的，与理论解一致，而式（3-45）需要三次矩阵求逆计算，因此计算复杂，需要占用更多的计算资源，但在多传感器状态融合估计中可以使融合公式具有统一形式，所以经常在状态融合估计方法中使用。

3.5　本 章 小 结

本章介绍了最小二乘估计方法，包括已知测量噪声方差的加权最小二乘方法和递推的最小二乘方法。通过仿真实例发现，已知系统的测量模型很重要，当系统的待估计值是恒值时，最小二乘方法可以得到很好的效果。

本章还证明了最小二乘估计可以得到最优的估计结果，并且给出了两种滤波增益 $\boldsymbol{K}(k)$及三种估计方差 $\boldsymbol{P}(k)$的计算方法，不同的方法虽然形式上看起来不同，但在理论上是完全相同的。然而，有时候计算机会“骗”我们，它的舍入误差可能会导致数值解与理论解不同。

补充知识

1. 矩阵相关公式

设 $\boldsymbol{x}\in\boldsymbol{F}^{n\times n}$ 为常矩阵，有 $\boldsymbol{f}=\boldsymbol{x}^{\mathrm{T}}\boldsymbol{A}\boldsymbol{x}$，则 $\dfrac{\mathrm{d}\boldsymbol{f}}{\mathrm{d}\boldsymbol{x}}=(\boldsymbol{A}+\boldsymbol{A}^{\mathrm{T}})\boldsymbol{x}$。

证明　根据乘法公式：

$$\frac{\mathrm{d}\boldsymbol{f}}{\mathrm{d}\boldsymbol{x}}=\frac{\mathrm{d}}{\mathrm{d}\boldsymbol{x}}(\boldsymbol{x}^{\mathrm{T}}\boldsymbol{A}\boldsymbol{x})=\frac{\mathrm{d}\boldsymbol{x}^{\mathrm{T}}}{\mathrm{d}\boldsymbol{x}}\boldsymbol{A}\boldsymbol{x}+\frac{\mathrm{d}(\boldsymbol{A}\boldsymbol{x})^{\mathrm{T}}}{\mathrm{d}\boldsymbol{x}}\boldsymbol{x}=\boldsymbol{A}\boldsymbol{x}+\boldsymbol{A}^{\mathrm{T}}\boldsymbol{x}=(\boldsymbol{A}+\boldsymbol{A}^{\mathrm{T}})\boldsymbol{x}$$

2. 矩阵逆运算的求法

$$\begin{cases}(\boldsymbol{A}-\boldsymbol{B}\boldsymbol{D}^{-1}\boldsymbol{C})^{-1}=\boldsymbol{A}^{-1}+\boldsymbol{A}^{-1}\boldsymbol{B}(\boldsymbol{D}-\boldsymbol{C}\boldsymbol{A}^{-1}\boldsymbol{B})^{-1}\boldsymbol{C}\boldsymbol{A}^{-1}\\(\boldsymbol{A}+\boldsymbol{B}\boldsymbol{D}^{-1}\boldsymbol{C})^{-1}=\boldsymbol{A}^{-1}-\boldsymbol{A}^{-1}\boldsymbol{B}(\boldsymbol{D}+\boldsymbol{C}\boldsymbol{A}^{-1}\boldsymbol{B})^{-1}\boldsymbol{C}\boldsymbol{A}^{-1}\end{cases}\tag{3-48}$$

3. 矩阵转置运算计算

$$(\boldsymbol{A}+\boldsymbol{B})^{\mathrm{T}}=\boldsymbol{A}^{\mathrm{T}}+\boldsymbol{B}^{\mathrm{T}}$$

$$(\boldsymbol{A}\boldsymbol{B})^{\mathrm{T}}=\boldsymbol{B}^{\mathrm{T}}\boldsymbol{A}^{\mathrm{T}}$$

附录证明

（1）如果考虑 $\boldsymbol{W}(k)=\boldsymbol{R}^{-1}(k)$，式（3-38）中的 $\boldsymbol{M}_k$ 实际上是估计方差 $\boldsymbol{P}(k)$。

证明　先设

$$\boldsymbol{S}(k)=\boldsymbol{R}(k)+\boldsymbol{H}(k)\boldsymbol{P}(k-1)\boldsymbol{H}^{\mathrm{T}}(k)\tag{3-49}$$

则式（3-41）中的滤波增益变为

$$\boldsymbol{K}(k)=\boldsymbol{P}(k-1)\boldsymbol{H}^{\mathrm{T}}(k)\boldsymbol{S}^{-1}(k) \tag{3-50}$$

将其代入式（3-40）并展开，得

$$\begin{aligned}\boldsymbol{P}(k)=&\boldsymbol{P}(k-1)-\boldsymbol{P}(k-1)\boldsymbol{H}^{\mathrm{T}}(k)\boldsymbol{S}^{-1}(k)\boldsymbol{H}(k)\boldsymbol{P}(k-1)-\boldsymbol{P}(k-1)\boldsymbol{H}^{\mathrm{T}}(k)\boldsymbol{S}^{-1}(k)\boldsymbol{H}(k)\boldsymbol{P}(k-1)\\&+\boldsymbol{P}(k-1)\boldsymbol{H}^{\mathrm{T}}(k)\boldsymbol{S}^{-1}(k)\boldsymbol{H}(k)\boldsymbol{P}(k-1)\boldsymbol{H}^{\mathrm{T}}(k)\boldsymbol{S}^{-1}(k)\boldsymbol{H}(k)\boldsymbol{P}(k-1)\\&+\boldsymbol{P}(k-1)\boldsymbol{H}^{\mathrm{T}}(k)\boldsymbol{S}^{-1}(k)\boldsymbol{R}(k)\boldsymbol{S}^{-1}(k)\boldsymbol{H}(k)\boldsymbol{P}(k-1)\end{aligned}$$

将上式第 2、3 两项合并，第 4、5 两项合并后，得到

$$\begin{aligned}\boldsymbol{P}(k)=&\boldsymbol{P}(k-1)-2\boldsymbol{P}(k-1)\boldsymbol{H}^{\mathrm{T}}(k)\boldsymbol{S}^{-1}(k)\boldsymbol{H}(k)\boldsymbol{P}(k-1)\\&+\boldsymbol{P}(k-1)\boldsymbol{H}^{\mathrm{T}}(k)\boldsymbol{S}^{-1}(k)[\boldsymbol{H}(k)\boldsymbol{P}(k-1)\boldsymbol{H}^{\mathrm{T}}(k)+\boldsymbol{R}(k)]\boldsymbol{S}^{-1}(k)\boldsymbol{H}(k)\boldsymbol{P}(k-1)\end{aligned}$$

发现最后一项中间的求逆部分即为 $\boldsymbol{S}(k)$（用横线标出），它和 $\boldsymbol{S}(k)$ 前后两个 $\boldsymbol{S}^{-1}(k)$ 中的一个相乘，从而消去其中一个，进一步化解最后一项得

$$\begin{aligned}\boldsymbol{P}(k)=&\boldsymbol{P}(k-1)-2\boldsymbol{P}(k-1)\boldsymbol{H}^{\mathrm{T}}(k)\boldsymbol{S}^{-1}(k)\boldsymbol{H}(k)\boldsymbol{P}(k-1)\\&+\boldsymbol{P}(k-1)\boldsymbol{H}^{\mathrm{T}}(k)\boldsymbol{S}^{-1}(k)\boldsymbol{S}(k)\boldsymbol{S}^{-1}(k)\boldsymbol{H}(k)\boldsymbol{P}(k-1)\\=&\boldsymbol{P}(k-1)-2\boldsymbol{P}(k-1)\boldsymbol{H}^{\mathrm{T}}(k)\boldsymbol{S}^{-1}(k)\boldsymbol{H}(k)\boldsymbol{P}(k-1)\\&+\boldsymbol{P}(k-1)\boldsymbol{H}^{\mathrm{T}}(k)\boldsymbol{S}^{-1}(k)\boldsymbol{H}(k)\boldsymbol{P}(k-1)\end{aligned}$$

再合并后两项，得到

$$\boldsymbol{P}(k)=\boldsymbol{P}(k-1)-\underline{\boldsymbol{P}(k-1)\boldsymbol{H}^{\mathrm{T}}(k)\boldsymbol{S}^{-1}(k)}\boldsymbol{H}(k)\boldsymbol{P}(k-1) \tag{3-51}$$

现在注意到式（3-50）中的 $\boldsymbol{K}(k)$ 表达式隐藏在式（3-51）（用横线标出）中，所以可以把式（3-51）重新写一下，为

$$\begin{aligned}\boldsymbol{P}(k)=&\boldsymbol{P}(k-1)-\boldsymbol{K}(k)\boldsymbol{H}(k)\boldsymbol{P}(k-1)\\=&[\boldsymbol{I}-\boldsymbol{K}(k)\boldsymbol{H}(k)]\boldsymbol{P}(k-1)\end{aligned} \tag{3-52}$$

它是 $\boldsymbol{P}(k)$ 的另一个表达式，形式上相对简单，但是因为含有相减的运算，在使用计算机进行数值计算时可能会由于计算机的舍入误差而导致 $\boldsymbol{P}(k)$ 不是正定的，这与理论解是矛盾的，因为从方差的定义式（3-40）可知，$\boldsymbol{P}(k)$ 一定是非负定的。

其实 $\boldsymbol{P}(k)$ 还有第三种表达式，它需要三次求逆矩阵运算，在实际估计时计算较复杂，但会发现要证明的结果。

将 $\boldsymbol{S}(k)=\boldsymbol{R}(k)+\boldsymbol{H}(k)\boldsymbol{P}(k-1)\boldsymbol{H}^{\mathrm{T}}(k)$ 代入式（3-51），得到

$$\boldsymbol{P}(k)=\boldsymbol{P}(k-1)-\boldsymbol{P}(k-1)\boldsymbol{H}^{\mathrm{T}}(k)[\boldsymbol{R}(k)+\boldsymbol{H}(k)\boldsymbol{P}(k-1)\boldsymbol{H}^{\mathrm{T}}(k)]^{-1}\boldsymbol{H}(k)\boldsymbol{P}(k-1)$$

对上式两边求逆，利用补充知识 2 中矩阵的求法（3-48）。

$$(\boldsymbol{A}-\boldsymbol{B}\boldsymbol{D}^{-1}\boldsymbol{C})^{-1}=\boldsymbol{A}^{-1}+\boldsymbol{A}^{-1}\boldsymbol{B}(\boldsymbol{D}-\boldsymbol{C}\boldsymbol{A}^{-1}\boldsymbol{B})^{-1}\boldsymbol{C}\boldsymbol{A}^{-1}$$

先弄清楚式子中 $\boldsymbol{A}$、$\boldsymbol{B}$、$\boldsymbol{C}$、$\boldsymbol{D}$ 所代表的矩阵：

$$\boldsymbol{A}=\boldsymbol{P}(k-1)$$
$$\boldsymbol{B}=\boldsymbol{P}(k-1)\boldsymbol{H}^{\mathrm{T}}(k)$$
$$\boldsymbol{C}=\boldsymbol{R}(k)+\boldsymbol{H}(k)\boldsymbol{P}(k-1)\boldsymbol{H}^{\mathrm{T}}(k)$$
$$\boldsymbol{D}=\boldsymbol{H}(k)\boldsymbol{P}(k-1)$$

再利用补充知识式（3-48）所述的矩阵求逆的方法，得到

$$\begin{aligned}\boldsymbol{P}^{-1}(k)=&\boldsymbol{P}^{-1}(k-1)+\boldsymbol{P}^{-1}(k-1)\boldsymbol{P}(k-1)\boldsymbol{H}^{\mathrm{T}}(k)\\&\times[\boldsymbol{R}(k)+\boldsymbol{H}(k)\boldsymbol{P}(k-1)\boldsymbol{H}^{\mathrm{T}}(k)-\boldsymbol{H}(k)\boldsymbol{P}(k-1)\boldsymbol{P}^{-1}(k-1)\boldsymbol{P}(k-1)\boldsymbol{H}^{\mathrm{T}}(k)]^{-1}\\&\times\boldsymbol{H}(k)\boldsymbol{P}(k-1)\boldsymbol{P}^{-1}(k-1)\end{aligned}$$

即

$$\boldsymbol{P}^{-1}(k)=\boldsymbol{P}^{-1}(k-1)+\boldsymbol{H}^{\mathrm{T}}(k)\boldsymbol{R}^{-1}(k)\boldsymbol{H}(k)\tag{3-53}$$

比较式（3-38）和式（3-53）发现，若考虑 $\boldsymbol{W}(k)=\boldsymbol{R}^{-1}(k)$，在初值 $\boldsymbol{M}_0$ 和 $\boldsymbol{P}(0)$ 相同的条件下，$\boldsymbol{M}_k$ 和 $\boldsymbol{P}(k)$ 在每一个 k 值的迭代结果都是相同的，因此二者是等价的，也就是 $\boldsymbol{M}_k$ 和 $\boldsymbol{P}(k)$ 是相同的。

（2）如果用 $\boldsymbol{P}(k)$ 来代替式（3-38）中的 $\boldsymbol{M}_k$，则式（3-41）所表达的滤波增益 $\boldsymbol{K}(k)=\boldsymbol{P}(k-1)\boldsymbol{H}^{\mathrm{T}}(k)[\boldsymbol{R}(k)+\boldsymbol{H}(k)\boldsymbol{P}(k-1)\boldsymbol{H}^{\mathrm{T}}(k)]^{-1}$ 和 $\boldsymbol{K}(k)=\boldsymbol{P}(k)\boldsymbol{H}^{\mathrm{T}}(k)\boldsymbol{R}^{-1}(k)$ 是等价的。

证明　首先利用 $\boldsymbol{P}(k)\boldsymbol{P}^{-1}(k)=\boldsymbol{I}$ 左乘式（3-41）等号右边多项式，得到

$$\boldsymbol{K}(k)=\boldsymbol{P}(k)\boldsymbol{P}^{-1}(k)\boldsymbol{P}(k-1)\boldsymbol{H}^{\mathrm{T}}(k)[\boldsymbol{R}(k)+\boldsymbol{H}(k)\boldsymbol{P}(k-1)\boldsymbol{H}^{\mathrm{T}}(k)]^{-1}$$

将式（3-45）代入上式后，可以得到下面的式子：

$$\begin{aligned}\boldsymbol{K}(k)=&\boldsymbol{P}(k)[\boldsymbol{P}^{-1}(k-1)+\boldsymbol{H}^{\mathrm{T}}(k)\boldsymbol{R}^{-1}(k)\boldsymbol{H}(k)]\\&\times\boldsymbol{P}(k-1)\boldsymbol{H}^{\mathrm{T}}(k)[\boldsymbol{R}(k)+\boldsymbol{H}(k)\boldsymbol{P}(k-1)\boldsymbol{H}^{\mathrm{T}}(k)]^{-1}\end{aligned}$$

注意到 $\boldsymbol{P}(k-1)\boldsymbol{H}^{\mathrm{T}}(k)$ 在第一个具有加和关系小括号的右边，可以将它乘入第一个小括号 $[\boldsymbol{P}^{-1}(k-1)+\boldsymbol{H}^{\mathrm{T}}(k)\boldsymbol{R}^{-1}(k)\boldsymbol{H}(k)]$ 内，得到

$$\begin{aligned}\boldsymbol{K}(k)=&\boldsymbol{P}(k)[\boldsymbol{H}^{\mathrm{T}}(k)+\boldsymbol{H}^{\mathrm{T}}(k)\boldsymbol{R}^{-1}(k)\boldsymbol{H}(k)\boldsymbol{P}(k-1)\boldsymbol{H}^{\mathrm{T}}(k)]\\&\times[\boldsymbol{R}(k)+\boldsymbol{H}(k)\boldsymbol{P}(k-1)\boldsymbol{H}^{\mathrm{T}}(k)]^{-1}\end{aligned}$$

现在把第一个加和关系括号里面的 $\boldsymbol{H}^{\mathrm{T}}(k)$ 向左提出来，得到

$$\begin{aligned}\boldsymbol{K}(k)=&\boldsymbol{P}(k)\boldsymbol{H}^{\mathrm{T}}(k)[\boldsymbol{I}+\boldsymbol{R}^{-1}(k)\boldsymbol{H}(k)\boldsymbol{P}(k-1)\boldsymbol{H}^{\mathrm{T}}(k)]\\&\times[\boldsymbol{R}(k)+\boldsymbol{H}(k)\boldsymbol{P}(k-1)\boldsymbol{H}^{\mathrm{T}}(k)]^{-1}\end{aligned}$$

再在第一个加和关系括号里面的 $\boldsymbol{R}^{-1}(k)$ 向左提出来，得到

$$\boldsymbol{K}(k)=\boldsymbol{P}(k)\boldsymbol{H}^{\mathrm{T}}(k)\boldsymbol{R}^{-1}(k)[\boldsymbol{R}(k)+\boldsymbol{H}(k)\boldsymbol{P}(k-1)\boldsymbol{H}^{\mathrm{T}}(k)]$$
$$\times[\boldsymbol{R}(k)+\boldsymbol{H}(k)\boldsymbol{P}(k-1)\boldsymbol{H}^{\mathrm{T}}(k)]^{-1}$$

由于后两个括号是求逆的关系，则最终得到

$$\boldsymbol{K}(k)=\boldsymbol{P}(k)\boldsymbol{H}^{\mathrm{T}}(k)\boldsymbol{R}^{-1}(K) \tag{3-54}$$

可见，考虑到 $\boldsymbol{M}_k$ 和 $\boldsymbol{P}(k)$ 是相同的，式（3-41）所表达的滤波增益：

$\boldsymbol{K}(k)=\boldsymbol{P}(k-1)\boldsymbol{H}^{\mathrm{T}}(k)[\boldsymbol{R}(k)+\boldsymbol{H}(k)\boldsymbol{P}(k-1)\boldsymbol{H}^{\mathrm{T}}(k)]^{-1}$ 和 $\boldsymbol{K}(k)=\boldsymbol{P}(k)\boldsymbol{H}^{\mathrm{T}}(k)\boldsymbol{R}^{-1}(k)$ 是等价的。

第 4 章　Kalman 滤波器

跟其他著名的理论（如傅里叶变换、泰勒级数等）一样，Kalman 滤波器也是以一个人的名字命名的，即匈牙利数学家 Rudolf Emil Kalman。现在要学习的 Kalman 滤波器，源于 Rudolf Emil Kalman 的博士论文以及 1960 年左右发表的论文。简单说，Kalman 滤波器是一个“optimal recursive data processing algorithm”（最优自回归数据处理算法）。对于很多问题，Kalman 滤波器都可以得到最优的（最小均方差）估计结果，效率最高甚至是最有用的。目前，Kalman 滤波器在许多领域具有广泛应用，包括机器人导航与控制、雷达跟踪系统等。近年来还被应用于计算机图像处理，如视频图像跟踪等。

Kalman 滤波器的推导有很多方法，采用最小二乘估计方法进行推导，先给出系统模型，再考虑由待估计状态变化引起的向前一步状态预测问题，进而给出 Kalman 滤波器五个著名的递推公式。

例 4.1　下面来看一下如果实际测量数据发生突变，利用最小二乘方法得到的结果如何呢？真实值在第 501 个点处由 20 跳变至 30，如图 4-1 所示，噪声方差不变，仍为 2。

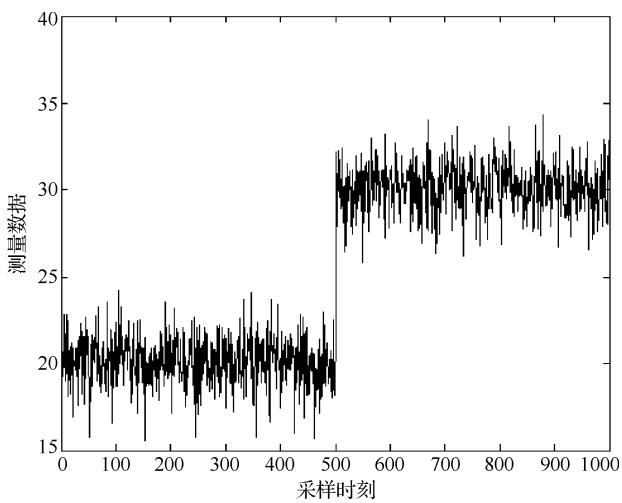


图 4-1　传感器得到的具有突变特征的数据

测量方程为 $Z(k)=\boldsymbol{\theta}+N(k)$，测量矩阵为 $\boldsymbol{H}(k)=1$，测量噪声方差为 $\boldsymbol{R}(k)=2$，因为测量矩阵是一个标量，权值也是一个标量，取为测量噪声方差的倒数 0.5，$\boldsymbol{H}^{\mathrm{T}}(k)\boldsymbol{W}(k)\boldsymbol{H}(k)$ 计算后也是一个标量，所以 $\boldsymbol{M}_k$ 也是一个标量，其初始值设为 1000。

首先需要用模拟的方法产生具有突变特征的传感器数据，前 500 个数据的真值为 20，后 500 个数据的真值为 30，传感器测量噪声的方差为 2，程序如下：

```
clc
clear
t=0.01:0.01:10;
za=[20+sqrt(2)*randn(500,1);30+sqrt(2)*randn(500,1)];
```

接下来利用递推最小二乘方法估计具有突变特征的数据真值，程序中 h 表示 $\boldsymbol{H}(k)$，w 表示 $\boldsymbol{W}(k)$，M 表示 $\boldsymbol{M}_k$，ea0 为待估计量的初值，在实现递推估计的循环中第 i 次的估计结果用 ea(i)表示，如图 4-2 所示，程序如下：

```
h=1;
w=0.5;
M=1000;
ea0=0;
for i=1:1000
M=inv(inv(M)+h'*w*h);
ea(i)=ea0+M*h'*w*(za(i)-h*ea0);
ea0=ea(i);
end
plot(ea)
```

可以看到，具有递推效果的最小二乘估计方法能够发现真实值的变化，并将这种变化反映在估计值中。但经过后 500 个递推估计，还是没有得到较为准确的估计（真实值是 30）。

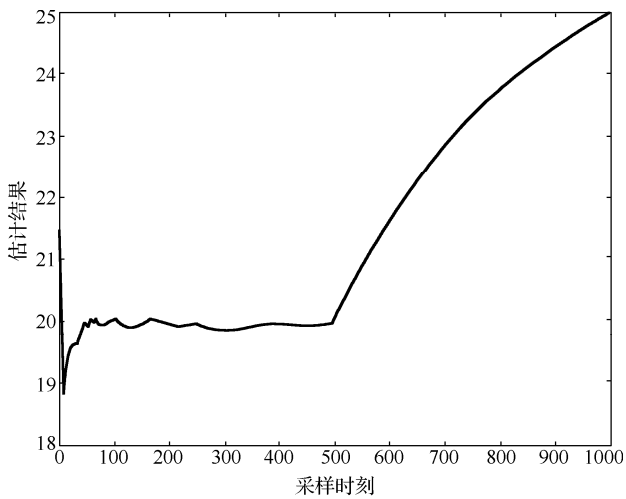


图 4-2　具有突变特征变量的估计结果

如果真实值的变化再复杂一些，例如，很多信号的变化都满足一阶马尔可夫过程

$\boldsymbol{\theta}(k+1)=p\boldsymbol{\theta}(k)+\boldsymbol{w}(k)$，其中 p 为任意常数。可以推断其估计的结果会更差。为什么会这样呢？回忆一下，最小二乘估计所研究的问题只给出了测量方程，描述的是测量数据和待估计量的真值之间的线性关系，而且，由于认为被估计量$\boldsymbol{\theta}$恒定不变，那么使用递推最小二乘估计方法，其实也是默认待估计量并不改变，只是在这个过程中估计结果随着时间的变化不断向真值靠近。所以在实际应用的时候，如果待估计量发生了改变，不考虑这种变化就不会得到很好的结果。那么如何来解决这个问题呢？本章 Kalman 滤波器的内容表明，该方法就是研究待估计量随着时间而改变情况下的估计。

4.1　系统模型描述

第 3 章假设待估计变量$\boldsymbol{\theta}$不变，与之不同的是，本章将认为待估计量 $\boldsymbol{x}$ 随着时间发生变化。为与第 3 章待估计量$\boldsymbol{\theta}$相区别，本章使用 $\boldsymbol{x}$ 表示待估计量，写成离散形式是 $\boldsymbol{x}(k)$，并且，$\boldsymbol{x}(k)$ 含有不确定的变化。估计含有噪声变量的方法叫做贝叶斯估计，Kalman 滤波器也可以通过贝叶斯估计推导得到，本书使用第 3 章最小二乘估计的结果来推导 Kalman 滤波器，这样更简洁，更易于理解。

假设有线性离散系统的过程模型及测量模型如下：

$$\boldsymbol{x}(k+1)=\boldsymbol{A}(k)\boldsymbol{x}(k)+\boldsymbol{w}(k) \tag{4-1}$$

$$\boldsymbol{z}(k)=\boldsymbol{C}(k)\boldsymbol{x}(k)+\boldsymbol{v}(k) \tag{4-2}$$

式中，$\boldsymbol{x}(k)$ 是待估计量，$\boldsymbol{z}(k)$ 是通过传感器得到的测量数据。一般将式（4-1）称为系统过程模型，指的是系统中待估计状态随时间变化的规律。前一时刻 k 的状态 $\boldsymbol{x}(k)$，在下一时刻即 $k+1$ 时刻会变成 $\boldsymbol{x}(k+1)$，称 $\boldsymbol{A}(k)$ 为过程矩阵，表示状态变换的关系。$\boldsymbol{w}(k)$ 叫做过程噪声，表示 $\boldsymbol{x}(k)$ 变成 $\boldsymbol{x}(k+1)$ 过程中的不确定程度。式（4-2）称为测量模型，$\boldsymbol{C}(k)$ 为测量矩阵，$\boldsymbol{v}(k)$ 为测量噪声。假设 $\boldsymbol{w}(k)$ 和 $\boldsymbol{v}(k)$ 是零均值、不相关白噪声，并且协方差矩阵已知，分别用 $\boldsymbol{Q}(k)$ 和 $\boldsymbol{R}(k)$ 表示，即

$$\begin{cases}\boldsymbol{w}(k)\sim(0,\boldsymbol{Q}(k))\\ \boldsymbol{v}(k)\sim(0,\boldsymbol{R}(k))\\ \boldsymbol{E}[\boldsymbol{w}(k)\boldsymbol{w}^{\mathrm{T}}(j)]=\boldsymbol{Q}(k)\delta(k-j)\\ \boldsymbol{E}[\boldsymbol{v}(k)\boldsymbol{v}^{\mathrm{T}}(j)]=\boldsymbol{R}(k)\delta(k-j)\\ \boldsymbol{E}[\boldsymbol{w}(k)\boldsymbol{v}^{\mathrm{T}}(j)]=0\end{cases} \tag{4-3}$$

式中，$\delta(k-j)$ 是 Kronecker-δ 函数，即如果 $k=j$，那么 $\delta(k-j)=1$，如果 $k\neq j$，那么 $\delta(k-j)=0$。这样做的目的是，在已知系统方程式（4-1）、式（4-2）和含噪声的测量 $\boldsymbol{z}(k)(k=1,2,3\cdots)$ 基础上估计状态 $\boldsymbol{x}(k)$。状态量的具体物理含义取决于要研究的问题，如果关注的是跟踪目标问题，状态量一般取位移、速度和加速度等。

接下来介绍两个量的表示方法：一个是后验估计 $\hat{\boldsymbol{x}}(k\,|\,k)$；另一个是先验估计

$\hat{\boldsymbol{x}}(k|k-1)$。二者的区别在于测量与待估计量之间的时间关系。$\hat{\boldsymbol{x}}(k|k)$ 是使用 k 时刻及其以前各个时刻的测量值 $\boldsymbol{z}(k)$ 估计 k 时刻的状态 $\boldsymbol{x}(k)$ 期望值，即 $\hat{\boldsymbol{x}}(k|k)=E[\boldsymbol{x}(k)|\boldsymbol{z}(1),\boldsymbol{z}(2),\cdots,\boldsymbol{z}(k)]$，而 $\hat{\boldsymbol{x}}(k|k-1)$ 是使用 k 时刻以前的各个时刻（注意：不包括 k 时刻）的测量值，来估计 k 时刻的状态 $\boldsymbol{x}(k)$ 期望值，即 $\hat{\boldsymbol{x}}(k|k-1)=\boldsymbol{E}[\boldsymbol{x}(k)|\boldsymbol{z}(1),\boldsymbol{z}(2),\cdots,\boldsymbol{z}(k-1)]$，注意到，先验估计 $\hat{\boldsymbol{x}}(k|k-1)$ 和后验估计 $\hat{\boldsymbol{x}}(k|k)$ 都是同一个量的 $\boldsymbol{x}(k)$ 估计，然而 $\hat{\boldsymbol{x}}(k|k-1)$ 考虑的是测量值 $\boldsymbol{z}(k)$ 之前的估计，也称为向前一步预测估计，而 $\hat{\boldsymbol{x}}(k|k)$ 考虑的是测量值 $\boldsymbol{z}(k)$ 之后的估计。可见需要用更多的信息计算 $\hat{\boldsymbol{x}}(k|k)$，所以自然希望 $\hat{\boldsymbol{x}}(k|k)$ 比 $\hat{\boldsymbol{x}}(k|k-1)$ 更好。那么应该设计什么估计方法来得到 $\hat{\boldsymbol{x}}(k|k)$ 和 $\hat{\boldsymbol{x}}(k|k-1)$ 呢？$\hat{\boldsymbol{x}}(k|k)$ 的估计是不是比 $\hat{\boldsymbol{x}}(k|k-1)$ 更好呢？下面就来回答这两个问题。

4.2　向前一步预测估计 $\hat{\boldsymbol{x}}(k|k-1)$ 的求法

重新考虑第 3 章最小二乘估计的问题，已知含有噪声测量，使用式（3-42）～式（3-46）可以得到最优估计。但第 3 章假设待估计量不随时间而改变，而本章研究的待估计量 $\boldsymbol{x}(k)$ 随着时间的推移按照式（4-1）而改变，也就是说，在 k 时刻的测量 $\boldsymbol{z}(k)$ 到来之前，状态 $\boldsymbol{x}(k)$ 在 $k-1$ 时刻的估计值 $\hat{\boldsymbol{x}}(k-1|k-1)$ 会随着时间的推移而改变，在 $\boldsymbol{z}(k)$ 到来时，$\hat{\boldsymbol{x}}(k-1|k-1)$ 已经变化为 $\hat{\boldsymbol{x}}(k|k-1)$，改变遵循式（4-1）系统状态转移方程：

$$\hat{\boldsymbol{x}}(k|k-1)=\boldsymbol{A}(k-1)\hat{\boldsymbol{x}}(k-1|k-1) \tag{4-4}$$

如果已知 $\hat{\boldsymbol{x}}(k-1|k-1)$ 的估计方差：

$$\boldsymbol{P}(k-1|k-1)=E[(\boldsymbol{x}(k-1)-\hat{\boldsymbol{x}}(k-1|k-1))(\boldsymbol{x}(k-1)-\hat{\boldsymbol{x}}(k-1|k-1))^{\mathrm{T}}] \tag{4-5}$$

那么 $\hat{\boldsymbol{x}}(k|k-1)$ 的估计方差 $\boldsymbol{P}(k|k-1)$ 是多少呢？$\boldsymbol{P}(k|k-1)$ 的定义式应为

$$\boldsymbol{P}(k|k-1)=E[(\boldsymbol{x}(k)-\hat{\boldsymbol{x}}(k|k-1))(\boldsymbol{x}(k)-\hat{\boldsymbol{x}}(k|k-1))^{\mathrm{T}}] \tag{4-6}$$

将式（4-1）和式（4-4）代入式（4-6），得到

$$\begin{aligned}\boldsymbol{P}(k|k-1)=&E[(\boldsymbol{A}(k-1)\boldsymbol{x}(k-1)+\boldsymbol{w}(k-1)-\boldsymbol{A}(k-1)\hat{\boldsymbol{x}}(k-1|k-1))\\&\times(\boldsymbol{A}(k-1)\boldsymbol{x}(k-1)+\boldsymbol{w}(k-1)-\boldsymbol{A}(k-1)\hat{\boldsymbol{x}}(k-1|k-1))^{\mathrm{T}}]\\=&E[(\boldsymbol{A}(k-1)(\boldsymbol{x}(k-1)-\hat{\boldsymbol{x}}(k-1|k-1))+\boldsymbol{w}(k-1))(\boldsymbol{A}(k-1)\\&(\boldsymbol{x}(k-1)-\hat{\boldsymbol{x}}(k-1|k-1))+\boldsymbol{w}(k-1))^{\mathrm{T}}]\end{aligned}$$

考虑到状态与过程噪声不相关，将上式两个括号相乘，得到

$$\begin{aligned}\boldsymbol{P}(k|k-1)=&\boldsymbol{A}(k-1)E[(\boldsymbol{x}(k-1)-\hat{\boldsymbol{x}}(k-1|k-1))(\boldsymbol{x}(k-1)-\hat{\boldsymbol{x}}(k-1|k-1))^{\mathrm{T}}]\boldsymbol{A}^{\mathrm{T}}(k-1)\\&+E[\boldsymbol{w}(k-1)\boldsymbol{w}^{\mathrm{T}}(k-1)]\end{aligned}$$

注意到上式中第一项的中间项为 $\hat{\boldsymbol{x}}(k-1|k-1)$ 的估计方差 $\boldsymbol{P}(k-1|k-1)$，并利用式（4-3）中过程噪声的方差项，可以得到

$$\boldsymbol{P}(k|k-1)=\boldsymbol{A}(k-1)\boldsymbol{P}(k-1|k-1)\boldsymbol{A}^{\mathrm{T}}(k-1)+\boldsymbol{Q}(k-1) \tag{4-7}$$

总结一下，在 $\boldsymbol{z}(k)$ 到来时，$\hat{\boldsymbol{x}}(k-1|k-1)$ 已经变化为 $\hat{\boldsymbol{x}}(k|k-1)$，其估计方差也由 $\boldsymbol{P}(k-1|k-1)$ 转变为 $\boldsymbol{P}(k|k-1)$，为读者阅读方便，将其转换公式重写如下：

$$\hat{\boldsymbol{x}}(k|k-1)=\boldsymbol{A}(k-1)\hat{\boldsymbol{x}}(k-1|k-1) \tag{4-8}$$

$$\boldsymbol{P}(k|k-1)=\boldsymbol{A}(k-1)\boldsymbol{P}(k-1|k-1)\boldsymbol{A}^{\mathrm{T}}(k-1)+\boldsymbol{Q}(k-1) \tag{4-9}$$

4.3　更新估计 $\hat{\boldsymbol{x}}(k|k)$ 的求法

继续考虑在 $\hat{\boldsymbol{x}}(k|k-1)$ 的情况下，如果又得到了 k 时刻的测量 $\boldsymbol{z}(k)$，如何利用新到的测量值来得到更准确的估计 $\hat{\boldsymbol{x}}(k|k)$ 呢？这时候，就可以很方便地利用第 3 章中学习到的最小二乘估计方法了，在式（3-42）～式（3-47）中，用 $\hat{\boldsymbol{x}}(k|k-1)$ 代替 $\hat{\boldsymbol{\theta}}(k-1)$，$\hat{\boldsymbol{x}}(k|k)$ 代替 $\hat{\boldsymbol{\theta}}(k)$，用 $\boldsymbol{P}(k|k-1)$ 代替 $\boldsymbol{P}(k-1)$，$\boldsymbol{P}(k|k)$ 代替 $\boldsymbol{P}(k)$，对式（3-42）、式（3-44）、式（3-47）按如下形式重写，其余的公式变换与之相似：

$$\hat{\boldsymbol{x}}(k|k)=\hat{\boldsymbol{x}}(k|k-1)+\boldsymbol{K}(k)[\boldsymbol{z}(k)-\boldsymbol{C}(k)\hat{\boldsymbol{x}}(k|k-1)] \tag{4-10}$$

$$\boldsymbol{K}(k)=\boldsymbol{P}(k|k-1)\boldsymbol{C}^{\mathrm{T}}(k)[\boldsymbol{C}^{\mathrm{T}}(k)\boldsymbol{P}(k|k-1)\boldsymbol{C}(k)+\boldsymbol{R}(k)]^{-1} \tag{4-11}$$

$$\boldsymbol{P}(k|k)=[\boldsymbol{I}-\boldsymbol{K}(k)\boldsymbol{C}(k)]\boldsymbol{P}(k|k-1) \tag{4-12}$$

下面来仔细分析一下滤波增益 $\boldsymbol{K}(k)$。用式（4-13）表示预测的测量：

$$\boldsymbol{z}(k|k-1)=\boldsymbol{C}(k)\hat{\boldsymbol{x}}(k|k-1) \tag{4-13}$$

虽然预测的测量与真实测量会有不同，但这个量还是反映了预测状态的结果。设 $\boldsymbol{P}_{xz}$ 定义为状态预测与测量预测的协方差，则

$$\boldsymbol{P}_{xz}=E\left[\left(\left(\boldsymbol{x}(k)-\hat{\boldsymbol{x}}(k|k-1)\right)\left(\boldsymbol{z}(k)-\boldsymbol{z}(k|k-1)\right)^{\mathrm{T}}\right]\right. \tag{4-14}$$

将式（4-2）和式（4-13）代入式（4-14），得

$$\begin{aligned}\boldsymbol{P}_{xz}&=E[(\boldsymbol{x}(k)-\hat{\boldsymbol{x}}(k|k-1))(\boldsymbol{C}(k)\boldsymbol{x}(k)+\boldsymbol{v}(k)-\boldsymbol{C}(k)\hat{\boldsymbol{x}}(k|k-1))^{\mathrm{T}}]\\&=E[(\boldsymbol{x}(k)-\hat{\boldsymbol{x}}(k|k-1))(\boldsymbol{C}(k)(\boldsymbol{x}(k)-\hat{\boldsymbol{x}}(k|k-1))+\boldsymbol{v}(k))^{\mathrm{T}}]\end{aligned}$$

因为 $\boldsymbol{x}(k)$ 与 $\boldsymbol{v}(k)$ 之间不相关，则

$$\boldsymbol{P}_{xz}=\boldsymbol{P}(k|k-1)\boldsymbol{C}^{\mathrm{T}}(k) \tag{4-15}$$

再定义预测测量的方差为

$$\boldsymbol{P}_{zz}=E[(\boldsymbol{z}(k)-\boldsymbol{z}(k|k-1))(\boldsymbol{z}(k)-\boldsymbol{z}(k|k-1))^{\mathrm{T}}] \tag{4-16}$$

同样，将式（4-2）和式（4-13）代入式（4-16）经整理后得

$$\boldsymbol{P}_{zz}=\boldsymbol{C}^{\mathrm{T}}(k)\boldsymbol{P}(k|k-1)\boldsymbol{C}(k)+\boldsymbol{R}(k) \tag{4-17}$$

因此，滤波增益也可以表示为

$$\boldsymbol{K}(k)=\boldsymbol{P}_{xz}\boldsymbol{P}_{zz}^{-1} \tag{4-18}$$

再来看一下式（4-12），将右边等式展开得

$$\begin{aligned}\boldsymbol{P}(k\mid k)&=\boldsymbol{P}(k\mid k-1)-\boldsymbol{K}(k)\boldsymbol{C}(k)\boldsymbol{P}(k\mid k-1)\\&=\boldsymbol{P}(k\mid k-1)-\boldsymbol{K}(k)\boldsymbol{P}_{xz}^{\mathrm{T}}\end{aligned} \tag{4-19}$$

式（4-19）考虑到 $\boldsymbol{P}(k\mid k-1)=\boldsymbol{P}^{\mathrm{T}}(k\mid k-1)$，即 $\boldsymbol{P}(k\mid k-1)$ 为对称矩阵。又在等式（4-18）两边右乘 $\boldsymbol{P}_{zz}$ 得

$$\boldsymbol{P}_{xz}=\boldsymbol{K}(k)\boldsymbol{P}_{zz} \tag{4-20}$$

将式（4-20）代入式（4-19），还可以得到 $\boldsymbol{P}(k\mid k)$ 的另一个表达式：

$$\boldsymbol{P}(k\mid k)=\boldsymbol{P}(k\mid k-1)-\boldsymbol{K}(k)\boldsymbol{P}_{zz}\boldsymbol{K}^{\mathrm{T}}(k) \tag{4-21}$$

式(4-18)、式(4-19)、式(4-21)是利用 $\boldsymbol{P}_{xz}$、$\boldsymbol{P}_{zz}$ 求增益 $\boldsymbol{K}(k)$ 及状态估计方差 $\boldsymbol{P}(k\mid k)$ 的方法。有时通过贝叶斯估计的方法获得 $\boldsymbol{P}_{xz}$、$\boldsymbol{P}_{zz}$ 后，就可以利用式（4-18）滤波增益 $\boldsymbol{K}(k)$，而相应的估计方差即为式（4-21）或式（4-19）。这个方法在第 5 章的非线性滤波器中会使用。

4.4　离散 Kalman 滤波器

现将离散 Kalman 滤波器总结如下。

（1）系统方程为

$$\boldsymbol{x}(k+1)=\boldsymbol{A}(k)\boldsymbol{x}(k)+\boldsymbol{w}(k)$$

$$\begin{cases}\boldsymbol{z}(k)=\boldsymbol{C}(k)\boldsymbol{x}(k)+\boldsymbol{v}(k)\\E[\boldsymbol{w}(k)\boldsymbol{w}^{\mathrm{T}}(j)]=\boldsymbol{Q}(k)\delta(k-j)\\E[\boldsymbol{v}(k)\boldsymbol{v}^{\mathrm{T}}(j)]=\boldsymbol{R}(k)\delta(k-j)\\E[\boldsymbol{w}(k)\boldsymbol{v}^{\mathrm{T}}(j)]=0\end{cases} \tag{4-22}$$

（2）Kalman 滤波器初始化：

$$\begin{aligned}\hat{\boldsymbol{x}}(0\mid 0)&=E[\boldsymbol{x}(0)]\\\boldsymbol{P}(0\mid 0)&=E[(\boldsymbol{x}(0)-\hat{\boldsymbol{x}}(0\mid 0))(\boldsymbol{x}(0)-\hat{\boldsymbol{x}}(0\mid 0))^{\mathrm{T}}]\end{aligned} \tag{4-23}$$

（3）Kalman 滤波器每一步计算如下，其中 $k=1,2,3,\cdots$

$$\hat{\boldsymbol{x}}(k\mid k)=\hat{\boldsymbol{x}}(k\mid k-1)+\boldsymbol{K}(k)[\boldsymbol{z}(k)-\boldsymbol{C}(k)\hat{\boldsymbol{x}}(k\mid k-1)] \tag{4-24}$$

$$\hat{\boldsymbol{x}}(k|k-1)=\boldsymbol{A}(k-1)\hat{\boldsymbol{x}}(k-1|k-1) \tag{4-25}$$

$$\boldsymbol{K}(k)=\boldsymbol{P}(k|k-1)\boldsymbol{C}^{\mathrm{T}}(k)[\boldsymbol{C}^{\mathrm{T}}(k)\boldsymbol{P}(k|k-1)\boldsymbol{C}(k)+\boldsymbol{R}(k)]^{-1} \tag{4-26}$$

$$\boldsymbol{P}(k|k-1)=\boldsymbol{A}(k-1)\boldsymbol{P}(k-1|k-1)\boldsymbol{A}^{\mathrm{T}}(k-1)+\boldsymbol{Q}(k-1) \tag{4-27}$$

$$\boldsymbol{P}(k|k)=[\boldsymbol{I}-\boldsymbol{K}(k)\boldsymbol{C}(k)]\boldsymbol{P}(k|k-1) \tag{4-28}$$

使用下面的 MATLAB 函数实现 Kalman 滤波器，函数输入为系统矩阵的参数，其中 A 表示过程矩阵 $\boldsymbol{A}(k-1)$，C 表示测量矩阵 $\boldsymbol{C}(k)$，Q 表示过程噪声方差 $\boldsymbol{Q}(k-1)$，R 表示测量噪声方差 $\boldsymbol{R}(k)$。函数的变量 xe 表示前一步的状态估计值 $\hat{\boldsymbol{x}}(k-1|k-1)$，p 表示前一步的状态估计方差 $\boldsymbol{P}(k-1|k-1)$，z 表示当前测量值 $\boldsymbol{z}(k)$。函数的输出包括三个量，分别是当前步的状态估计值 $\hat{\boldsymbol{x}}(k|k)$、向前一步的递推状态估计方差 $\boldsymbol{P}(k|k-1)$ 以及当前步的状态估计方差 $\boldsymbol{P}(k|k)$，分别用 xe、p1、pk 表示。另外，使用函数 inv 实现矩阵求逆。

```
function [xe,pk,p1]=kalmanfun(A,C,Q,R,xe,z,p)
%此函数通过 Kalman 滤波器计算状态估计值。
   xe=A*xe; %根据式（4-25）计算向前一步预测 x̂(k|k-1)
   P1=A*p*A'+Q; %根据式（4-27）计算向前一步估计 P(k|k-1)
   K=p1*C'*inv(C*p1*C'+R); %根据式（4-26）计算估计增益 K(k)
   xe=xe+K*(z-C*xe);% 根据式（4-25）计算估计 x̂(k|k)
   pk=(eye(size(p1))-l*C)*p1;   % 根据式（4-28）计算估计方差 P(k|k)
```

第 3 章在推导递推最小二乘法时，增益的方差还有另外两种形式，所以 Kalman 滤波器的增益和估计方差也还有另外两种形式：

$$\boldsymbol{K}(k)=\boldsymbol{P}(k|k)\boldsymbol{C}^{\mathrm{T}}(k)\boldsymbol{R}^{-1}(k) \tag{4-29}$$

和

$$\boldsymbol{P}^{-1}(k|k)=\boldsymbol{P}^{-1}(k|k-1)+\boldsymbol{C}^{\mathrm{T}}(k)\boldsymbol{R}^{-1}(k)\boldsymbol{C}(k) \tag{4-30}$$

计算表明，估计方差 $\boldsymbol{P}(k|k)$ 是一个对称矩阵。从式（4-26）～式（4-28）还可以看出，$\boldsymbol{P}(k|k)$、$\boldsymbol{K}(k)$和 $\boldsymbol{P}(k|k-1)$ 的计算不依赖于测量 $\boldsymbol{z}(k)$，仅依赖于系统参数，因此当系统参数已知时，可以在系统测量到达并估计之前离线地计算。在系统运行时仅计算 $\hat{\boldsymbol{x}}(k|k)$ 和 $\hat{\boldsymbol{x}}(k|k-1)$，如果 Kalman 滤波器在计算量有严格要求的嵌入式系统中运行，这一点决定系统是否能够实时运行。此外，Kalman 滤波器的工作性能可以在实施运行前评估，这是因为代表滤波器的估计的准确度 $\boldsymbol{P}(k|k)$ 也不取决于测量值，可以脱机计算。

当 $k\to\infty$ 时，$\boldsymbol{P}(k|k)$ 的收敛值 $\boldsymbol{P}(\infty|\infty)$ 表示系统的稳态估计方差。先来看一下当 $k\to\infty$ 时，$\boldsymbol{P}(k|k-1)$ 的稳态值，将式（4-27）中的 k 变为 $k+1$，再将式（4-28）代入，并利用式（4-28）的滤波器增益得到

$$\begin{aligned}\boldsymbol{P}(k+1|k) &= \boldsymbol{A}(k)\boldsymbol{P}(k|k)\boldsymbol{A}^{\mathrm{T}}(k)+\boldsymbol{Q}(k)\\ &= \boldsymbol{A}(k)[\boldsymbol{I}-\boldsymbol{K}(k)\boldsymbol{C}(k)]\boldsymbol{P}(k|k-1)\boldsymbol{A}^{\mathrm{T}}(k)+\boldsymbol{Q}(k)\\ &= \boldsymbol{A}(k)\boldsymbol{P}(k|k-1)\boldsymbol{A}^{\mathrm{T}}(k)-\boldsymbol{A}(k)\boldsymbol{K}(k)\boldsymbol{C}(k)\boldsymbol{P}(k|k-1)\boldsymbol{A}^{\mathrm{T}}(k)+\boldsymbol{Q}(k)\\ &= \boldsymbol{A}(k)\boldsymbol{P}(k|k-1)\boldsymbol{A}^{\mathrm{T}}(k)\\ &\quad -\boldsymbol{A}(k)\boldsymbol{P}(k|k-1)\boldsymbol{C}^{\mathrm{T}}(k)[\boldsymbol{R}(k)+\boldsymbol{C}(k)\boldsymbol{P}(k|k-1)\boldsymbol{C}^{\mathrm{T}}(k)]^{-1}\boldsymbol{C}(k)\\ &\quad \boldsymbol{P}(k|k-1)\boldsymbol{A}^{\mathrm{T}}(k)+\boldsymbol{Q}(k)\end{aligned} \tag{4-31}$$

如果协方差矩阵收敛，即 $k\to\infty$， $\boldsymbol{P}(k+1|k)=\boldsymbol{P}(k|k-1)\to\boldsymbol{P}_{\infty}$，式（4-31）变为

$$\boldsymbol{P}_{\infty}=\boldsymbol{A}(k)\boldsymbol{P}_{\infty}\boldsymbol{A}^{\mathrm{T}}(k)-\boldsymbol{A}(k)\boldsymbol{P}_{\infty}\boldsymbol{C}^{\mathrm{T}}(k)[\boldsymbol{R}(k)+\boldsymbol{C}(k)\boldsymbol{P}_{\infty}\boldsymbol{C}^{\mathrm{T}}(k)]^{-1}\boldsymbol{C}(k)\boldsymbol{P}_{\infty}\boldsymbol{A}^{\mathrm{T}}(k)+\boldsymbol{Q}(k) \tag{4-32}$$

这是著名的离散 Riccati 方程，利用该式可以判断估计结果的好坏，还可以计算稳态的估计方差 $\boldsymbol{P}(\infty|\infty)$ 和稳态增益 $\boldsymbol{K}(\infty)$ 来简化 Kalman 滤波器的计算。

例 4.2　假设对于无噪声牛顿动力学系统，位置、速度、恒加速度分别为 r、v、a。这个系统可以描述为

$$\begin{bmatrix}\dot{r}\\ \dot{v}\\ \dot{a}\end{bmatrix}=\begin{bmatrix}0&1&0\\0&0&1\\0&0&0\end{bmatrix}\begin{bmatrix}r\\v\\a\end{bmatrix} \tag{4-33}$$

即

$$\dot{\boldsymbol{x}}=\boldsymbol{F}\boldsymbol{x} \tag{4-34}$$

该系统离散化（采样时间为 T）后可以写为

$$\boldsymbol{x}(k+1)=\boldsymbol{A}\boldsymbol{x}(k) \tag{4-35}$$

式中

$$\boldsymbol{A}=\exp(\boldsymbol{F}T)=\boldsymbol{I}+\boldsymbol{F}T+\frac{(\boldsymbol{F}T)^2}{2!}+\cdots \tag{4-36}$$

得到

$$\boldsymbol{A}=\begin{bmatrix}1&T&T^2/2\\0&1&T\\0&0&1\end{bmatrix} \tag{4-37}$$

假设测量位置的噪声方差为 σ^2，则系统的测量方程可写为

$$\begin{aligned}\boldsymbol{z}(k)&=\boldsymbol{C}(k)\boldsymbol{x}(k)+\boldsymbol{v}(k)\\ &=[1\quad 0\quad 0]\boldsymbol{x}(k)+\boldsymbol{v}(k)\end{aligned} \tag{4-38}$$

式中，测量噪声为

$$\boldsymbol{v}(k) \sim [0, \boldsymbol{R}(k)] \tag{4-39}$$

根据题意测量噪声方差为

$$\boldsymbol{R}(k) = \sigma^2 \tag{4-40}$$

下面来研究一下估计方差的变化，假设向前一步预测估计方差为

$$\boldsymbol{P}(k|k-1) = \begin{bmatrix} P_{11}(k|k-1) & P_{21}(k|k-1) & P_{31}(k|k-1) \\ P_{12}(k|k-1) & P_{22}(k|k-1) & P_{32}(k|k-1) \\ P_{13}(k|k-1) & P_{23}(k|k-1) & P_{33}(k|k-1) \end{bmatrix} \tag{4-41}$$

状态估计方差为

$$\boldsymbol{P}(k|k) = \begin{bmatrix} P_{11}(k|k) & P_{21}(k|k) & P_{31}(k|k) \\ P_{12}(k|k) & P_{22}(k|k) & P_{32}(k|k) \\ P_{13}(k|k) & P_{23}(k|k) & P_{33}(k|k) \end{bmatrix} \tag{4-42}$$

先将 $\boldsymbol{C}(k) = [1\ \ 0\ \ 0]$ 代入滤波增益式（4-26），得到

$$\boldsymbol{K}(k) \begin{bmatrix} P_{11}(k|k-1) \\ P_{12}(k|k-1) \\ P_{13}(k|k-1) \end{bmatrix} \frac{1}{P_{11}(k|k-1) + \sigma^2} \tag{4-43}$$

将式（4-43）代入式（4-28），得到

$$\boldsymbol{P}(k|k) = \boldsymbol{P}(k|k-1) - \frac{1}{P_{11}(k|k-1) + \sigma^2} \begin{bmatrix} P_{11}(k|k-1) & 0 & 0 \\ P_{12}(k|k-1) & 0 & 0 \\ P_{13}(k|k-1) & 0 & 0 \end{bmatrix} \boldsymbol{P}(k|k-1) \tag{4-44}$$

将式（4-44）的后两项相乘后得到

$$\boldsymbol{P}(k|k) = \boldsymbol{P}(k|k-1) - \frac{1}{P_{11}(k|k-1) + \sigma^2} \begin{bmatrix} P_{11}^2(k|k-1) & P_{11}(k|k-1)P_{12}(k|k-1) & P_{11}(k|k-1)P_{13}(k|k-1) \\ P_{12}(k|k-1)P_{11}(k|k-1) & P_{22}^2(k|k-1) & P_{12}(k|k-1)P_{13}(k|k-1) \\ P_{13}(k|k-1)P_{11}(k|k-1) & P_{13}(k|k-1)P_{12}(k|k-1) & P_{33}^2(k|k-1) \end{bmatrix} \tag{4-45}$$

用这个表达式说明估计方差从 $\boldsymbol{P}(k|k-1)$ 到 $\boldsymbol{P}(k|k)$ 的迹在减小。 $\boldsymbol{P}(k|k-1)$ 的迹为

$$\mathrm{Tr}[\boldsymbol{P}(k|k-1)] = P_{11}(k|k-1) + P_{22}(k|k-1) + P_{33}(k|k-1) \tag{4-46}$$

从方程（4-45）可以看出， $\boldsymbol{P}(k|k)$ 的迹为

$$\begin{aligned}\mathrm{Tr}[\boldsymbol{P}(k\mid k)] &= P_{11}(k\mid k)+P_{22}(k\mid k)+P_{33}(k\mid k)\\ &= \mathrm{Tr}[\boldsymbol{P}(k\mid k-1)]-\frac{P_{11}^2(k\mid k-1)+P_{22}^2(k\mid k-1)+P_{33}^2(k\mid k-1)}{P_{11}(k\mid k-1)+\sigma^2}\end{aligned}\tag{4-47}$$

当得到一个新的观测值时，希望状态的估计值变得更为准确，即希望协方差减小，上述方程表明它的确减小了，即 $\mathrm{Tr}[\boldsymbol{P}(k\mid k)]<\mathrm{Tr}[\boldsymbol{P}(k\mid k-1)]$。从 $\boldsymbol{P}(k\mid k-1)=\boldsymbol{A}(k-1)\boldsymbol{P}(k-1\mid k-1)\boldsymbol{A}^{\mathrm{T}}(k-1)+\boldsymbol{Q}(k-1)$ 还可以看出 $\mathrm{Tr}[\boldsymbol{P}(k-1\mid k)]>\mathrm{Tr}[\boldsymbol{P}(k-1\mid k-1)]$，也就是说，在向前一步预测阶段，估计方差变大了，而进行测量后，估计方差又会减小。

例 4.3　考虑如下游走模型的标量系统：

$$x(k+1)=x(k)+w(k)$$

$$z(k)=x(k)+v(k)$$

$$w(k)\sim(0,9)$$
$$v(k)\sim(0,4)$$

这是一个很简单但是在很多应用中会碰到的系统。例如，它可能代表直接测量的缓慢变量 $x(k)$，其变化由噪声项决定，测量误差由测量噪声项决定，问题是：求解滤波增益的稳态值 K_∞。

解　由系统可知，$A=C=1,\ Q=9,\ R=4$，代入式（4-31），得到 $P_\infty=12$，再将 P_∞ 代入式（4-26）得到的 $K_\infty=\dfrac{12}{12+4}=0.75$。

进一步还可以将 P_∞ 和 K_∞ 代入式（4-28）得状态估计方差稳态值 $P(\infty|\infty)$，得

$$P(\infty\mid\infty)=(I-K_\infty C)P_\infty=(1-0.75)\times12=3$$

除了上述数值计算方法之外，使用程序进行仿真，通过迭代计算式（4-27）和式（4-28），也可以得到估计方差的稳态值。首先介绍一下下面的函数，其中 A、C、Q、R 分别是系统参数，I 是可以设置的循环次数，p0 是估计方差初值。将下面的函数保存为以“steadycov”命名的 m 文件，根据不同的系统进行参数设置就会得到向前一步预测估计方差、状态估计方差以及滤波器增益向稳态值收敛的过程，MATLAB 程序中函数的输出量为 pp1、pp、KK。

```
function [pp1,pp,KK]=steadycov(A,C,Q,R,I,p0)
p=p0*ones(size(A));
pp1=[];pp=[];KK=[];
for i=1:I
p1=A*p*A'+Q;
K=p1*C'*inv(C'*p1*C+R);
p=(eye(size(A))-K*C)*p1;
pp1=[pp1 diag(p1)];
```

```
pp=[pp diag(p)];
KK=[KK K];
end
```

下一段程序是调用上面函数的主程序，根据例题 4.3 设置相应的参数，$A=1$，$C=1$，$Q=9$，$R=4$，循环次数 I 和估计方差的初值 P_0 都设置为 10，调用函数后，对计算结果画图并进行适当的标注，即可得到如图 4-3 所示的结果。

```
clc
clear
A=1;C=1;Q=9;R=4;I=10;p0=10;
[pp1,pp,KK]=steadycov(A,C,Q,R,I,p0);
subplot(3,1,1);plot(pp1);xlabel('k'),ylabel('向前一步预测估计方差')
subplot(3,1,2);plot(pp);xlabel('k'),ylabel('状态估计方差')
subplot(3,1,3);plot(KK);xlabel('k'),ylabel('滤波器增益')
```

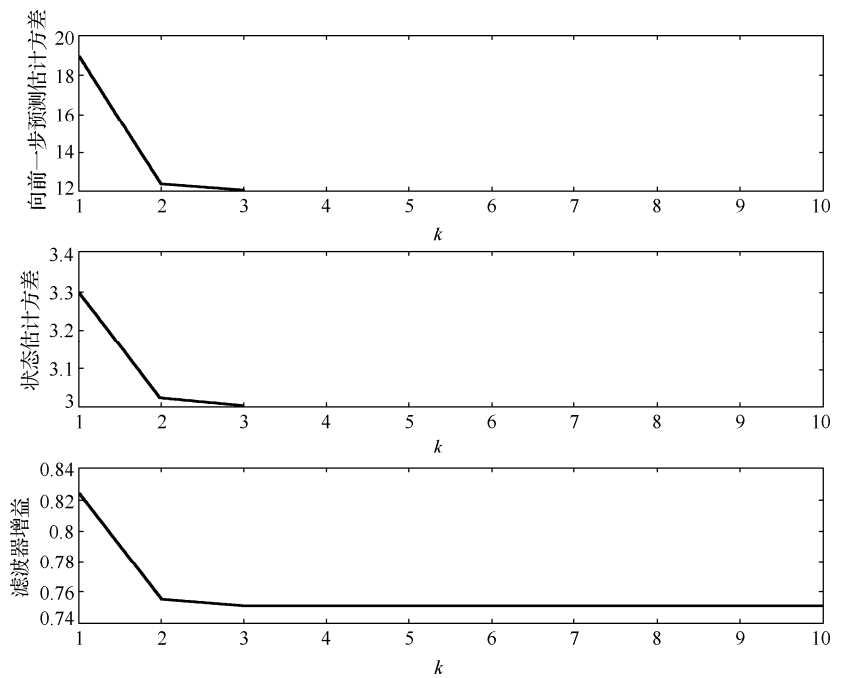


图 4-3　稳态估计结果图

可以看到结果收敛得非常快，只要经过 3、4 步的时间就可以收敛至稳态值。根据这些稳态值，就可以简化 Kalman 滤波器计算过程。利用滤波器增益的稳态值，将式（4-24）、式（4-25）状态估计的过程简化如下：

$$\hat{x}(k\,|\,k)=\hat{x}(k-1\,|\,k-1)+0.75[z(k)-\hat{x}(k-1\,|\,k-1)] \qquad (4\text{-}48)$$

在每一步的测量到来之后，可以利用式（4-48）快速得到当前步的状态估计，计算非常简单，这也是 Kalman 滤波器可以在很多情况下都能够做到实时估计的原因。

4.5　本 章 小 结

本章介绍了离散的 Kalman 滤波器，在过去的几十年里，这种估计方法几乎应用于工程的各个领域，如跟踪、导航等。Kalman 的滤波方程可以写出不同的形式，但在功能上是等价的。

Kalman 滤波器要求系统参数是已知的，如果参数未知，或者参数的值有误差，Kalman 滤波器就不能得到最优估计，甚至无法得到状态的估计，即发散。但是 Kalman 滤波器对参数的误差还是有一定的容忍度的，虽然系统参数有误差时不能得到最优估计，但很多时候都能获得和真值还算相近的估计，再加上 Kalman 可以进行实时的估计，因此，在很多需要实时处理的应用中使用非常广泛。

第 5 章　非线性 Kalman 滤波器

第 4 章讨论的线性系统滤波方法，过程方程（4-1）和测量方程（4-2）都是线性方程。在实际中，真正严格的线性系统几乎不存在，几乎所有的系统都是非线性的。我们认为是线性的许多系统，也只是在一定的条件下才成立，如果超出这个范围，就会变为非线性系统，如所熟悉的欧姆定律关系也只是在一定的电压或电流范围内才满足线性关系。

另外，还有些系统本身就具有非线性关系，如平时见到的一些测量系统，包括雷达测量系统、射频标签系统（Radio Frequency Identification，RFID），都不是线性的。针对这些非线性系统，第 4 章所讲的 Kalman 滤波器就不再适用了，本章将介绍针对非线性系统的滤波器。本章内容是在第 4 章 Kalman 滤波器基础上的衍生，以期能够处理非线性系统。

本章介绍两种非线性 Kalman 滤波器，一种是扩展 Kalman 滤波器（Extended Kalman Filter，EKF），其基本思想是利用泰勒展开，将非线性方程直接线性化。线性化后的系统模型和系统实际的非线性模型会有差别，非线性越强，差别就会越大。不过，EKF 的优势也不容忽视，由于没有附加计算，EKF 的计算量和基本的 Kalman 滤波器差不多。另一种非线性 Kalman 滤波器在原理上和 EKF 完全不同，称为不敏 Kalman 滤波器（Unscented Kalman Filter，UKF），和 EKF 相比，该滤波器由于使用了所谓的不敏变换，线性化带来的状态均值和协方差的改变要小些。本章将详细讲解这两种非线性估计方法。

5.1　扩展 Kalman 滤波器

设系统模型具有非线性关系：

$$\boldsymbol{x}(k) = f[\boldsymbol{x}(k-1), \boldsymbol{w}(k-1)] \tag{5-1}$$

$$\boldsymbol{z}(k) = h[\boldsymbol{x}(k), \boldsymbol{v}(k)] \tag{5-2}$$

式中，$\boldsymbol{x}(k)$ 为系统待估计状态，$\boldsymbol{w}(k-1)$ 为系统过程噪声，$\boldsymbol{z}(k)$ 为系统测量，$\boldsymbol{v}(k)$ 为测量噪声，$f(\cdot)$ 和 $h(\cdot)$ 是非线性的过程方程与测量方程。

式（4-1）、式（4-2）的系统矩阵 $\boldsymbol{A}(k)$、$\boldsymbol{C}(k)$ 在式（5-1）、式（5-2）中没有出现，式（5-1）、式（5-2）中的系统关系是非线性函数 $f(\cdot)$ 及 $h(\cdot)$，回忆 4.4 节的离散 Kalman

滤波器式（4-24）～式（4-28），发现系统矩阵 $\boldsymbol{A}(k)$、$\boldsymbol{C}(k)$ 还是很重要的，没有了这两个矩阵，这 5 个公式就没有了意义。

但仔细观察，式（4-24）中的新息 $\boldsymbol{Z}(k)-\boldsymbol{C}(k)\hat{\boldsymbol{x}}(k|k-1)$，其实也可以使用式（5-2）中的非线性关系实现，$\boldsymbol{C}(k)\hat{\boldsymbol{x}}(k|k-1)$ 表示的是向前一步预测状态 $\hat{\boldsymbol{x}}(k|k-1)$ 反映的测量预测值，若知道了向前一步预测状态 $\hat{\boldsymbol{x}}(k|k-1)$，根据式（5-2）可以得到相应的测量预测值 $h[\hat{\boldsymbol{x}}(k|k-1),0]$，则新息也就变为 $\boldsymbol{z}(k)-h[\hat{\boldsymbol{x}}(k|k-1),0]$，即式（4-24）可修改为

$$\hat{\boldsymbol{x}}(k|k)=\hat{\boldsymbol{x}}(k|k-1)+\boldsymbol{K}(k)[\boldsymbol{z}(k)-h(\hat{\boldsymbol{x}}(k|k-1),0] \tag{5-3}$$

同样式（4-25）说明的是如何从估计 $\hat{\boldsymbol{x}}(k-1|k-1)$ 得到向前一步预测状态 $\hat{\boldsymbol{x}}(k|k-1)$，式（5-1）反映了状态的变化，也可以从式（5-1）得到具有非线性关系的 $\hat{\boldsymbol{x}}(k|k-1)$，来代替式（4-25）：

$$\hat{\boldsymbol{x}}(k|k-1)=f[\hat{\boldsymbol{x}}(k-1|k-1),0] \tag{5-4}$$

但是滤波增益式（4-26）及估计方差式（4-27）、式（4-28）就有点麻烦，$\boldsymbol{A}(k)$ 及 $\boldsymbol{C}(k)$ 的地位无法替代。其实 $\boldsymbol{A}(k)$ 与 $f(\cdot)$，$\boldsymbol{C}(k)$ 与 $h(\cdot)$ 还是有关系的，$f(\cdot)$ 是 $\boldsymbol{x}(k-1)$ 与 $\boldsymbol{x}(k)$ 的非线性关系，而 $\boldsymbol{A}(k)$ 是线性关系，同样 $h(\cdot)$ 是 $\boldsymbol{x}(k)$ 与 $\boldsymbol{z}(k)$ 的线性关系，而 $\boldsymbol{C}(k)$ 是线性关系。

而目前的情况是系统实际上是非线性的，即 $f(\cdot)$ 与 $h(\cdot)$ 是非线性函数。因此无法使用 4.4 节的 Kalman 滤波器直接估计状态，也就是说，无法直接使用线性 Kalman 滤波器获得系统的状态估计。

为了能估计非线性系统的状态，人们首先想到的是近似估计结果，这是设计算法的一个基本思想，如果不能得到最优解，但也不想交“白卷”，给出一个差不多的结果也是个答案。但必须要说明这个“差不多”的结果到底“差”多少，这是研究人员特有的严谨。一般情况下，要定量地说明到底“差多少”有两种方法：从理论上给出证明，或者设计大量合理的仿真实验来说明。

接下来说明该如何估计非线性系统（5-1）、系统（5-2）的状态。将非线性函数 $f[\boldsymbol{x}(k-1),\boldsymbol{w}(k-1)]$ 在 $\boldsymbol{x}(k-1)=\hat{\boldsymbol{x}}(k-1|k-1)$ 和 $\boldsymbol{w}(k-1)=0$ 处进行一阶泰勒级数展开：

$$\begin{aligned}\boldsymbol{x}(k)=&f[\hat{\boldsymbol{x}}(k-1|k-1),0]+\left.\frac{\partial f}{\partial \boldsymbol{x}}\right|_{\hat{\boldsymbol{x}}(k-1|k-1)}[\boldsymbol{x}(k-1)-\hat{\boldsymbol{x}}(k-1|k-1)]\\&+\left.\frac{\partial f}{\partial \boldsymbol{w}}\right|_{\hat{\boldsymbol{x}}(k-1|k-1)}\cdot\boldsymbol{w}(k-1)\\=&\left.\frac{\partial f}{\partial \boldsymbol{x}}\right|_{\hat{\boldsymbol{x}}(k-1|k-1)}\boldsymbol{x}(k-1)+[f(\hat{\boldsymbol{x}}(k-1|k-1),0)-\left.\frac{\partial f}{\partial \boldsymbol{x}}\right|_{\hat{\boldsymbol{x}}(k-1|k-1)}\cdot\hat{\boldsymbol{x}}(k-1|k-1)]\\&+\left.\frac{\partial f}{\partial \boldsymbol{w}}\right|_{\hat{\boldsymbol{x}}(k-1|k-1)}\cdot\boldsymbol{w}(k-1)\end{aligned}$$

设

$$\begin{cases} \left.\dfrac{\partial f}{\partial \boldsymbol{x}}\right|_{\hat{\boldsymbol{x}}(k-1|k-1)} = \boldsymbol{F}(k) \\ \left.\dfrac{\partial f}{\partial \boldsymbol{w}}\right|_{\hat{\boldsymbol{x}}(k-1|k-1)} = \boldsymbol{L}(k) \end{cases} \tag{5-5}$$

则式（5-5）可变为

$$\boldsymbol{x}(k) = \boldsymbol{F}(k)\boldsymbol{x}(k-1) + [f(\hat{\boldsymbol{x}}(k-1|k-1),0) - \boldsymbol{F}(x)\hat{\boldsymbol{x}}(k-1|k-1)] + \boldsymbol{L}(k)\boldsymbol{w}(k-1) \tag{5-6}$$

现在来看式（5-1）与式（5-6）的关系，很明显，式（5-1）是与实际非线性状态转移关系一致的描述，而式（5-6）则是式（5-1）一阶泰勒展开的近似，两者是不相同的。从数学知识可知，泰勒级数展开的阶次越多，两者就越接近，展开为无穷阶，两者才会相同。但在一般情况下，可以取合适的阶次，并承认两者之间存在差距。不过式（5-6）给出了一个状态转移的线性关系 $\boldsymbol{F}(x)$，可以根据这个关系利用 Kalman 滤波器实现状态估计。

需要考虑的是过程噪声项也附加了系数，所以经过泰勒级数展开后，过程噪声的方差也有了变化，不再是原来的 $\boldsymbol{Q}(k)$，而是 $\boldsymbol{L}(k)\boldsymbol{Q}(k)\boldsymbol{L}^{\mathrm{T}}(k)$。同理，也可以利用泰勒级数展开将测量方程线性化，将式（5-2）在 $\boldsymbol{x}(k) = \hat{\boldsymbol{x}}(k|k-1)$ 及 $\boldsymbol{v}(k) = 0$ 处展开：

$$\begin{aligned} \boldsymbol{y}(k) &= h[\hat{\boldsymbol{x}}(k|k-1),0] + \left.\frac{\partial h}{\partial \boldsymbol{x}}\right|_{\boldsymbol{x}=\hat{\boldsymbol{x}}(k|k-1)} \cdot [\boldsymbol{x}(k) - \hat{\boldsymbol{x}}(k|k-1)] + \left.\frac{\partial h}{\partial \boldsymbol{v}}\right|_{\boldsymbol{x}=\hat{\boldsymbol{x}}(k|k-1)} \cdot \boldsymbol{v}(k) \\ &= \left.\frac{\partial h}{\partial \boldsymbol{x}}\right|_{\boldsymbol{x}=\hat{\boldsymbol{x}}(k|k-1)} \cdot \boldsymbol{x}(k) + \left[h(\hat{\boldsymbol{x}}(k|k-1),0) - \left.\frac{\partial h}{\partial \boldsymbol{x}}\right|_{\boldsymbol{x}=\hat{\boldsymbol{x}}(k|k-1)} \cdot \hat{\boldsymbol{x}}(k|k-1) \right] + \left.\frac{\partial h}{\partial \boldsymbol{v}}\right|_{\boldsymbol{x}=\hat{\boldsymbol{x}}(k|k-1)} \cdot \boldsymbol{v}(k) \end{aligned}$$

设

$$\begin{cases} \left.\dfrac{\partial h}{\partial \boldsymbol{x}}\right|_{\boldsymbol{x}=\hat{\boldsymbol{x}}(k|k-1)} = \boldsymbol{H}(k) \\ \left.\dfrac{\partial h}{\partial \boldsymbol{v}}\right|_{\boldsymbol{x}=\hat{\boldsymbol{x}}(k|k-1)} = \boldsymbol{M}(k) \end{cases} \tag{5-7}$$

则

$$\boldsymbol{y}(k) = \boldsymbol{H}(k)\boldsymbol{x}(k) + \left[h(\hat{\boldsymbol{x}}(k|k-1),0) - \left.\frac{\partial h}{\partial \boldsymbol{x}}\right|_{\boldsymbol{x}=\hat{\boldsymbol{x}}(k|k-1)} \cdot \hat{\boldsymbol{x}}(k|k-1) \right] + \boldsymbol{M}(k)\boldsymbol{v}(k) \tag{5-8}$$

可见，测量噪声的方差变为 $\boldsymbol{M}(k)\boldsymbol{R}(k)\boldsymbol{M}^{\mathrm{T}}(k)$。所以，利用式（5-5）、式（5-6）的 $\boldsymbol{F}(k)$、$\boldsymbol{H}(k)$ 及相应的噪声方差，改写式（4-26）～式（4-28）的滤波增益及估计方差如下：

$$\boldsymbol{K}(k) = \boldsymbol{P}(k|k-1)\boldsymbol{H}^{\mathrm{T}}(k)[\boldsymbol{H}^{\mathrm{T}}(k)\boldsymbol{P}(k|k-1)\boldsymbol{H}(k) + \boldsymbol{M}(k)\boldsymbol{R}(k)\boldsymbol{M}^{\mathrm{T}}(k)]^{-1} \tag{5-9}$$

$$\boldsymbol{P}(k\mid k-1)=\boldsymbol{F}(k-1)\boldsymbol{P}(k-1\mid k-1)\boldsymbol{F}^{\mathrm{T}}(k-1)+\boldsymbol{L}(k-1)\boldsymbol{Q}(k-1)\boldsymbol{L}^{\mathrm{T}}(k-1) \quad (5\text{-}10)$$

$$\boldsymbol{P}(k\mid k)=[\boldsymbol{I}-\boldsymbol{K}(k)\boldsymbol{H}(k)]\boldsymbol{P}(k\mid k-1) \quad (5\text{-}11)$$

如表 5-1 所示，给出了标准 Kalman 滤波器和 EKF 的差别。

表 5-1　标准 Kalman 滤波器和 EKF 的差别

<table>
<tr><th colspan="2"></th><th>标准 Kalman 滤波器</th><th colspan="2">EKF</th></tr>
<tr><td colspan="2">系统模型描述</td><td>$\boldsymbol{x}(k+1)=\boldsymbol{A}(k)\boldsymbol{x}(k)+\boldsymbol{w}(k)$
$\boldsymbol{z}(k)=\boldsymbol{C}(k)\boldsymbol{x}(k)+\boldsymbol{v}(k)$
$E[\boldsymbol{w}(k)\boldsymbol{w}^{\mathrm{T}}(j)]=\boldsymbol{Q}(k)\delta(k-j)$
$E[\boldsymbol{v}(k)\boldsymbol{v}^{\mathrm{T}}(j)]=\boldsymbol{R}(k)\delta(k-j)$
$E[\boldsymbol{w}(k)\boldsymbol{v}^{\mathrm{T}}(j)]=0$</td><td colspan="2">$\boldsymbol{x}(k)=f[\boldsymbol{x}(k-1),\boldsymbol{w}(k-1)]$
$\boldsymbol{z}(k)=h[\boldsymbol{x}(k),\boldsymbol{v}(k)]$
$E[\boldsymbol{w}(k)\boldsymbol{w}^{\mathrm{T}}(j)]=\boldsymbol{Q}(k)\delta(k-j)$
$E[\boldsymbol{v}(k)\boldsymbol{v}^{\mathrm{T}}(j)]=\boldsymbol{R}(k)\delta(k-j)$
$E[\boldsymbol{w}(k)\boldsymbol{v}^{\mathrm{T}}(j)]=0$</td></tr>
<tr><td colspan="2"></td><td></td><td>线性化</td><td>$\left.\frac{\partial f}{\partial \boldsymbol{x}}\right\vert_{\hat{\boldsymbol{x}}(k-1\mid k-1)}=\boldsymbol{F}(k)$
$\left.\frac{\partial f}{\partial \boldsymbol{w}}\right\vert_{\hat{\boldsymbol{x}}(k-1\mid k-1)}=\boldsymbol{L}(k)$
$\left.\frac{\partial h}{\partial \boldsymbol{x}}\right\vert_{\boldsymbol{x}=\tilde{\boldsymbol{x}}(k\mid k-1)}=\boldsymbol{H}(k)$
$\left.\frac{\partial h}{\partial \boldsymbol{v}}\right\vert_{\boldsymbol{x}=\tilde{\boldsymbol{x}}(k\mid k-1)}=\boldsymbol{M}(k)$</td></tr>
<tr><td rowspan="5">Kalman 滤波器</td><td>预测</td><td>$\hat{\boldsymbol{x}}(k\mid k-1)=\boldsymbol{A}(k-1)\hat{\boldsymbol{x}}(k-1\mid k-1)$</td><td colspan="2">$\hat{\boldsymbol{x}}(k\mid k-1)=f[\hat{\boldsymbol{x}}(k-1\mid k-1),0]$</td></tr>
<tr><td>更新</td><td>$\hat{\boldsymbol{x}}(k\mid k)=\hat{\boldsymbol{x}}(k\mid k-1)+\boldsymbol{K}(k)$
$[\boldsymbol{z}(k)-\boldsymbol{C}(k)\hat{\boldsymbol{x}}(k\mid k-1)]$</td><td colspan="2">$\hat{\boldsymbol{x}}(k\mid k)=\hat{\boldsymbol{x}}(k\mid k-1)$
$+\boldsymbol{K}(k)[\boldsymbol{z}(k)-h(\hat{\boldsymbol{x}}(k\mid k-1),0]$</td></tr>
<tr><td>滤波器增益</td><td>$\boldsymbol{K}(k)=\boldsymbol{P}(k\mid k-1)\boldsymbol{C}^{\mathrm{T}}(k)$
$\times[\boldsymbol{C}^{\mathrm{T}}(k)\boldsymbol{P}(k\mid k-1)\boldsymbol{C}(k)+\boldsymbol{R}(k)]^{-1}$</td><td colspan="2">$\boldsymbol{K}(k)=\boldsymbol{P}(k\mid k-1)\boldsymbol{H}^{\mathrm{T}}(k)\times[\boldsymbol{H}^{\mathrm{T}}(k)\boldsymbol{P}(k\mid k-1)$
$\boldsymbol{H}(k)+\boldsymbol{M}(k)\boldsymbol{R}(k)\boldsymbol{M}^{\mathrm{T}}(k)]^{-1}$</td></tr>
<tr><td>向前一步预测方差</td><td>$\boldsymbol{P}(k\mid k-1)=\boldsymbol{A}(k-1)\boldsymbol{P}(k-1\mid k-1)$
$\boldsymbol{A}^{\mathrm{T}}(k-1)+\boldsymbol{Q}(k-1)$</td><td colspan="2">$\boldsymbol{P}(k\mid k-1)=\boldsymbol{F}(k-1)\boldsymbol{P}(k-1\mid k-1)\boldsymbol{F}^{\mathrm{T}}(k-1)$
$+\boldsymbol{L}(k-1)\boldsymbol{Q}(k-1)\boldsymbol{L}^{\mathrm{T}}(k-1)$</td></tr>
<tr><td>状态估计方差</td><td>$\boldsymbol{P}(k\mid k)=[\boldsymbol{I}-\boldsymbol{K}(k)\boldsymbol{C}(k)]\times\boldsymbol{P}(k\mid k-1)$</td><td colspan="2">$\boldsymbol{P}(k\mid k)=[\boldsymbol{I}-\boldsymbol{K}(k)\boldsymbol{H}(k)]\times\boldsymbol{P}(k\mid k-1)$</td></tr>
</table>

下面举例说明如何使用 EKF。

例 5.1　假设在二维空间内只能测量运动目标到原点的距离，试利用 EKF 方法给出非线性的计算方法。

解析　先来看一下系统方程，设二维空间分别用横、纵坐标用 x、y 来表示，假设 x、y 轴运动的过程噪声不相关，则过程方程为

$$\begin{bmatrix}x(k)\\ y(k)\end{bmatrix}=\begin{bmatrix}1 & 0\\ 0 & 1\end{bmatrix}\begin{bmatrix}x(k-1)\\ y(k-1)\end{bmatrix}+\begin{bmatrix}w_x(k-1)\\ w_y(k-1)\end{bmatrix} \quad (5\text{-}12)$$

过程噪声为

$$\boldsymbol{Q}(k-1)=\begin{bmatrix}E(w_x(k-1)w_x^{\mathrm{T}}(k-1) & 0\\ 0 & E(w_y(k-1)w_y^{\mathrm{T}}(k-1)\end{bmatrix} \quad (5\text{-}13)$$

测量方程为

$$z(k)=\sqrt{x^2(k)+y^2(k)}+v(k) \tag{5-14}$$

该题中的过程方程是线性的，测量方程是非线性的，但测量噪声与所测距离之间也是线性的，这是式（5-1）、式（5-2）所描述非线性方程的特殊情况。那么，应该如何使用 EKF（式（5-3）～式（5-5）、式（5-7）、式（5-9）～式（5-11））的方法呢？

因为过程方程是线性的，所以不需要进行线性化，即不需要计算式（5-3）和式（5-10）中的线性化矩阵 $\boldsymbol{K}(k-1)$ 和 $\boldsymbol{L}(k-1)$，而是使用式（4-25）、式（4-27）即可。

待估计状态是一个向量，包括横轴位置 $x(k)$ 及纵轴位置 $y(k)$，为了和前面的变量区别开来，本例题中的待估计状态使用 $\vec{\boldsymbol{x}}(k)$ 表示，即 $\vec{\boldsymbol{x}}(k)=\begin{bmatrix}x(k)\\y(k)\end{bmatrix}$，则

$$\begin{aligned}\boldsymbol{H}(k)&=\left.\frac{\partial h}{\partial \vec{\boldsymbol{x}}}\right|_{\vec{\boldsymbol{x}}=[\hat{\boldsymbol{x}}(k|k-1),\hat{\boldsymbol{y}}(k|k-1)]^{\mathrm{T}}}\\&=\frac{[\hat{\boldsymbol{x}}(k\,|\,k-1),\hat{\boldsymbol{y}}(k\,|\,k-1)]^{\mathrm{T}}}{\sqrt{\hat{\boldsymbol{x}}^2(k\,|\,k-1)+\hat{\boldsymbol{y}}^2(k\,|\,k-1)}}\end{aligned} \tag{5-15}$$

当前一步预测估计 $\hat{\boldsymbol{x}}(k\,|\,k-1)$、$\hat{\boldsymbol{y}}(k\,|\,k-1)$ 已知时，$\boldsymbol{H}(k)$ 是一个 1×2 矩阵，利用 EKF 的估计方程为

$$\hat{\vec{\boldsymbol{x}}}(k\,|\,k)=\hat{\vec{\boldsymbol{x}}}(k\,|\,k-1)+\boldsymbol{K}(k)[\boldsymbol{z}(k)-\sqrt{\hat{\boldsymbol{x}}^2(k\,|\,k-1)+\hat{\boldsymbol{y}}^2(k\,|\,k-1)}] \tag{5-16}$$

因为系统的过程方程是线性的，所以向前一步预测估计和第 4 章线性 Kalman 滤波器一样，即 $\boldsymbol{A}(k)=\begin{bmatrix}1&0\\0&1\end{bmatrix}$，所以

$$\hat{\vec{\boldsymbol{x}}}(k\,|\,k-1)=\boldsymbol{A}(k)\hat{\vec{\boldsymbol{x}}}(k-1\,|\,k-1) \tag{5-17}$$

其估计方差为

$$\boldsymbol{P}(k\,|\,k-1)=\boldsymbol{A}(k)\boldsymbol{P}(k-1\,|\,k-1)\boldsymbol{A}^{\mathrm{T}}(k)+\boldsymbol{Q}(k-1) \tag{5-18}$$

又式（5-14）中状态与测量噪声之间为线性加和关系，则 $\dfrac{\partial h}{\partial \boldsymbol{v}}=\boldsymbol{I}$，即 $\boldsymbol{M}(k)=\boldsymbol{I}$，所以滤波增益为

$$\boldsymbol{K}(k)=\boldsymbol{P}(k\,|\,k-1)\boldsymbol{H}^{\mathrm{T}}(k)[\boldsymbol{H}^{\mathrm{T}}(k)\boldsymbol{P}(k\,|\,k-1)\boldsymbol{H}(k)+\boldsymbol{R}(k)]^{-1} \tag{5-19}$$

估计方差计算方法为

$$\boldsymbol{P}(k\,|\,k)=[\boldsymbol{I}-\boldsymbol{K}(k)]\boldsymbol{H}(k)\boldsymbol{P}(k\,|\,k-1) \tag{5-20}$$

式中，$\boldsymbol{H}(k)$ 为式（5-15）测量方程求偏导后的矩阵。

下面总结一下 EKF。从前面的分析可以看到，EKF 的基本原理是利用一阶泰勒级

数展开，使用展开后的一阶分量来代替原来的非线性方程。从泰勒级数的原理可知，当泰勒级数本身是无穷次级数时，等式才成立。因此，如果只有一阶展开，由 EKF 获得的估计状态其实和实际的非线性方程的结果是有差别的。

前面已经提到过，如果 Kalman 滤波器参数完全已知，Kalman 滤波器就可以获得最优估计。但是经过一阶泰勒级数展开之后，EKF 使用的参数和系统的实际参数不同，从理论上来讲，这种在非线性 Kalman 滤波器中利用一阶泰勒级数展开的 EKF 已经不能得到最优估计了。

为了减小估计方差，研究者也给出了二阶 EKF，此时 Kalman 滤波器的方程也变复杂了。当然，从理论上讲，三阶、四阶的 EKF 可能会使系统的参数与原来的非线性关系更为接近。但可想而知，系统在进行估计的时候，滤波器的方程会更加复杂，而且从理论上来讲，这种使用线性关系来代替非线性关系的方法，只要其阶数有限，就一定会带有误差。所以，不管是几阶的 EKF，都会有系统误差。甚至有时候如果非线性非常强，经过线性化的结果并不能很好体现原来的非线性关系，可能会导致估计系统发散。应该了解到，EKF 的这些缺陷是由工作原理造成的，所以，即便扩大阶数，进行二阶或三阶泰勒展开，也不能从根本上消除由于线性化造成的估计结果存在偏差。

因此，人们研究了其他的非线性 Kalman 滤波的方法，例如，接下来将讲到的不敏 Kalman 滤波器，其基本原理和上面讲到的 EKF 有着本质上的差别，在很多时候 UKF 都会比 EKF 获得更好的结果。

5.2　不敏 Kalman 滤波器

先前已经讨论过，当系统的非线性比较严重时，EKF 的估计并不准确。这是因为 EKF 是依赖线性化传播状态均值和方差的，即 EKF 要通过线性化非线性系统函数得到线性关系，进而用这种线性关系来重新表达原来系统的状态和噪声均值的非线性关系。本章要讨论的 UKF 是 Kalman 滤波器的延伸，UKF 降低了 EKF 线性化带来的误差。

本节首先讨论在已知状态均值和协方差时，通过非线性关系，非线性函数状态和协方差是如何改变的，接下来再讨论所谓的“不敏变换”。通过不敏变换，随机变量的均值及协方差的计算结果比直接用非线性函数计算更接近一些。同时，还会利用这种不敏变换来改进 Kalman 滤波器，这会比 EKF 具有更小的线性化误差，最后，还会介绍一些改进的 UKF。

5.2.1　非线性变换的均值和方差

本节将研究随机变量经过非线性函数时，状态变量的均值和协方差是如何变化的，进而说明 EKF 的线性近似是如何产生均值和协方差误差的。本节知识本质上和状态估计、Kalman 滤波器、EKF、UKF 都没有直接联系，但为后面推导 UKF 作了

铺垫，也为理解依赖线性化处理的 EKF 为何会导致较大估计误差提供依据。考虑非线性函数：

$$\begin{cases} y_1 = r\cos\theta \\ y_2 = r\sin\theta \end{cases} \tag{5-21}$$

假设有传感器可以测量距离 r 和角度 θ，想将测量数据变换到直角坐标系 y_1 和 y_2，这种坐标变换通常为

$$\boldsymbol{y} = h(\boldsymbol{x}) \tag{5-22}$$

式中，$\boldsymbol{y}$ 是 $h(\boldsymbol{x})$ 的状态输出函数，状态的定义为 $\boldsymbol{x} = \begin{bmatrix} r \\ \theta \end{bmatrix}$。

假设 x_1（距离 r）的测量数据均值为 1，并带有方差为 δ_r 均匀分布的测量噪声，x_2（角度 θ）的均值为 $\dfrac{\pi}{2}$，且噪声方差为 δ_θ，并且 r 和 θ 是相互独立的。

现在考虑 y_1、y_2 的均值，由式（5-21）直观的感觉是 y_1 的均值是 0，y_2 的均值是 1，那到底是不是这样呢？对式（5-22）进行一阶线性化，两边取期望值，得到

$$\begin{aligned} \overline{\boldsymbol{y}} = E[h(\boldsymbol{x})] &\approx E\left[h(\overline{\boldsymbol{x}}) + \left.\frac{\partial h}{\partial \boldsymbol{x}}\right|_{\overline{x}} (\boldsymbol{x} - \overline{\boldsymbol{x}}) \right] \\ &= h(\overline{\boldsymbol{x}}) = \begin{bmatrix} 0 \\ 1 \end{bmatrix} \end{aligned} \tag{5-23}$$

式（5-23）中使用了关系 $\left.\dfrac{\partial h}{\partial \boldsymbol{x}}\right|_{\overline{x}} (\boldsymbol{x} - \overline{\boldsymbol{x}}) = 0$。式（5-23）也说明了 y_1 的均值为 0，y_2 的均值为 1。直觉和一阶线性化得出了形同的结论。但这里不得不提前透漏一下，这不是事实，下面进行分析。

把 r 和 θ 写成如下均值加噪声的形式：

$$\begin{cases} r = \overline{r} + \tilde{r} \\ \theta = \overline{\theta} + \tilde{\theta} \end{cases} \tag{5-24}$$

式中，$\overline{r}$ 和 $\overline{\theta}$ 分别为 r 和 θ 的均值，$\tilde{r}$ 和 $\tilde{\theta}$ 分别为测量噪声。现在分别来看看 y_1 和 y_2 的值，以及与 $\overline{r}$、$\overline{\theta}$、$\tilde{r}$ 及 $\tilde{\theta}$ 的关系：

$$\begin{aligned} y_1 &= r\cos\theta \\ &= (\overline{r} + \tilde{r})\cos(\overline{\theta} + \tilde{\theta}) \\ &= (\overline{r} + \tilde{r})(\cos\overline{\theta}\cos\tilde{\theta} - \sin\overline{\theta}\sin\tilde{\theta}) \end{aligned} \tag{5-25}$$

对式（5-25）求期望值，得

$$E(y_1) = E[(\overline{r} + \tilde{r})(\cos\overline{\theta}\cos\tilde{\theta} - \sin\overline{\theta}\sin\tilde{\theta})] \tag{5-26}$$

因为 $\bar{r}$ 与 $\tilde{\theta}$ 是相互独立的，相乘后再求期望值，得

$$E(y_1)=\bar{y}_1=\bar{r}\cos\bar{\theta}=0 \tag{5-27}$$

这里和式（5-23）矩阵中的第一个 0 是一致的。接下来看看 y_2，重复上述过程，得

$$\begin{aligned} y_2&=E(r\sin\theta)=E[(\bar{r}+\tilde{r})\sin(\bar{\theta}+\tilde{\theta})]\\ &=E[(\bar{r}+\tilde{r})(\sin\bar{\theta}\cos\tilde{\theta}+\cos\bar{\theta}\sin\tilde{\theta})] \end{aligned} \tag{5-28}$$

即

$$\bar{y}_2=\bar{r}\sin\bar{\theta}E(\cos\tilde{\theta})=E(\cos\tilde{\theta}) \tag{5-29}$$

除非假设 $\tilde{\theta}$ 的分布，否则就无法求出 $\bar{y}_2$ 的具体值。因此，这里假设 $\tilde{\theta}$ 是在 $\pm\theta_m$ 之间的均匀分布，这样可以计算出

$$y_2=E(\cos\tilde{\theta})=\frac{\sin\theta_m}{\theta_m} \tag{5-30}$$

期望 $\bar{y}_2$ 等于 1 来证实式（5-23）的合理性，但发现对于所有的 $\theta_m>0$，有 $y_2<0$，则

$$\frac{\sin\theta_m}{\theta_m}<1 \tag{5-31}$$

并且，只有 $\theta_m\to 0$ 时满足

$$\lim_{\theta_m\to 0}\frac{\sin\theta_m}{\theta_m}=1 \tag{5-32}$$

这个分析解释了一阶线性化存在问题，就是与其实际的非线性变换并不一致。

下面用仿真程序来说明以上的分析过程。利用 MATLAB 产生 300 个随机生成的 r 值和 θ 值，计算 y_1 和 y_2 后得到 300 个点，分别作为横、纵坐标画在图上，如图 5-1 所示。其中 $\tilde{r}$ 在 ±0.01 间的均匀分布，$\tilde{\theta}$ 在 0.35rad 之间分布，$\tilde{r}$ 的小方差和 $\tilde{\theta}$ 的大方差导致点的分布形式呈弧形，这种弧形的分布结果导致 $\bar{y}_2<1$。MATLAB 程序中利用均匀分布函数 rand 产生 rnoise 表示 r 的噪声 $\tilde{r}$，与 r 均值 rmean 相加得到 r。θ 的取值方式与 r 类似，程序中 θ 用变量 sita 表示。然后利用非线性关系式（5-21）求 y_1 和 y_2，在程序中用 y1 和 y2 表示。再利用均值函数 mean，分别求出 y_1 和 y_2 的值，得到 meany1 和 meany2，即随机变量 y_1 和 y_2 的真实均值。程序中使用 Monte Carlo 仿真的方法，对 300 个点的横、纵坐标分别求均值得（–0.0009，0.9801），这与式（5-27）和式（5-30）计算值（0，0.9797）有一定差别，但是也非常相似，如果增加点数，二者会更接近。程序剩余部分用于绘制结果图。

```
clc
clear
a=-0.01;b=0.01;
```

```
rnoise= a + (b-a).*rand(300,1);
rmean=1;
r=rnoise+rmean;

c=-0.35;d=0.35;
sitanoise= c + (d-c).*rand(300,1);
sitamean=pi/2;
sita=sitanoise+sitamean;

y1=r.*cos(sita);
y2=r.*sin(sita);

meany1=mean(y1);
meany2=mean(y2);
%画出结果图。
plot(y1,y2,'.')
xlabel('y1');ylabel('y2')
hold on


plot(0,1,'ko')
plot(meany1,meany2,'k*')
```

最后结果如图 5-1 所示。

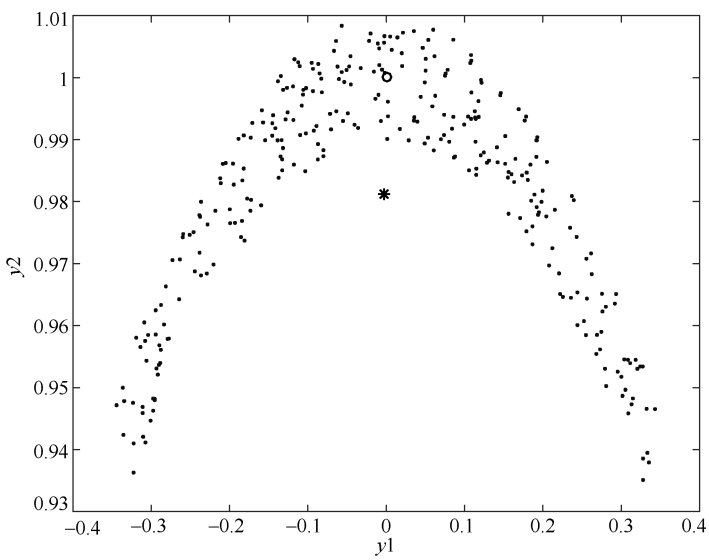


图 5-1　非线性变换均值

图 5-1 中给出了 300 个点的一阶线性近似及其非线性变换的真实均值，其中“○”表示线性变换之后的均值（0，1），“*”表示非线性变换的真实均值（–0.0023，0.9811）。当 EKF 使用一阶线性化来更新状态均值时，得到的均值也是（0，1），而非真实均值（–0.0023，0.9811）。

下面再来看看非线性变换的协方差。y 的协方差表示为

$$\boldsymbol{P}_y = E[(\boldsymbol{y}-\bar{\boldsymbol{y}})(\boldsymbol{y}-\bar{\boldsymbol{y}})^{\mathrm{T}}] \tag{5-33}$$

先来看一下基于线性变换的协方差变化。利用 EKF 的方法，对于式（5-22）的测量方程，因为没有噪声项，所以有

$$\boldsymbol{P}_y = \boldsymbol{H}\boldsymbol{P}_x\boldsymbol{H}^{\mathrm{T}} \tag{5-34}$$

式中，$\boldsymbol{H}$ 为一阶泰勒级数展开式，即 $\boldsymbol{H}=\left.\dfrac{\partial h}{\partial \boldsymbol{x}}\right|_{\boldsymbol{x}=\bar{\boldsymbol{x}}}$，具体表示为

$$\boldsymbol{H}=\left.\frac{\partial h}{\partial \boldsymbol{x}}\right|_{\boldsymbol{x}=\bar{\boldsymbol{x}}}=\left.\begin{bmatrix}\cos\theta & -r\sin\theta\\ \sin\theta & r\cos\theta\end{bmatrix}\right|_{\boldsymbol{x}=\bar{\boldsymbol{x}}}=\begin{bmatrix}0 & -1\\ 1 & 0\end{bmatrix} \tag{5-35}$$

式中，使用 $\boldsymbol{x}$ 的均值 $\bar{\boldsymbol{x}}=\begin{bmatrix}1\\ \dfrac{\pi}{2}\end{bmatrix}$，又 $\boldsymbol{P}_x$ 为状态 $\begin{bmatrix}r\\ \theta\end{bmatrix}$ 的方差，即

$$\boldsymbol{P}_x = E\left(\begin{bmatrix}r-\bar{r}\\ \theta-\bar{\theta}\end{bmatrix}[r-\bar{r}\quad \theta-\bar{\theta}]^{\mathrm{T}}\right)=\begin{bmatrix}\delta_r^2 & 0\\ 0 & \delta_\theta^2\end{bmatrix} \tag{5-36}$$

则由式（5-34）得一阶线性化后的协方差为

$$\boldsymbol{P}_y = \boldsymbol{H}\boldsymbol{P}_y\boldsymbol{H}^{\mathrm{T}}=\begin{bmatrix}0 & -1\\ 1 & 0\end{bmatrix}\begin{bmatrix}\delta_r^2 & 0\\ 0 & \delta_\theta^2\end{bmatrix}\begin{bmatrix}0 & -1\\ 1 & 0\end{bmatrix}=\begin{bmatrix}\delta_\theta^2 & 0\\ 0 & \delta_r^2\end{bmatrix} \tag{5-37}$$

再来看看非线性方程（5-22）的实际协方差，利用数值计算的方法比较复杂，这里采用仿真方法求图 5-1 中 300 个数据点横、纵轴的方差。利用 Monte Carlo 仿真方法，N 个数据点的方差为：$\mathrm{cov}(x)=\dfrac{\sum_{i=1}^{N}(x_i-\bar{x})^2}{N}$，在本例中，横轴、纵轴的方差分别为

$$\begin{cases}\mathrm{cov}(x)=\dfrac{\sum_{i=1}^{300}(x_i-\bar{x})^2}{300}\\ \mathrm{cov}(y)=\dfrac{\sum_{i=1}^{300}(y_i-\bar{y})^2}{300}\end{cases} \tag{5-38}$$

式中，x_i、y_i 分别为横轴、纵轴的数据，$\bar{x}$、$\bar{y}$ 分别为横轴、纵轴的均值，并且 $\bar{x}=0, \bar{y}=\dfrac{\sin(0.35)}{0.35}$。

下面要将方差画在图上。为了更好地理解程序，先介绍一下将要使用的 rectangle 函数。该函数使用参数'Curvature',[1,1]画出圆形或椭圆形，画出图形的位置使用参数'Position'来设置。

如参数设置为'Position'，[0-0.35,1-0.01,0.35*2,0.01*2]，要画的圆心在（0,1），横轴、纵轴分别为 0.7、0.02 的椭圆。还要说明的是，对所求方差（5-38）开方，得到误差的上下界，作为椭圆横、纵轴的半径，画的圆才是非线性变换的协方差范围。对应的程序语句如下：

```
rectangle('Position',[0-0.35,1-0.01,0.35*2,0.01*2],'Curvature',[1,1])

e=sqrt(sum((y1-0).^2)/300)
f=sqrt(sum((y2-sin(0.35)/0.35).^2)/300)
rectangle('Position',[0-e,sin(0.35)/0.35-f,e*2,f*2],'Curvature',[1,1])
```

其中第一个 rectangle 语句产生的是线性化的方差结果，即均值为（0, 1），r、θ 的方差由式（5-37）计算得到，开方后分别为 0.35、0.01。要画成椭圆，则横、纵轴的直径分别为 0.7、0.02。程序中 e 和 f 是使用式（5-38）求的 300 个点的实际横纵轴的方差，再进行开方，利用 rectangle 函数画出的椭圆表示 y_1 和 y_2 的方差范围，结果如图 5-2 所示。

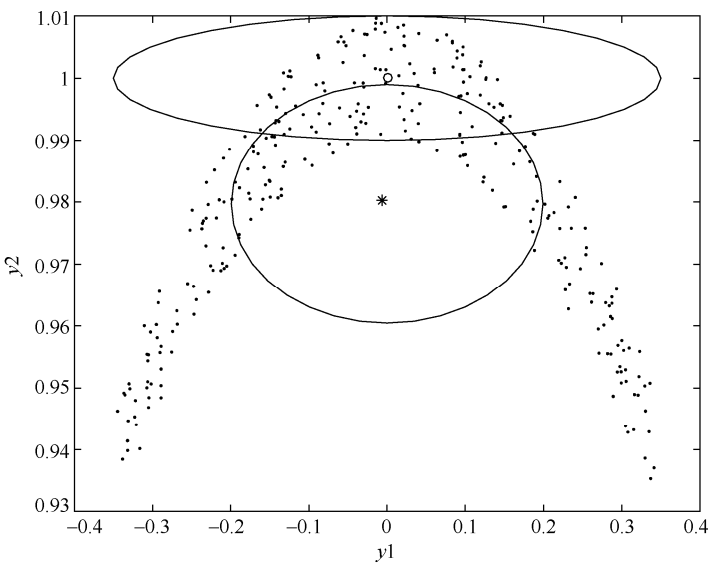


图 5-2　线性近似均值、协方差和非线性的真实均值、协方差比较

图 5-2 中 300 个随即生成点线性近似协方差和非线性的真实协方差，其中 r 的偏差在 ±0.01 之间均匀分布，θ 在 ±0.35 rad 间均匀分布。由于每次产生的随机数据并不完全相同，根据式（5-38）求得的方差每次会略有偏差，但相差不多。可以确定的是经过非线性函数的方差和一阶线性近似有很大差别，一阶线性近似的结果很不准确。

5.2.2　不敏变换

前面已经知道可以使用很多点来求均值和方差，这样得到的估计值和真实值比较接近。如上面的例子，使用 300 个点通过程序计算之后，得到的结果和真实值比较接近。因此，可以不经过一次泰勒级数进行线性展开，而是使用很多点，让每个点经过非线性关系之后，得到这些点的均值和方差，进而得到相对比较准确的估计值。这种已知噪声分布情况，通过产生很多点并利用每一个点进行估计的方法叫 Monte Carla 滤波器。

可是 300 个点比较多，由此就有人想到，可不可以在这些点当中选择几个具有足够代表性的合适的点，再利用这些点求取均值和方差，使估计值尽量接近真实均值和方差。如果这种方法可行，就不需要用 300 个点，而是使用几个点就可以达到目的。

对于这个问题需要考虑两个方面的内容：一是怎么选择这几个点，选取多少合适；二是这几个点对真实的均值和方差的重现度是多少，利用这几个点的非线性关系计算均值和方差与通过非线性函数得到的实际值相比，是否有足够的相似度。

具体来说，如何选择这几个点的方法就叫做不敏变换。这几个点选择之后对均值和方差的影响不大，被选择的点就称为 sigma 点。至于如何选择这几个点，许多研究者也已给出了答案，在本节中先来介绍几类不敏变换。

再重述一下问题：已知均值为 $\overline{\boldsymbol{x}}$、协方差为 $\boldsymbol{P}$ 的 n 维向量 $\boldsymbol{x}$ 和非线性函数 $\boldsymbol{z}=h(\boldsymbol{x})$，想估计 $\boldsymbol{z}$ 的均值 $\overline{\boldsymbol{y}}$ 和 $\boldsymbol{P}_z$，按照如下方式选取 $2n$ 个 sigma 点 $x^{(i)}$：

$$\begin{cases} \boldsymbol{x}^{(i)}=\overline{\boldsymbol{x}}+\tilde{\boldsymbol{x}}^{(i)}, & i=1,2,\cdots,2n \\ \tilde{\boldsymbol{x}}^{(i)}=(\sqrt{n\boldsymbol{P}})_i^{\mathrm{T}}, & i=1,2,\cdots,n \\ \tilde{\boldsymbol{x}}^{(n+i)}=-(\sqrt{n\boldsymbol{P}})_i^{\mathrm{T}}, & i=1,2,\cdots,n \end{cases} \tag{5-39}$$

式中，$\sqrt{n\boldsymbol{P}}$ 是 $n\boldsymbol{P}$ 的矩阵平方根，即 $(\sqrt{n\boldsymbol{P}})^{\mathrm{T}}(\sqrt{n\boldsymbol{P}})=n\boldsymbol{P}$，$(\sqrt{n\boldsymbol{P}})_i$ 是 $(\sqrt{n\boldsymbol{P}})$ 的第 i 行。接下来计算 sigma 点经过非线性变换的结果：

$$\boldsymbol{z}^{(i)}=h(\boldsymbol{x}^{(i)}),\quad i=1,2,\cdots,2n \tag{5-40}$$

则 $\boldsymbol{z}$ 的均值 $\overline{\boldsymbol{z}}$ 和协方差的近似如下：

$$\begin{cases} \overline{\boldsymbol{z}}=\dfrac{1}{2n}\displaystyle\sum_{i=1}^{2n}\boldsymbol{y}^{(i)} \\ \boldsymbol{P}_{(z)}=\dfrac{1}{2n}\displaystyle\sum_{i=1}^{2n}(\boldsymbol{z}^{(i)}-\overline{\boldsymbol{z}})(\boldsymbol{z}^{(i)}-\overline{\boldsymbol{z}})^{\mathrm{T}} \end{cases} \tag{5-41}$$

与前面按照方差选取 300 个点不同，现在先来分析一下这 2n 个点的选取方式。根据方差的大小，“精心”挑选 2n 个点，而且这些点均值为 $\bar{\boldsymbol{x}}$，协方差为 $\boldsymbol{P}$，那么这 2n 个点经过非线性函数后，$\bar{\boldsymbol{y}}$ 和 $\boldsymbol{P}_z$ 与真实值是否接近呢？通过下面具体的例子，给出求具体问题 sigma 点的 MATLAB 程序。

例 5.2　考虑式（5-21）的非线性变换。因为有两个独立变量（r 和 θ），所以 $n=2$。x 的协方差是 $\boldsymbol{P}=\begin{bmatrix}\delta_r^2 & \\ & \delta_\theta^2\end{bmatrix}$，则按照式（5-39）计算 sigma 点如下：

$$\begin{cases}\boldsymbol{x}^{(1)}=\bar{\boldsymbol{x}}+(\sqrt{2\boldsymbol{P}})_1^{\mathrm{T}}=\begin{bmatrix}1\\ \pi/2\end{bmatrix}+\begin{bmatrix}\sqrt{2}\delta_r\\ 0\end{bmatrix}=\begin{bmatrix}1+\sqrt{2}\delta_r\\ \pi/2\end{bmatrix}\\ \boldsymbol{x}^{(2)}=\bar{\boldsymbol{x}}+(\sqrt{2\boldsymbol{P}})_2^{\mathrm{T}}=\begin{bmatrix}1\\ \pi/2\end{bmatrix}+\begin{bmatrix}0\\ \sqrt{2}\delta_\theta\end{bmatrix}=\begin{bmatrix}1\\ \pi/2+\sqrt{2}\delta_\theta\end{bmatrix}\\ \boldsymbol{x}^{(3)}=\bar{\boldsymbol{x}}-(\sqrt{2\boldsymbol{P}})_1^{\mathrm{T}}=\begin{bmatrix}1-\sqrt{2}\delta_r\\ \pi/2\end{bmatrix}\\ \boldsymbol{x}^{(4)}=\bar{\boldsymbol{x}}-(\sqrt{2\boldsymbol{P}})_2^{\mathrm{T}}=\begin{bmatrix}1\\ \pi/2-\sqrt{2}\delta_\theta\end{bmatrix}\end{cases}\tag{5-42}$$

计算 sigma 点经过非线性变换 $\boldsymbol{z}^{(i)}=h(\boldsymbol{x}^{(i)})$ 得

$$\begin{cases}\boldsymbol{z}^{(1)}=\begin{bmatrix}x_1^{(1)}\cos x_2^{(1)}\\ x_1^{(1)}\sin x_2^{(1)}\end{bmatrix}=\begin{bmatrix}(1+\sqrt{2}\delta_r)\cos\dfrac{\pi}{2}\\ (1+\sqrt{2}\delta_r)\sin\dfrac{\pi}{2}\end{bmatrix}=\begin{bmatrix}0\\ 1+\sqrt{2}\delta_r\end{bmatrix}\\ \boldsymbol{z}^{(2)}=\begin{bmatrix}x_1^{(1)}\cos x_2^{(2)}\\ x_1^{(2)}\sin x_2^{(2)}\end{bmatrix}=\begin{bmatrix}1\times\cos\left(\dfrac{\pi}{2}+\sqrt{2}\delta_\theta\right)\\ 1\times\sin\left(\dfrac{\pi}{2}+\sqrt{2}\delta_\theta\right)\end{bmatrix}=\begin{bmatrix}\cos\left(\dfrac{\pi}{2}+\sqrt{2}\delta_\theta\right)\\ \sin\left(\dfrac{\pi}{2}+\sqrt{2}\delta_\theta\right)\end{bmatrix}\\ \boldsymbol{z}^{(3)}=\begin{bmatrix}x_1^{(3)}\cos x_2^{(3)}\\ x_1^{(3)}\sin x_2^{(3)}\end{bmatrix}=\begin{bmatrix}0\\ 1-\sqrt{2}\delta_r\end{bmatrix}\\ \boldsymbol{z}^{(4)}=\begin{bmatrix}x_1^{(4)}\cos x_2^{(4)}\\ x_1^{(4)}\sin x_2^{(4)}\end{bmatrix}=\begin{bmatrix}\cos\left(\dfrac{\pi}{2}-\sqrt{2}\delta_\theta\right)\\ \sin\left(\dfrac{\pi}{2}-\sqrt{2}\delta_\theta\right)\end{bmatrix}\end{cases}\tag{5-43}$$

则均值为

$$\overline{\boldsymbol{z}}=\frac{1}{2n}\sum_{i=1}^{4}z^{(i)}=\frac{1}{4}\begin{bmatrix}\cos\left(\frac{\pi}{2}+\sqrt{2}\delta_\theta\right)+\cos\left(\frac{\pi}{2}-\sqrt{2}\delta_\theta\right)\\ 2+\sin\left(\frac{\pi}{2}+\sqrt{2}\delta_\theta\right)+\sin\left(\frac{\pi}{2}-\sqrt{2}\delta_\theta\right)\end{bmatrix} \tag{5-44}$$

方差为

$$\boldsymbol{P}_z=\frac{1}{4}\sum_{i=1}^{4}(\boldsymbol{z}^{(i)}-\overline{\boldsymbol{z}})(\boldsymbol{z}^{(i)}-\overline{\boldsymbol{z}})^{\mathrm{T}} \tag{5-45}$$

下面给出 MATLAB 程序实现式（5-42）～式（5-45）的不敏变换过程。

```
%下面四句完成式（5-42），求 sigma 点计算。
clc
clear
b=0.01;d=0.35;
x(:,1)=[1+sqrt(2)*b;pi/2];
x(:,2)=[1;pi/2+sqrt(2)*d];
x(:,3)=[1-sqrt(2)*b;pi/2];
x(:,4)=[1;pi/2-sqrt(2)*d];

 %下面利用循环完成式（5-43），求非线性 z^(i) 。
for i=1:4
   z(:,i)=[x(1,i)*cos(x(2,i));x(1,i)*sin(x(2,i))];
end
%画出所求非线性变量。
hold on
plot(z(1,1),z(2,1),'ko','MarkerSize',8,'MarkerFaceColor',[.5 1 .3]);
plot(z(1,2),z(2,2),'ko','MarkerSize',8,'MarkerFaceColor',[.5 1 .3]);
plot(z(1,3),z(2,3),'ko','MarkerSize',8,'MarkerFaceColor',[.5 1 .3]);
plot(z(1,4),z(2,4),'ko','MarkerSize',8,'MarkerFaceColor',[.5 1 .3]);

%根据式（5-44）求平均值，再把均值画出来。
sigmamean=sum(z,2)/4;
plot(sigmamean(1),sigmamean(2),'^','MarkerSize',8,'MarkerFaceColor',
[.5 1 .3])
 %根据式（5-45）求方差。
Pz=zeros(2);
 for i=1:4
Pz=Pz+(z(:,i)-sigmamean)*(z(:,i)-sigmamean)';
 end
```

```
%将求得的方差值求平方后，再画出一个范围，是一个椭圆。
Pz=sqrt(Pz/4);
rectangle('Position',[sigmamean(1)-Pz(1,1),sigmamean(2)-Pz(2,2),Pz(1
,1)*2,Pz(2,2)*2],'Curvature',[1,1],'LineWidth',3)
```

不敏变换的结果如图 5-3 所示。

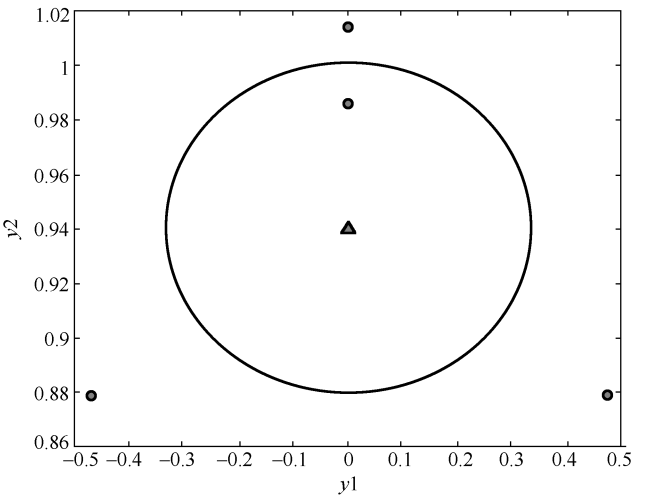


图 5-3　无迹变换的结果

程序中先用四条语句给出了 $\boldsymbol{x}^{(i)}$，为使程序运行方便，这四个 sigma 点被排成一个矩阵，每一列分别代表一个 sigma 点。然后用 4 次 for 循环，产生每一个 sigma 点的非线性变换 $\boldsymbol{z}^{(i)}$。再使用 mean 函数得到 sigma 点的均值 sigmamean，用 plot 语句将 sigma 点经过非线性变换得到四个点，用圆点画在图中。从图中可看出四个点位置比较特殊，左下角和右下角的圆点由于方差 $\sqrt{2}\times 0.35$ 的作用，两个点几乎离开了原来 300 个点群。$y_1=0$ 点由于方差 $\sqrt{2}\times 0.01$ 的作用，这两个圆点在所有点的上下边界附近。所化的大圆就是求得的方差，小三角是均值。求方差的程序是经过 sigma 点的非线性变换获得 y 值，并与其均值相减，然后求平方和再乘以权值。此时的权值是 0.25，将所得结果开方作为椭圆的半径。图中黑色大圆圈就是用四个 sigma 点经过非线性变换获得方差。观察图 5-2 中的非线性变换的真实均值和协方差，发现图 5-3 中四个 sigma 点的均值和图 5-2 中的真实值（即星号表示的 300 个点的均值）相差依然较大。那么图 5-3 中“小三角”具体代表什么，下面将会讲到。

对于非线性变换，将 $\boldsymbol{y}=\boldsymbol{h}(\boldsymbol{x})$ 围绕 $\bar{\boldsymbol{x}}$ 用二阶泰勒级数展开：

$$\boldsymbol{y}\approx \boldsymbol{h}(\boldsymbol{x})=\boldsymbol{h}(\bar{\boldsymbol{x}})+D_{\tilde{\boldsymbol{x}}}\boldsymbol{h}+\frac{1}{2!}D_{\tilde{\boldsymbol{x}}}^{2}\boldsymbol{h} \tag{5-46}$$

代入非线性关系式（5-21）后，得

$$\begin{aligned}\overline{\boldsymbol{y}} &\approx h(\overline{\boldsymbol{x}})+\frac{1}{2!}E(D_{\tilde{x}}^2 h)\\ &=\begin{bmatrix}0\\1\end{bmatrix}+\frac{1}{2}\delta_\theta^2\begin{bmatrix}1\\-1\end{bmatrix}=\begin{bmatrix}0\\0.9387\end{bmatrix}\end{aligned}$$

可以看到，图 5-3 中无迹变换的结果所获得的“小三角”其实是使用二阶泰勒展开之后所获得的均值，这两者是一致的。即经过 sigma 变换可以把 y 的均值匹配到真实均值的三阶，而线性化均值只使用式（5-46）的第一项，仅能匹配到真实均值的一阶。所以说，通过 sigma 变换可以将均值向真实值进一步逼近。sigma 变换的方差也是一样的，向非线性变换的方差进一步逼近，感兴趣的读者可以参见相关参考文献。

必须指出，这样无迹变换得到的值并不与真值相等，而是向真值逼近。尽管这样，依然认为无迹变换是很有潜力的，和原来 EKF 的线性化相比，计算显然简单很多。这里选择四个点，就可以逼近真实值的三阶泰勒展开，与需要经过求偏导的 EKF 相比，把均值和协方差都向前推进了两阶，原来的 EKF 只是逼近到一阶。不敏变换还有其他类型，主要包括以下几种。

1. 一般型不敏变换

这种变换选取 2n+1 个 sigma 点，并且取值的选择和前面也有所不同。

$$\begin{cases}\boldsymbol{x}^{(0)}=\overline{\boldsymbol{x}}\\ \boldsymbol{x}^{(i)}=\overline{\boldsymbol{x}}+\tilde{\boldsymbol{x}}^{(i)}, & i=1,2,\cdots,2n\\ \tilde{\boldsymbol{x}}^{(i)}=(\sqrt{(n+k)\boldsymbol{P}})_i^{\mathrm{T}}, & i=1,2,\cdots,n\\ \tilde{\boldsymbol{x}}^{(i)}=-(\sqrt{(n+k)\boldsymbol{P}})_i^{\mathrm{T}}, & i=1,2,\cdots,n\end{cases}\tag{5-47}$$

2n+1 个加权系数为

$$\begin{cases}w^{(0)}=\dfrac{k}{n+k}\\ w^{(i)}=\dfrac{1}{2(n+k)}, & i=1,2,\cdots,2n\end{cases}\tag{5-48}$$

经过非线性变换：

$$\boldsymbol{z}^{(i)}=h(\boldsymbol{x}^{(i)})\tag{5-49}$$

均值和协方差分别为

$$\begin{cases}\overline{\boldsymbol{z}}=\sum\limits_{i=0}^{2n}w^{(i)}\boldsymbol{z}^{(i)}\\ \boldsymbol{P}_{\mathrm{z}}=\sum\limits_{i=0}^{2n}w^{(i)}(\boldsymbol{z}^{(i)}-\overline{\boldsymbol{z}})(\boldsymbol{z}^{(i)}-\overline{\boldsymbol{z}})^{\mathrm{T}}\end{cases}\tag{5-50}$$

该变换的 MATLAB 程序如下：

```
clc
clear
b=0.01;d=0.35;

k=1;n=2;
x(:,1)=[1+sqrt(k+n)*b;pi/2];
x(:,2)=[1;pi/2+sqrt(k+n)*d];
x(:,3)=[1-sqrt(k+n)*b;pi/2];
x(:,4)=[1;pi/2-sqrt(k+n)*d];
x0=[1,pi/2];

nn=4;
w(1:nn)=1/(2*(n+k));
w0=k/(k+n);

y0=[x0(1)*cos(x0(2));x0(1)*sin(x0(2))];
for i=1:nn
     y(:,i)=[x(1,i)*cos(x(2,i));x(1,i)*sin(x(2,i))];
end

sigmamean=w0*y0;
for i=1:nn
sigmamean=sigmamean+w(i)*y(:,i);
end

hold on

plot(y0(1),y0(2),'ko','MarkerSize',8,…
'MarkerFaceColor',[.5 0.10 .9]);
plot(y(1,1),y(2,1),'ko','MarkerSize',8,…
'MarkerFaceColor',[.5 0.10 .9]);
plot(y(1,2),y(2,2),'ko','MarkerSize',8,…
'MarkerFaceColor',[.5 0.10 .9]);
plot(y(1,3),y(2,3),'ko','MarkerSize',8,…
'MarkerFaceColor',[.5 0.10 .9]);
plot(y(1,4),y(2,4),'ko','MarkerSize',8,…
'MarkerFaceColor',[.5 0.10 .9]);

plot(sigmamean(1),sigmamean(2),'^','MarkerSize',…
8,'MarkerFaceColor',[.5 1 .3])
```

```
Py=w0*(y0-sigmamean)*(y0-sigmamean)';
for i=1:4
Py=Py+w(i)*(y(:,i)-sigmamean)*(y(:,i)-sigmamean)';
end
Py=sqrt(Py);
rectangle('Position',[sigmamean(1)-Py(1,1),…
sigmamean(2)-Py(2,2),Py(1,1)*2,Py(2,2)*2],…
'Curvature',[1,1],'LineWidth',3)
```

这一段程序和前面的非常相似，同样给出了几个 sigma 点。这里有五个 sigma 点，还包括一个 x_0，而且根号下的 n 变为 $k+n$。k 现在取值为 1，依次计算五个 sigma 点经过非线性变换，最后通过权值求取均值，同时求出方差。在图 5-4 上可以清楚地看到这五个点的分布（增加了（0,1）点），求得的结果和图 5-3 不敏变换的均值是完全一样的，图上的小三角依然在原来的位置。但是由于式中的 $\sqrt{2}$ 变为 $\sqrt{3}$，所以左下角和右下角两个点向更远方伸展，这样求得的方差就比原来大，可以看到大圆变得更大了，但其均值并没有改变。

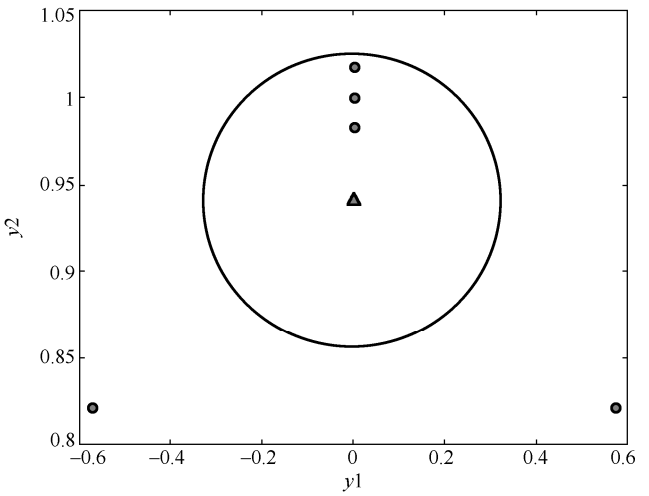


图 5-4　一般型不敏变换的均值与协方差

2. 球型不敏变换

球型不敏变换略微复杂一些，有以下几个步骤。

（1）选择权系数 $w^{(0)} \in [0,1)$。

（2）选择其余的权系数：

$$w^{(i)} = \frac{1-w^{(i)}}{n+1}, \quad i = 1,2,\cdots,n+1 \tag{5-51}$$

（3）初始化线一元向量：

$$\begin{cases}\boldsymbol{\delta}_0^{(1)}=0\\ \boldsymbol{\delta}_1^{(1)}=\dfrac{-1}{\sqrt{2w^{(1)}}}\\ \boldsymbol{\delta}_2^{(1)}=\dfrac{1}{\sqrt{2w^{(1)}}}\end{cases} \tag{5-52}$$

对于 $j=2,\cdots,n$，按照式（5-53）递推获得向量 $\boldsymbol{\delta}$：

$$\boldsymbol{\delta}_i^{(i)}=\begin{cases}\begin{bmatrix}\boldsymbol{\delta}_0^{(j-1)}\\ 0\end{bmatrix}, & i=0\\ \begin{bmatrix}\boldsymbol{\delta}_i^{(j-1)}\\ \dfrac{-1}{\sqrt{j(j+1)w^{(1)}}}\end{bmatrix}, & i=1,\cdots,j\\ \begin{bmatrix}O_j\\ \dfrac{j}{\sqrt{j(j+1)w^{(1)}}}\end{bmatrix}, & i=j+1\end{cases} \tag{5-53}$$

式中，O_j 是包含 j 个零的列向量。

（4）上面的递推完成后，得到 n 维向量 $\boldsymbol{\delta}_i^{(n)}(i=0,1,\cdots,n+1)$。球形不敏变换的 sigma 点为

$$\boldsymbol{x}^{(i)}=\bar{\boldsymbol{x}}+\sqrt{\boldsymbol{P}_x}\boldsymbol{\delta}_i^{(n)},\quad i=0,1\cdots,n+1 \tag{5-54}$$

式中，$\bar{\boldsymbol{x}}$ 是状态 $\boldsymbol{x}$ 的均值，$\boldsymbol{P}_x$ 为其方差。

（5）非线性变换及均值、协方差的求法同式（5-49）与式（5-50）。

球形不敏变换有点复杂，下面通过 5.2.1 节的例子来看一下球形不敏变换具体的实现过程。

（1）选择权系数 $w^{(0)}=0$。

（2）由于 n=2，即状态向量分别为 $\boldsymbol{r}$ 和 $\boldsymbol{\theta}$，所以其余的权系数为

$$w^{(1)}=w^{(2)}=w^{(3)}=\frac{1-w^{(0)}}{n+1} \tag{5-55}$$

（3）初始化下面的一元向量：

$$\begin{cases}\boldsymbol{\delta}_0^{(1)}=0\\ \boldsymbol{\delta}_1^{(1)}=\dfrac{1}{\sqrt{2w^{(1)}}}\\ \boldsymbol{\delta}_2^{(1)}=\dfrac{1}{\sqrt{2w^{(1)}}}\end{cases} \tag{5-56}$$

（4）对于 $j=2$，得向量 $\boldsymbol{\delta}$：

$$\begin{cases} \boldsymbol{\delta}_0^{(2)}=\begin{bmatrix}\boldsymbol{\delta}_0^{(1)}\\0\end{bmatrix}=\begin{bmatrix}0\\0\end{bmatrix}, & i=0\\ \boldsymbol{\delta}_1^{(2)}=\begin{bmatrix}\boldsymbol{\delta}_1^{(1)}\\\dfrac{-1}{\sqrt{6w^{(1)}}}\end{bmatrix}=\begin{bmatrix}\dfrac{1}{\sqrt{2w^{(1)}}}\\\dfrac{-1}{\sqrt{6w_{(1)}}}\end{bmatrix}, & i=1\\ \boldsymbol{\delta}_2^{(2)}=\begin{bmatrix}\boldsymbol{\delta}_2^{(1)}\\\dfrac{-1}{\sqrt{6w^{(1)}}}\end{bmatrix}=\begin{bmatrix}\dfrac{1}{\sqrt{2w^{(1)}}}\\\dfrac{-1}{\sqrt{6w_{(1)}}}\end{bmatrix}, & i=2\\ \boldsymbol{\delta}_3^{(2)}=\begin{bmatrix}0\\\dfrac{2}{\sqrt{6w^{(1)}}}\end{bmatrix}, & i=3\end{cases} \tag{5-57}$$

（5）则 sigma 点为

$$\begin{aligned}\boldsymbol{x}^{(i)}&=\bar{\boldsymbol{x}}+\sqrt{\boldsymbol{P}}\boldsymbol{\delta}_i^{(2)}\\&=\begin{bmatrix}1\\\dfrac{\pi}{2}\end{bmatrix}+\begin{bmatrix}0.01&0\\0&0.35\end{bmatrix}\boldsymbol{\delta}_i^{(2)}\end{aligned} \tag{5-58}$$

即

$$\begin{cases} \boldsymbol{x}^{(0)}=\begin{bmatrix}1\\\dfrac{\pi}{2}\end{bmatrix}\\ \boldsymbol{x}^{(1)}=\begin{bmatrix}1\\\dfrac{\pi}{2}\end{bmatrix}+\begin{bmatrix}0.01&0\\0&0.35\end{bmatrix}\begin{bmatrix}\dfrac{1}{\sqrt{2w^{(1)}}}\\\dfrac{-1}{\sqrt{6w^{(1)}}}\end{bmatrix}=\begin{bmatrix}1+\dfrac{0.01}{\sqrt{2w^{(1)}}}\\\dfrac{\pi}{2}+\dfrac{-0.35}{\sqrt{6w^{(1)}}}\end{bmatrix}\\ \boldsymbol{x}^{(2)}=\begin{bmatrix}1\\\dfrac{\pi}{2}\end{bmatrix}+\begin{bmatrix}0.01&0\\0&0.35\end{bmatrix}\begin{bmatrix}\dfrac{1}{\sqrt{2w^{(1)}}}\\\dfrac{-1}{\sqrt{6w^{(1)}}}\end{bmatrix}=\begin{bmatrix}1+\dfrac{0.01}{\sqrt{2w^{(1)}}}\\\dfrac{\pi}{2}+\dfrac{-0.35}{\sqrt{6w^{(1)}}}\end{bmatrix}\\ \boldsymbol{x}^{(3)}=\begin{bmatrix}1\\\dfrac{\pi}{2}\end{bmatrix}+\begin{bmatrix}0.01&0\\0&0.35\end{bmatrix}\begin{bmatrix}0\\\dfrac{2}{\sqrt{6w^{(1)}}}\end{bmatrix}=\begin{bmatrix}1\\\dfrac{\pi}{2}+\dfrac{7}{\sqrt{6w^{(1)}}}\end{bmatrix}\end{cases} \tag{5-59}$$

（6）再利用非线性关系求出 $\boldsymbol{z}^{(i)}$，进而求出 $\boldsymbol{z}$ 的均值及方差。

为方便读者理解，给出如下 MATLAB 程序来实现式（5-55）～式（5-59）的球型不敏变换：

```
clc
clear
n=2;nn=4;
W0=0;
W(1)=(1-W0)/(n+1);
W(2)=W(1);
W(3)=W(1);

gama(:,1)=[0;0];
gama(:,2)=[-1/sqrt(2*W(1));-1/sqrt(6*W(1))];
gama(:,3)=[1/sqrt(2*W(1));-1/sqrt(6*W(1))];
gama(:,4)=[0;2/sqrt(6*W(1))];

meanx=[1;pi/2];
Px=diag([0.01^2,0.35^2]);
for i=1:4
x(:,i)=meanx+sqrt(Px)*gama(:,i);
end

for i=1:nn
   y(:,i)=[x(1,i)*cos(x(2,i));x(1,i)*sin(x(2,i))];
end

sigmamean=W0*y(:,1);
for i=1:nn-1
sigmamean=sigmamean+W(i)*y(:,i+1);
end

hold on
plot(y(1,1),y(2,1),'ko','MarkerSize',8,'MarkerFaceColor',[.5
0.90 .9]);
plot(y(1,2),y(2,2),'ko','MarkerSize',8,'MarkerFaceColor',[.5
0.90 .9]);
plot(y(1,3),y(2,3),'ko','MarkerSize',8,'MarkerFaceColor',[.5
0.90 .9]);
plot(y(1,4),y(2,4),'ko','MarkerSize',8,'MarkerFaceColor',[.5
0.90 .9]);
```

```
plot(sigmamean(1),sigmamean(2),'^','MarkerSize',8,'MarkerFaceColor',
[.5 1 .3])

Py=W0*(y(:,1)-sigmamean)*(y(:,1)-sigmamean)';
for i=1:3
Py=Py+W(i)*(y(:,i+1)-sigmamean)*(y(:,i+1)-sigmamean)';
end
Py=sqrt(Py);
rectangle('Position',[sigmamean(1)-Py(1,1),sigmamean(2)-Py(2,2),
Py(1,1)*2,Py(2,2)*2],'Curvature',[1,1],'LineWidth',1)
```

先来看看程序，这段程序和一般型不敏变换的程序相似，只是略微复杂一些。首先需要产生四个 sigma 点，即式（5-57）中的四个式子，在程序中用 gama 表示这四个式子中的 $\boldsymbol{\delta}_i^{(2)}$（$i=0,1,2,3$）；然后设置均值和方差，用 for 循环实现式（5-59）中求 sigma 点的过程，接下来通过非线性变换求均值和方差。

球型不敏变换求取均值与协方差的结果如图 5-5 所示。从图中可以看出，均值还是和原来一样，但有两点不同：一是这四个点不再是对称的；二是如图中所示，圆表示的方差变小，说明现在的方差比原来的小，得到的值会更精确。在程序中把 w_0（$0 \leqslant w_0 \leqslant 1$）的值设置为 0，通过仿真可以看出如果 w_0 的取值变大，其方差表示的圆就变大，但均值还是不变。因此，w_0 一般取为 0，只是用了 3 个 sigma 点，其中 $\boldsymbol{x}^{(0)}$ 这个点没被使用。

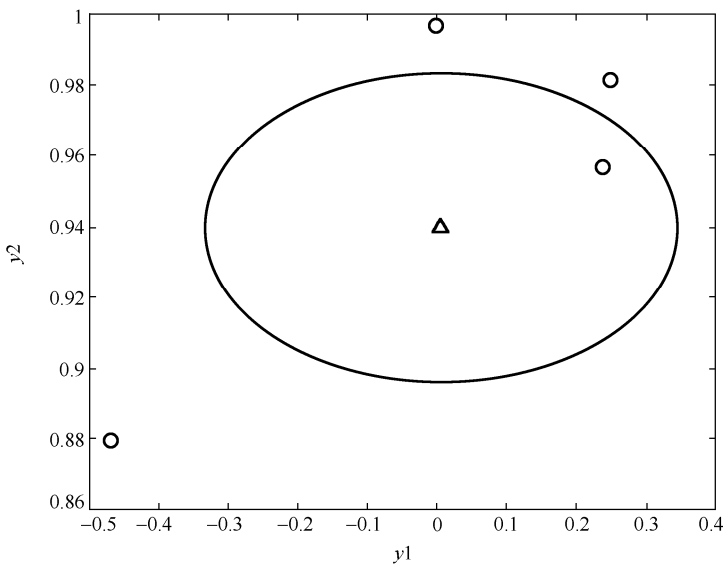


图 5-5　球型不敏变换的均值与协方差

5.2.3　无迹 Kalman 滤波器

通过前面学习知道，Kalman 滤波器的算法是利用更新时间和测量的方法来传播系统均值和协方差的。如果系统是线性的，均值和协方差可以进行准确的更新。如果是非线性的，采用扩展 Kalman 滤波器对均值和协方差更新，结果是非常不准确的，这一点从 5.2.2 节的例子中即可看出。而利用不敏变换可以把均值和协方差向真值逼近到三阶，和二阶的泰勒级数一致。下面，来看一下如何利用不敏变换来修正非线性的 Kalman 滤波器。由于标准的 Kalman 滤波器是通过状态方程和测量方程两次向前预测并更新状态，如果过程方程和状态方程都是非线性的，那么这两次更新状态（一是预测，二是利用测量进行更新）都需要使用不敏变换进行更准确的预测和更新，下面给出具体的 UKF 算法。

（1）已知 n 维离散时间非线性系统：

$$\begin{cases} \boldsymbol{x}(k+1)=f[\boldsymbol{x}(k)]+\boldsymbol{w}(k) \\ \boldsymbol{z}(k)=h[\boldsymbol{x}(k)]+\boldsymbol{v}(k) \\ \boldsymbol{w}(k)\sim[0,\boldsymbol{Q}(k)] \\ \boldsymbol{v}(k)\sim[0,\boldsymbol{R}(k)] \end{cases} \tag{5-60}$$

（2）滤波器初始化如下：

$$\begin{cases} \hat{\boldsymbol{x}}(0\,|\,0)=E(\boldsymbol{x}_0)=\boldsymbol{x}_0 \\ \boldsymbol{P}(0\,|\,0)=\boldsymbol{P}_0 \end{cases} \tag{5-61}$$

和前面情况一样，一般情况下，$\boldsymbol{x}_0$ 可以设为任意向量，只要与状态 $\boldsymbol{x}$ 的维数 n 相同即可，$\boldsymbol{P}_0$ 可以设为 $n\times n$ 的正定对角阵。

（3）状态向前一步预测，步骤如下。

① 利用 $k-1$ 时刻的估计状态，$\hat{\boldsymbol{x}}(k-1\,|\,k-1)$ 即估计方差 $\boldsymbol{P}(k-1\,|\,k-1)$，产生 sigma 点 $\hat{\boldsymbol{x}}^{(i)}(k-1\,|\,k-1)$。sigma 点产生的方法可以使用最简单的不敏变换方法，也可以使用一般型不敏变换式（5-47），或球形不敏变换式（5-52）～式（5-54），这里使用式（5-47）作为例子说明如何使用 $\hat{\boldsymbol{x}}(k-1\,|\,k-1)$ 及 $\boldsymbol{P}(k-1\,|\,k-1)$ 产生向前一步预测的 sigma 点：

$$\begin{cases} \hat{\boldsymbol{x}}^{(i)}(k-1\,|\,k-1)=\hat{\boldsymbol{x}}(k-1\,|\,k-1)+\tilde{\boldsymbol{x}}^{(i)}, & i=1,\cdots,2n \\ \tilde{\boldsymbol{x}}^{(i)}=(\sqrt{n\boldsymbol{P}(k-1\,|\,k-1)})_i^{\mathrm{T}}, & i=1,\cdots,n \\ \tilde{\boldsymbol{x}}^{(n+i)}=-(\sqrt{n\boldsymbol{P}(k-1\,|\,k-1)})_i^{\mathrm{T}}, & i=1,\cdots,n \end{cases} \tag{5-62}$$

式中，$(\sqrt{n\boldsymbol{P}(k-1\,|\,k-1)})_i^{\mathrm{T}}$ 表示 n 乘以估计方差 $\boldsymbol{P}(k-1\,|\,k-1)$ 后，进行矩阵开方运算，选择其中的第 i 行并转置后得到 n 维向量。

② 将 sigma 点 $\hat{\boldsymbol{x}}^{(i)}(k-1\,|\,k-1)$ 代入式（5-60）中的系统状态方程，得

$$\hat{\boldsymbol{x}}^{(i)}(k\,|\,k-1)=f[\hat{\boldsymbol{x}}^{(i)}(k-1\,|\,k-1)] \tag{5-63}$$

③ 合并向量 $\hat{\boldsymbol{x}}^{(i)}(k|k-1)$ 来获得 k 时刻的向前一步状态估计：

$$\hat{\boldsymbol{x}}(k|k-1)=\frac{1}{2n}\sum_{i=1}^{2n}\hat{\boldsymbol{x}}^{(i)}(k|k-1) \tag{5-64}$$

④ 考虑过程噪声，用下面的公式求向前一步预测的估计协方差：

$$\begin{aligned}\boldsymbol{P}(k|k-1)=&\frac{1}{2n}\sum_{i=1}^{2n}[\hat{\boldsymbol{x}}^{(i)}(k|k-1)-\hat{\boldsymbol{x}}(k|k-1)][\hat{\boldsymbol{x}}^{(i)}(k|k-1)-\hat{\boldsymbol{x}}(k|k-1)]^{\mathrm{T}}\\&+\boldsymbol{Q}(k-1)\end{aligned} \tag{5-65}$$

（4）利用测量更新向前一步预测状态，步骤如下。

① 利用向前一步预测状态，$\hat{\boldsymbol{x}}(k|k-1)$ 及其方差 $\boldsymbol{P}(k|k-1)$，先产生 sigma 点 $\hat{\boldsymbol{x}}^{(i)}(k|k-1)$，方法如下：

$$\begin{cases}\hat{\boldsymbol{x}}^{(i)}(k|k-1)=\hat{\boldsymbol{x}}(k|k-1)+\tilde{\boldsymbol{x}}^{(i)}, & i=1,\cdots,2n\\ \tilde{\boldsymbol{x}}^{(i)}=(\sqrt{n\boldsymbol{P}(k|k-1)})_i^{\mathrm{T}}, & i=1,\cdots,n\\ \tilde{\boldsymbol{x}}^{(n+i)}=-(\sqrt{n\boldsymbol{P}(k|k-1)})_i^{\mathrm{T}}, & i=1,\cdots,n\end{cases} \tag{5-66}$$

注意，也可以重复利用刚才产生的那些点（式（5-61）），而不产生新的 sigma 点，这样虽然会损失一定精度，但可以减小计算负担。

② 用已知的非线性测量方差将 sigma 点转换为量测预测 $\hat{\boldsymbol{z}}^{(i)}(k|k-1)$：

$$\hat{\boldsymbol{z}}^{(i)}(k|k-1)=h[\hat{\boldsymbol{x}}^{(i)}(k|k-1)] \tag{5-67}$$

③ 合并向量 $\hat{\boldsymbol{z}}^{(i)}(k|k-1)$ 获得 k 时刻的测量预测：

$$\hat{\boldsymbol{z}}(k|k-1)=\frac{1}{2n}\sum_{i=1}^{2n}\hat{\boldsymbol{z}}^{(i)}(k|k-1) \tag{5-68}$$

④ 考虑测量噪声量测预测的协方差为

$$\boldsymbol{P}_z=\frac{1}{2n}\sum_{i=1}^{2n}[\hat{\boldsymbol{z}}^{(i)}(k|k-1)-\hat{\boldsymbol{z}}(k|k-1)][\hat{\boldsymbol{z}}^{(i)}(k|k-1)-\hat{\boldsymbol{z}}(k|k-1)]^{\mathrm{T}}+\boldsymbol{R}(k) \tag{5-69}$$

⑤ 再来估计 $\hat{\boldsymbol{x}}(k|k-1)$ 与 $\hat{\boldsymbol{z}}(k|k-1)$ 之间的协方差：

$$\boldsymbol{P}_{xz}=\frac{1}{2n}\sum_{i=1}^{2n}[\hat{\boldsymbol{x}}^{(i)}(k|k-1)-\hat{\boldsymbol{x}}(k|k-1)][\hat{\boldsymbol{z}}^{(i)}(k|k-1)-\hat{\boldsymbol{z}}(k|k-1)]^{\mathrm{T}} \tag{5-70}$$

（5）利用 Kalman 滤波方程进行估计，先计算滤波增益 $\boldsymbol{K}(k)=\boldsymbol{P}_{xz}\boldsymbol{P}_z^{-1}$，则更新的状态估计及方差为

$$\hat{\boldsymbol{x}}(k|k)=\hat{\boldsymbol{x}}(k|k-1)+\boldsymbol{K}(k)[\boldsymbol{z}(k)-\hat{\boldsymbol{z}}(k|k-1)] \tag{5-71}$$

$$\boldsymbol{P}(k+1|k+1)=\boldsymbol{P}(k|k+1)-\boldsymbol{K}(k)\boldsymbol{P}_z\boldsymbol{K}^{\mathrm{T}}(k) \tag{5-72}$$

上述算法中假设过程方程和测量方程相对于噪声是线性的，而过程和测量方程相对于噪声是非线性的公式如下：

$$\begin{cases} \boldsymbol{x}(k+1) = f[\boldsymbol{x}(k), \boldsymbol{w}(k)] \\ \boldsymbol{z}(k) = h[\boldsymbol{x}(k), \boldsymbol{v}(k)] \end{cases} \tag{5-73}$$

需要将噪声扩充到状态向量中，状态向量变为

$$\boldsymbol{x}_a(k) = \begin{bmatrix} \boldsymbol{x}(k) \\ \boldsymbol{w}(k) \\ \boldsymbol{v}(k) \end{bmatrix} \tag{5-74}$$

然后用 UKF 估计扩充的状态 $\boldsymbol{x}_a(k)$，UKF 初始化为

$$\hat{\boldsymbol{x}}_a = \begin{bmatrix} E(\boldsymbol{x}_0) \\ 0 \\ 0 \end{bmatrix} = \begin{bmatrix} \hat{\boldsymbol{x}}(0 \mid 0) \\ 0 \\ 0 \end{bmatrix} \tag{5-75}$$

$$\boldsymbol{P}_a(0 \mid 0) = \begin{bmatrix} E[(\boldsymbol{x} - \hat{\boldsymbol{x}}(0 \mid 0))(\boldsymbol{x} - \hat{\boldsymbol{x}}(0 \mid 0))^{\mathrm{T}}] & 0 & 0 \\ 0 & \boldsymbol{Q}_0 & 0 \\ 0 & 0 & \boldsymbol{R}_0 \end{bmatrix} \tag{5-76}$$

利用 UKF 算法估计的是扩充的均值和协方差，因此，需要从式（5-65）和式（5-69）中移除 $\boldsymbol{Q}(k-1)$ 和 $\boldsymbol{R}(k)$。

如表 5-2 所示，给出了标准 Kalman 滤波器和 UKF 的差别。

表 5-2　标准 Kalman 滤波器和 UKF 的差别

<table>
<tr><th></th><th>标准 Kalman 滤波器</th><th colspan="2">UKF</th></tr>
<tr><td rowspan="2">系统模型描述</td><td rowspan="2">$\boldsymbol{x}(k+1) = \boldsymbol{A}(k)\boldsymbol{x}(k) + \boldsymbol{w}(k)$
$\boldsymbol{z}(k) = \boldsymbol{C}(k)\boldsymbol{x}(k) + \boldsymbol{v}(k)$
$E[\boldsymbol{w}(k)\boldsymbol{w}^{\mathrm{T}}(j)] = \boldsymbol{Q}(k)\boldsymbol{\delta}(k-j)$
$E[\boldsymbol{v}(k)\boldsymbol{v}^{\mathrm{T}}(j)] = \boldsymbol{R}(k)\boldsymbol{\delta}(k-j)$
$E[\boldsymbol{w}(k)\boldsymbol{v}^{\mathrm{T}}(j)] = 0$</td><td colspan="2">$\boldsymbol{x}(k) = f[\boldsymbol{x}(k-1), \boldsymbol{w}(k-1)]$
$\boldsymbol{z}(k) = h[\boldsymbol{x}(k), \boldsymbol{v}(k)]$
$E[\boldsymbol{w}(k)\boldsymbol{w}^{\mathrm{T}}(j)] = \boldsymbol{Q}(k)\boldsymbol{\delta}(k-j)$
$E[\boldsymbol{v}(k)\boldsymbol{v}^{\mathrm{T}}(j)] = \boldsymbol{R}(k)\boldsymbol{\delta}(k-j)$
$E[\boldsymbol{w}(k)\boldsymbol{v}^{\mathrm{T}}(j)] = 0$</td></tr>
<tr><td>设置状态初值</td><td>$\boldsymbol{x}_a(k) = \begin{bmatrix} \boldsymbol{x}(k) \\ \boldsymbol{w}(k) \\ \boldsymbol{v}(k) \end{bmatrix}$
$\hat{\boldsymbol{x}}_a = \begin{bmatrix} E(x_0) \\ 0 \\ 0 \end{bmatrix} = \begin{bmatrix} \hat{\boldsymbol{x}}(0 \mid 0) \\ 0 \\ 0 \end{bmatrix}$
$\boldsymbol{P}_a(0 \mid 0) = \begin{bmatrix} E[(\boldsymbol{x} - \hat{\boldsymbol{x}}(0 \mid 0))(\boldsymbol{x} - \hat{\boldsymbol{x}}(0 \mid 0))^{\mathrm{T}}] & 0 & 0 \\ 0 & \boldsymbol{Q}_0 & 0 \\ 0 & 0 & \boldsymbol{R}_0 \end{bmatrix}$</td></tr>
</table>

续表

	标准 Kalman 滤波器		UKF
Kalman 滤波器	预测	$\hat{\boldsymbol{x}}(k\mid k-1)=\boldsymbol{A}(k-1)\hat{\boldsymbol{x}}(k-1\mid k-1)$	$\hat{\boldsymbol{x}}^{(i)}(k-1\mid k-1)=\hat{\boldsymbol{x}}(k-1\mid k-1)+\tilde{\boldsymbol{x}}^{(i)},\quad i=1,\cdots,2n$ $\tilde{\boldsymbol{x}}^{(i)}=(\sqrt{n\boldsymbol{P}(k-1\mid k-1)})_i^{\mathrm{T}},\quad i=1,\cdots,n$ $\tilde{\boldsymbol{x}}^{(n+i)}=-(\sqrt{n\boldsymbol{P}(k-1\mid k-1)})_i^{\mathrm{T}},\quad i=1,\cdots,n$ $\hat{\boldsymbol{x}}^{(i)}(k\mid k-1)=f[\hat{\boldsymbol{x}}^{(i)}(k-1\mid k-1)]$ $\hat{\boldsymbol{x}}(k\mid k-1)=\frac{1}{2n}\sum_{i=1}^{2n}\hat{\boldsymbol{x}}^{(i)}(k\mid k-1)$
	更新	$\hat{\boldsymbol{x}}(k\mid k)=\hat{\boldsymbol{x}}(k\mid k-1)+\boldsymbol{K}(k)[\boldsymbol{z}(k)-\boldsymbol{C}(k)\hat{\boldsymbol{x}}(k\mid k-1)]$	$\hat{\boldsymbol{x}}^{(i)}(k\mid k-1)=\hat{\boldsymbol{x}}(k\mid k-1)+\tilde{\boldsymbol{x}}^{(i)},\quad i=1,\cdots,2n$ $\tilde{\boldsymbol{x}}^{(i)}=(\sqrt{n\boldsymbol{P}(k\mid k-1)})_i^{\mathrm{T}},\quad i=1,\cdots,n$ $\tilde{\boldsymbol{x}}^{(n+i)}=-(\sqrt{n\boldsymbol{P}(k\mid k-1)})_i^{\mathrm{T}},\quad i=1,\cdots,n$ $\hat{\boldsymbol{z}}^{(i)}(k\mid k-1)=h[\hat{\boldsymbol{x}}^{(i)}(k\mid k-1)]$ $\hat{\boldsymbol{z}}(k\mid k-1)=\frac{1}{2n}\sum_{i=1}^{2n}\hat{\boldsymbol{z}}^{(i)}(k\mid k-1)$ $\hat{\boldsymbol{x}}(k\mid k)=\hat{\boldsymbol{x}}(k\mid k-1)+\boldsymbol{K}(k)[\boldsymbol{z}(k)-\hat{\boldsymbol{z}}(k\mid k-1)]$
	滤波器增益	$\boldsymbol{K}(k)=\boldsymbol{P}(k\mid k-1)\boldsymbol{C}^{\mathrm{T}}(k)\times[\boldsymbol{C}^{\mathrm{T}}(k)\boldsymbol{P}(k\mid k-1)\boldsymbol{C}(k)+\boldsymbol{R}(k)]^{-1}$	$\boldsymbol{P}_z=\frac{1}{2n}\sum_{i=1}^{2n}[\hat{\boldsymbol{z}}^{(i)}(k\mid k-1)-\hat{\boldsymbol{z}}(k\mid k-1)][\hat{\boldsymbol{z}}^{(i)}(k\mid k-1)-\hat{\boldsymbol{z}}(k\mid k-1)]^{\mathrm{T}}$ $\boldsymbol{P}_{xz}=\frac{1}{2n}\sum_{i=1}^{2n}[\hat{\boldsymbol{x}}^{(i)}(k\mid k-1)-\hat{\boldsymbol{x}}(k\mid k-1)][\hat{\boldsymbol{z}}^{(i)}(k\mid k-1)-\hat{\boldsymbol{z}}(k\mid k-1)]^{\mathrm{T}}$ $\boldsymbol{K}(k)=\boldsymbol{P}_{xz}\boldsymbol{P}_z^{-1}$
	向前一步预测方差	$\boldsymbol{P}(k\mid k-1)=\boldsymbol{A}(k-1)\boldsymbol{P}(k-1\mid k-1)\boldsymbol{A}^{\mathrm{T}}(k-1)+\boldsymbol{Q}(k-1)$	$\boldsymbol{P}(k\mid k-1)=\frac{1}{2n}\sum_{i=1}^{2n}[\hat{\boldsymbol{x}}^{(i)}(k\mid k-1)-\hat{\boldsymbol{x}}(k\mid k-1)][\hat{\boldsymbol{x}}^{(i)}(k\mid k-1)-\hat{\boldsymbol{x}}(k\mid k-1)]^{\mathrm{T}}$
	状态估计方差	$\boldsymbol{P}(k\mid k)=[\boldsymbol{I}-\boldsymbol{K}(k)\boldsymbol{C}(k)]\times\boldsymbol{P}(k\mid k-1)$	$\boldsymbol{P}(k+1\mid k+1)=\boldsymbol{P}(k+1\mid k+1)-\boldsymbol{K}(k)\boldsymbol{P}_z\boldsymbol{K}^{\mathrm{T}}(k)$

5.3　本 章 小 结

对于含有非线性过程函数及测量函数的系统，原来的只能处理线性系统的标准 Kalman 滤波器无法使用，因此，本章给出了几种处理方法。首先，研究了 EKF，通过一阶泰勒级数将非线性函数线性化。接下来研究了不敏变换以及由此得出的 UKF，并且，UKF 比 EKF 好一些，可以将非线性近似地逼近到三阶。

在仿真时选择 300 个点，求得的均值和真值很接近。由此想到，如果选择更多的点，而不仅仅是 2*n* 个 sigma 点，可以逼近更高阶。因此，在已知系统的非线性函数和噪声分布的情况下，使用更多的点可以逼近实际的均值和方差，但同时意味着要进行大量的计算。另一方面，要产生这些点，也要求必须已知非线性关系和噪声的分布情况。

更特殊的情况，在系统噪声分布关系未知的情况下。如不清楚状态和测量之间的

关系，也不清楚噪声的分布，只给一个条件概率密度函数，如 $\boldsymbol{P}[\boldsymbol{x}(k)\,|\,\boldsymbol{z}(1),\boldsymbol{z}(2),\cdots,\boldsymbol{z}(k)]$ 描述的状态 $\boldsymbol{x}(k)$ 和测量 $\boldsymbol{z}(1),\boldsymbol{z}(2),\cdots,\boldsymbol{z}(k)$ 关系。

同时也发现测量数据和状态的条件概率密度函数，可能不止一个峰值的。对于多峰值的概率密度函数，状态的均值并不真正代表实际状态的情况。由于不知道噪声的分布，就不能产生 sigma 点。对于这种情况，目前介绍的这些方法就都不能适用。那应该怎么办？研究者给出了粒子滤波器来处理这种特别复杂的情况。

在实际应用中到底选择哪一种滤波器呢？这就需要根据条件进行分析，如果情况比较清楚，就选择 EKF 或 UKF 滤波器。对于多峰值、非常复杂、情况不明的系统关系，只能使用粒子滤波器。关于粒子滤波器的知识，请读者参考相关书籍，本书不在此处详述，严格讲，粒子滤波器已经超过了 Kalman 滤波器的范围。

第 6 章　模型的离散化及目标机动轨迹仿真

作为第二部分“目标跟踪与系统仿真研究”的第一章，首先研究随机线性系统的数学描述以及离散化内容。什么是随机线性系统呢？指的是目标在运动和测量的过程含有噪声，为了把这些带有噪声的过程清楚地描述出来，必须要研究随机系统的数学描述。

那么离散化又是什么呢？有什么作用？已知跟踪系统主要研究的是系统的位移、速度和加速度这些物理量，这些物理量在实际问题当中都是连续的，可要对其实现跟踪，就要用计算机进行处理，所以要进行系统的离散化。模型离散化后是有变化的，但离散化模型必须要和实际真实的模型具有非常大的接近度，这样，跟踪算法结果与实际的目标运动才能具有很强的一致性，才可以跟踪得上，不至于“跟丢”。

通过第 4～5 章，知道 Kalman 滤波器要求系统的模型是完全准确的。但一个离散化的模型在描述连续系统的时候，已经发生了改变，从这种意义上讲，前面提到的最优估计方法在用计算机实现跟踪时是不能达到最优结果的。这时，只能做到两点：一是如何使线性化的系统和原来的连续系统具有尽量高的相似度；二是要研究在这种差别的情况下，给出的系统估计到底怎么样，给出一个定量的性能评价。

当跟踪一个运动目标时，要对其运动规律进行建模，也就是得到系统过程的模型。在承认所建数学模型和真正的目标运动有一定差距的前提下，通过系统仿真研究所提出的跟踪算法性能，这是研究的基本步骤。对于目标跟踪的研究来讲，仿真是非常重要的。

实际系统测量模型是由传感器决定的，选择什么样的测量传感器、测量模型是确定的。但在研究跟踪算法的性能时，往往会采用几种比较典型的传感器，如雷达、GPS、RFID 等传感器。本章将给出几种典型跟踪系统的测量模型，给出仿真研究时所需的数据和 MATLAB 程序。其实，这样的数据在很多资料上都有提及，并且很多研究者都在使用这些数据进行仿真，但是这些数据并没有公布，初学者在做仿真时需要自己重新采集，本书把这些仿真数据产生的过程用 MATLAB 程序重现。本章除了产生比较常用的蛇形机动，圆形机动等数据之外，还模拟了 GPS 数据和 RFID 测量数据，给出了 RFID 测量数据采集平台的软件著作权专利源代码。

本书的 Kalman 滤波器应用——目标跟踪与系统仿真研究部分共三章，内容安排如下：本章给出系统仿真需要的一些数据，并给出两种典型系统，GPS 和 RFID 系统的测量模型及数据。第 7 章给出几种目标跟踪系统的过程，第 8 章以 RFID 系统为例，研究具体跟踪算法的仿真过程。

6.1　随机线性系统的数学描述

随机线性系统一般指的是含有随机过程的线性系统，数学模型通常描述的是系统输入输出关系，对于机动目标运动模型来说，目标运动不受系统控制，所以一般不含有输入项，但目标运动含有用统计规律描述的噪声，需要在模型中有所体现。系统的输出为传感器观测到的测量值，如位移、温度等物理量。本节给出在目标跟踪中应用十分广泛的状态空间模型，由于目标运动的位移、速度及加速度等物理量是连续的，本节先从描述目标运动的连续系统动力学关系入手，给出系统连续状态方程模型，进而讨论运用线性系统求解的方法对连续系统进行离散化，给出两种目标跟踪的离散化模型，并指出这两种模型的根本区别。

值得一提的是，从系统的角度来讲，并不是所有的离散系统都与目标跟踪系统相同，是从连续系统离散化而来的。有很多离散系统，如股票数据预测系统，研究的是直接采样后得到的离散数据，像这样的系统或者由于原连续系统太过复杂，或者由于人们的研究空间只限于离散空间即可，人们往往不再关注原来连续系统的模型如何，而是直接研究其离散模型。一般的研究方法是先得到其自回归模型，再利用自回归模型得到状态空间模型。但对于目标跟踪系统，位移、速度及加速度等物理量的关系式是客观存在且十分直观的，连续系统模型是离散系统模型的基础，研究系统变量的连续变化可以从根本上剖析离散模型的机理，因此，清楚了解连续时间系统模型是十分必要的。

系统的状态方程及测量方程描述为

$$\begin{cases} \dot{\boldsymbol{x}}(t) = \boldsymbol{F}(t)\boldsymbol{x}(t) + \boldsymbol{G}(t)\boldsymbol{w}_c(t) \\ \boldsymbol{z}(t) = \boldsymbol{H}(t)\boldsymbol{x}(t) + \boldsymbol{v}(t) \end{cases} \tag{6-1}$$

式中，$\boldsymbol{x}(t)$ 是系统的 n 维状态向量，$\boldsymbol{z}(t)$ 是 m 维测量输出向量，$\boldsymbol{w}_c(t)$ 是 p 维随机过程噪声，$\boldsymbol{v}(t)$ 是 m 维随机测量噪声。如果 $\boldsymbol{F}(t)$、$\boldsymbol{G}(t)$、$\boldsymbol{H}(t)$ 的各元素均与时间无关，且 $\boldsymbol{w}_c(t)$、$\boldsymbol{v}(t)$ 都是平稳随机过程，则系统方程可简化为

$$\dot{\boldsymbol{x}}(t) = \boldsymbol{F}\boldsymbol{x}(t) + \boldsymbol{G}\boldsymbol{w}_c(t) \tag{6-2}$$

$$\boldsymbol{z}(t) = \boldsymbol{H}\boldsymbol{x}(t) + \boldsymbol{v}(t) \tag{6-3}$$

一般假设 $\boldsymbol{w}_c(t)$、$\boldsymbol{v}(t)$ 均为零均值高斯白噪声过程，其中 $E[\boldsymbol{w}_c(t)] = 0$，$E[\boldsymbol{w}_c(t)\boldsymbol{w}_c^{\mathrm{T}}(s)] = \boldsymbol{\sigma}^2\delta(t-s)$，$E[\boldsymbol{v}(t)] = 0$，$E[\boldsymbol{v}(t)\boldsymbol{v}^{\mathrm{T}}(s)] = \boldsymbol{R}\delta(t-s)$，$\delta(t-s)$ 为 Dirac 函数，T 表示矩阵转置。

下面，来求解式（6-2）状态方程描述的状态 $\boldsymbol{x}(t)$，这将是连续系统离散化的基础。式（6-2）方程为微分方程，由高等数学中求解微分方程的知识中可知，其解与指数函

数 $\mathrm{e}^{\boldsymbol{F}t}$ 有密切联系，这里，设 $\mathrm{e}^{\boldsymbol{F}t}=\boldsymbol{I}+\boldsymbol{F}t+\dfrac{\boldsymbol{F}^2t^2}{2!}+\dfrac{\boldsymbol{F}^3t^3}{3!}+\dfrac{\boldsymbol{F}^4t^4}{4!}+\cdots$。若要对状态方程（6-2）进行求解，需要利用 $\mathrm{e}^{\boldsymbol{F}t}$ 的以下两个重要性质：

$$\begin{cases}\dfrac{\mathrm{d}}{\mathrm{d}t}\mathrm{e}^{\boldsymbol{F}t}=\boldsymbol{F}+\boldsymbol{F}^2t+\dfrac{\boldsymbol{F}^3t^2}{2!}+\dfrac{\boldsymbol{F}^4t^3}{3!}+\dfrac{\boldsymbol{F}^5t^4}{4!}+\cdots=\boldsymbol{F}\cdot\mathrm{e}^{\boldsymbol{F}t}\\ \dfrac{\mathrm{d}}{\mathrm{d}t}[\mathrm{e}^{-\boldsymbol{F}t}\boldsymbol{x}(t)]=\mathrm{e}^{-\boldsymbol{F}t}[\dot{\boldsymbol{x}}(t)-\boldsymbol{F}\boldsymbol{x}(t)]\end{cases}\tag{6-4}$$

求解状态方程（6-2）描述状态 $\boldsymbol{x}(t)$ 的过程如下：对式（6-2）两边同时左乘 $\mathrm{e}^{-\boldsymbol{F}t}$，并整理得

$$\mathrm{e}^{-\boldsymbol{F}t}[\dot{\boldsymbol{x}}(t)-\boldsymbol{F}\boldsymbol{x}(t)]=\mathrm{e}^{-\boldsymbol{F}t}\boldsymbol{G}\boldsymbol{w}_c(t)\tag{6-5}$$

根据指数函数 $\mathrm{e}^{\boldsymbol{F}t}$ 如式（6-4）所示的性质可知，式（6-5）左边为 $\dfrac{\mathrm{d}}{\mathrm{d}t}[\mathrm{e}^{-\boldsymbol{F}t}\boldsymbol{x}(t)]$，即

$$\frac{\mathrm{d}}{\mathrm{d}t}[\mathrm{e}^{-\boldsymbol{F}t}\boldsymbol{x}(t)]=\mathrm{e}^{-\boldsymbol{F}t}\boldsymbol{G}\boldsymbol{w}_c(t)\tag{6-6}$$

对式（6-6）两边同时取积分，积分区间取为 $[t_0,t]$，即可得到

$$\mathrm{e}^{-\boldsymbol{F}t}\boldsymbol{x}(t)=\mathrm{e}^{-\boldsymbol{F}t_0}\boldsymbol{x}(t_0)+\int_{t_0}^{t}\mathrm{e}^{-\boldsymbol{F}\lambda}\boldsymbol{G}\boldsymbol{w}_c(\lambda)\mathrm{d}\lambda$$

然后，将上式两边同时左乘 $\mathrm{e}^{\boldsymbol{F}t}$，得

$$\boldsymbol{x}(t)=\mathrm{e}^{\boldsymbol{F}(t-t_0)}\boldsymbol{x}(t_0)+\int_{t_0}^{t}\mathrm{e}^{\boldsymbol{F}(t-\lambda)}\boldsymbol{G}\boldsymbol{w}_c(\lambda)\mathrm{d}\lambda\tag{6-7}$$

现在将状态完全解进行分段研究，设系统以 T_0 的时间间隔进行采样，即系统的采样周期为 T_0。取 $t_0=kT_0$ 及 $t=kT_0+T_0$ 代入式（6-7）得到

$$\boldsymbol{x}(kT_0+T_0)=\mathrm{e}^{\boldsymbol{F}T_0}\boldsymbol{x}(kT_0)+\int_{kT_0}^{kT_0+T_0}\mathrm{e}^{\boldsymbol{F}(kT_0+T_0-\lambda)}\boldsymbol{G}\boldsymbol{w}_c(\lambda)\mathrm{d}\lambda$$

将上式中 T_0 省略，只用 k 或 k+1 表示离散时刻取值，得到

$$\boldsymbol{x}(k+1)=\mathrm{e}^{\boldsymbol{F}T_0}x(k)+\int_{0}^{T_0}\mathrm{e}^{\boldsymbol{F}(T_0-\lambda)}\boldsymbol{G}\boldsymbol{w}_c(\lambda)\mathrm{d}\lambda\tag{6-8}$$

式（6-8）中右端第二项 $\int_0^{T_0}\mathrm{e}^{\boldsymbol{F}(T_0-\lambda)}\boldsymbol{G}\boldsymbol{w}_c(\lambda)\mathrm{d}\lambda$ 体现了噪声对系统状态的影响，它是一个积分项，如果设

$$\boldsymbol{w}(k)=\int_{kT_0}^{kT_0+T_0}\mathrm{e}^{\boldsymbol{F}(T_0-\lambda)}\boldsymbol{G}\boldsymbol{w}_c(\lambda)\mathrm{d}\lambda\tag{6-9}$$

设

$$\boldsymbol{A}=\mathrm{e}^{\boldsymbol{F}T_0}\tag{6-10}$$

根据式（6-8）可得

$$\boldsymbol{x}(k+1)=\boldsymbol{A}\boldsymbol{x}(k)+\boldsymbol{w}(k) \tag{6-11}$$

式（6-11）为系统的离散系统方程，此时系统过程噪声的统计描述为

$$E[\boldsymbol{w}(k)]=0$$

$$\boldsymbol{Q}(k)=E[\boldsymbol{w}(k)\boldsymbol{w}^{\mathrm{T}}(u)]=\int_0^{T_0}\mathrm{e}^{\boldsymbol{F}(T_0-\lambda)}\boldsymbol{G}\times\boldsymbol{\sigma}^2\times\boldsymbol{G}^{\mathrm{T}}[\mathrm{e}^{\boldsymbol{F}(T_0-\lambda)}]^{\mathrm{T}}\mathrm{d}\lambda\cdot\delta(k-u) \tag{6-12}$$

式中，$\delta(k-u)$ 与前面所提到的 $\delta(t-s)$ 不同，此时的 k 和 u 的取值为离散取值点，而 t 和 s 为连续时间中的任意两点。

下面研究式（6-9）的化简形式。如果 $\boldsymbol{w}_c(t)$ 在一个采样周期内可以视为时常噪声，则 $\boldsymbol{w}(k)$ 中的噪声部分可以从积分项中提出，即 $\int_0^{T_0}\mathrm{e}^{\boldsymbol{F}(T_0-\lambda)}\boldsymbol{G}\mathrm{d}\lambda\cdot\boldsymbol{w}_c(kT_0)$，为与式（6-9）中 $\boldsymbol{w}(k)$ 相区别，写为 $\boldsymbol{w}_s(k)=\int_0^{T_0}\mathrm{e}^{\boldsymbol{F}(T_0-\lambda)}\boldsymbol{G}\mathrm{d}\lambda\boldsymbol{w}_c(k)$。又设 $\boldsymbol{B}=\int_0^{T_0}\mathrm{e}^{\boldsymbol{F}(T_0-\lambda)}\boldsymbol{G}\mathrm{d}\lambda$，此时系统的离散状态方程为

$$\boldsymbol{x}(k+1)=\boldsymbol{A}\boldsymbol{x}(k)+\boldsymbol{w}_s(k) \tag{6-13}$$

同时系统的过程噪声统计描述不再与式（6-9）相同，为

$$E[\boldsymbol{w}_s(k)]=0,\quad \boldsymbol{Q}_s(k)=E[\boldsymbol{w}_s(k)\boldsymbol{w}_s^{\mathrm{T}}(u)]=E[\boldsymbol{B}\boldsymbol{w}_c(k)\boldsymbol{w}_c^{\mathrm{T}}(u)\boldsymbol{B}^{\mathrm{T}}]=\boldsymbol{B}\boldsymbol{\sigma}^2\boldsymbol{B}^{\mathrm{T}}\delta(k-u) \tag{6-14}$$

比较式（6-12）和式（6-14）可以发现，离散化后的过程噪声协方差矩阵不同，式（6-12）表明，离散化后的协方差是实际连续过程协方差的积分，计算的是过程噪声累积效果；而式（6-14）的前提假设连续时间噪声为分段时常噪声，使用增益矩阵 $\boldsymbol{B}$ 来体现分段噪声的累积效果。为区别上述两种过程方程离散化的不同，有文献将前者称为基于连续系统的离散化模型（Discrete Model Based on Continuous System，DCM），而将后者称为直接离散化模型（Direct Discrete Model，DDM）。

对系统模型（6-1）中的测量方程取 $t=kT_0+T_0$，并省略 T_0 得到

$$\boldsymbol{z}(k+1)=\boldsymbol{H}\boldsymbol{x}(k+1)+\boldsymbol{v}(k+1) \tag{6-15}$$

观察式（6-15）可以发现，式（6-15）描述的是 $\boldsymbol{y}(t)$ 、 $\boldsymbol{x}(t)$ 及 $\boldsymbol{v}(t)$ 同时刻的关系，因此通常情况下用 k 时刻的测量方程 $\boldsymbol{z}(k)=\boldsymbol{H}x(k)+\boldsymbol{v}(k)$ 描述即可。结合状态方程及测量方程，可以得到以下两种离散系统状态方程形式。

基于连续系统的离散化模型（DCM）：

$$\begin{cases}\boldsymbol{x}(k+1)=\boldsymbol{A}\boldsymbol{x}(k)+\boldsymbol{w}(k)\\ \boldsymbol{z}(k)=\boldsymbol{H}\boldsymbol{x}(k)+\boldsymbol{v}(k)\end{cases} \tag{6-16}$$

直接离散化模型（DDM）：

$$\begin{cases} \boldsymbol{x}(k+1)=\boldsymbol{A}\boldsymbol{x}(k)+\boldsymbol{B}\boldsymbol{w}_c(k) \\ \boldsymbol{z}(k)=\boldsymbol{H}\boldsymbol{x}(k)+\boldsymbol{v}(k) \end{cases} \tag{6-17}$$

两种模型的区别在于过程噪声的协方差矩阵，如式（6-12）、式（6-14）所示。

6.2　几类跟踪系统中常用曲线的模拟

本节给出几种常用的跟踪目标轨迹数据，包括目标做匀速直线运动、目标做蛇形机动、圆周运动、先做匀速直线运动再做圆周运动，以及在二维平面内做任意方向机动的模拟轨迹数据。本书不仅给出目标做各种运动的测量数据，还给出相应的程序，供读者使用。有的程序调用了函数，有的程序没有，并没有统一的格式，给出这些程序的另一个目的在于让读者熟悉这些 MATLAB 程序，方便读者编制自己的测量数据模拟程序。

下面函数 funtrackingline 能够模拟目标做匀速直线运动数据，其中输入变量 a 表示匀速直线运动的斜率，t 是一个向量，该向量通过从运动的开始时间到终止时间以采样周期为间隔对时间序列进行采样得到，R 是测量噪声，输出变量包括两个：一个是 yreal，它表示目标的真实运动，不含有传感器的测量噪声；另一个是 ym，加入了传感器的测量噪声，是实际系统的测量数据。

```
function [yreal,ym]=funtrackingline(a,t,R)
yreal=a*t;
ym=yreal+randn(size(yreal))*sqrt(R);
```

例 6.1　调用函数 funtrackingline，在程序中先设置输入参数，比如斜率 a 为 3，t 是从 0 到 10、采样间隔 0.1 的采样时刻向量，测量噪声的方差 R 为 30，然后把结果画出来，如图 6-1 所示。图中横坐标是采样时刻，纵坐标是测量数据。图中画出两条曲线：一条是目标运动的实际数据；另一条是带有传感器测量噪声的测量数据。具体程序如下：

```
clc
clear
a=3;t=0:0.1:10;R=30;
[yreal,ym]=funtrackingline(a,t,R);
plot(t,yreal,t,ym)
xlabel('采样时刻')
ylabel('测量数据')
legend('目标运动的实际数据,'传感器测量数据')
```

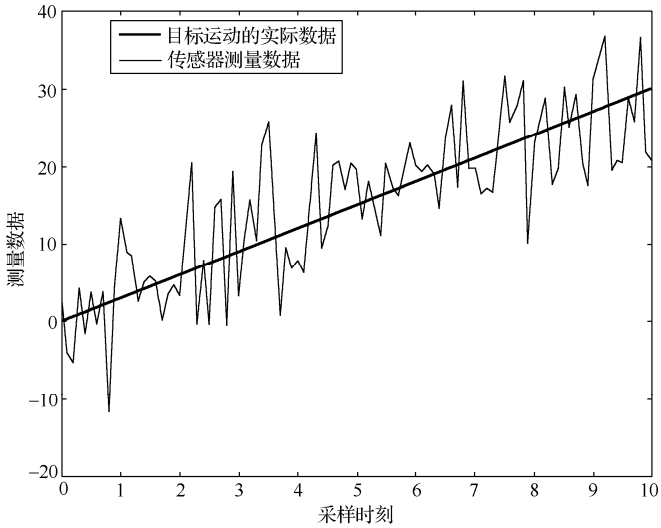


图 6-1　匀速直线运动模拟数据

例 6.2　蛇形机动的正弦曲线模拟。

给出函数 funtrackingsnake，里面包括四个输入变量，其中 a 是正弦曲线的振幅，omig 指正弦曲线的角加速度，和前面一样，t 表示从开始到结束由每个采样点组成的采样时刻向量，R 是测量噪声的方差。同样，输出变量有两个：一个是不带测量噪声的 yreal；另一个是带有测量噪声的 ym。

```
function [yreal,ym]=funtrackingsnake(a,omig,t,R)
yreal=a*sin(omig*t);
ym=yreal+randn(size(yreal))*sqrt(R);
```

调用函数 funtrackingsnake，设置相应的参数再进行适当标注，程序如下：

```
a=2;omig=pi/4;R=3; t=0:0.1:10;
[yreal,ym]=funtrackingsnake(a,omig,t,R);
plot(t,yreal,t,ym)
xlabel('采样时刻')
ylabel('测量数据')
legend('目标运动的实际数据,'传感器测量数据')
```

程序结果如图 6-2 所示。

例 6.3　圆周运动模拟。

下面这段程序没有使用函数，直接给出了产生圆周运动的程序，其中各个量的说明已经在程序中%后面做了标注。程序中直接设置了几个参量，包括起点（2000, 0），线速度 300，半径 2000，圆心（0, 0），读者可以根据自己的实际需要来重新定义各个变量的具体数值。ww 是角度，程序中用正弦、余弦分别计算横、纵坐标的数值，利用 plot(x(1,:),x(2,:),'*')画出二维图，第二张图分别给出横、纵坐标的量，把它们画在一张图上。

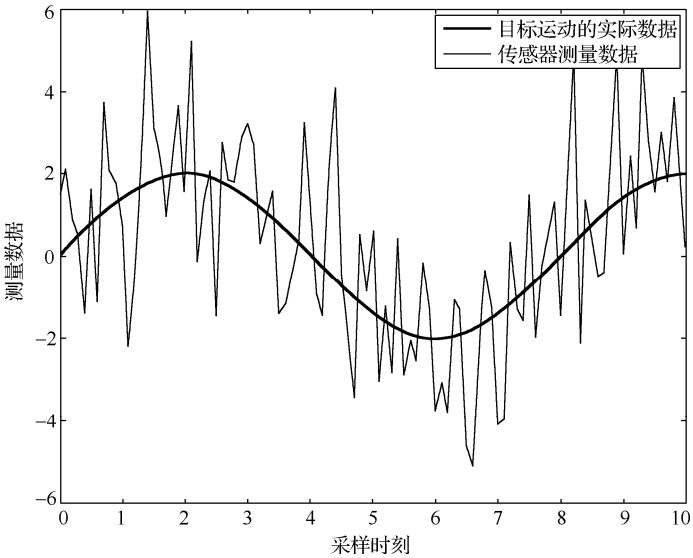


图 6-2　蛇形机动的正弦曲线模拟数据

程序具体如下：

```
clc
clear
x0=[2000,0];
v=300;%线速度
T=1;
r=2000;%半径
w=v/r;
op=[0 0];%圆心
x=[];xt=[];www=[];ww=0;
for t=1:T:50
ww=ww+w;
    x2=[cos(ww)*r+op(1);sin(ww)*r+op(2)];
    x=[x x2];
    xt=[xt t];
    www=[www ww];
end
plot(x(1,:),x(2,:),'*')
figure
plot(xt,x,'*')
```

程序结果如图 6-3～图 6-4 所示。

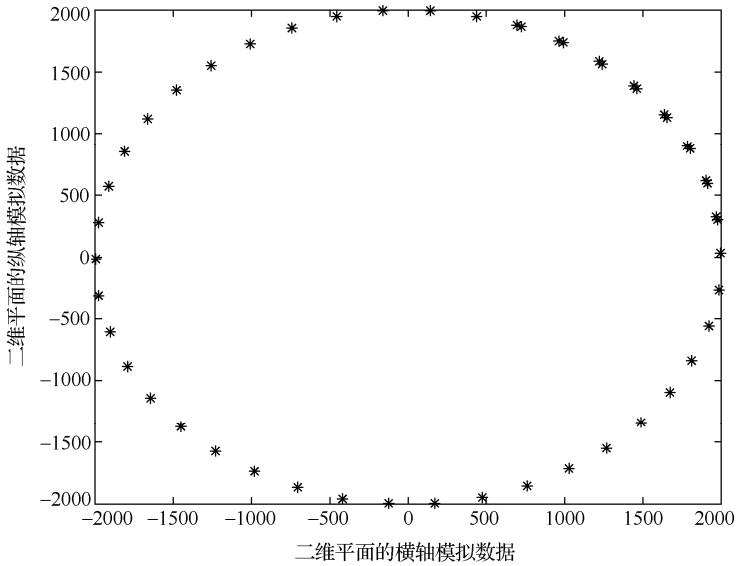


图 6-3　目标在二维平面内圆周运动模拟数据

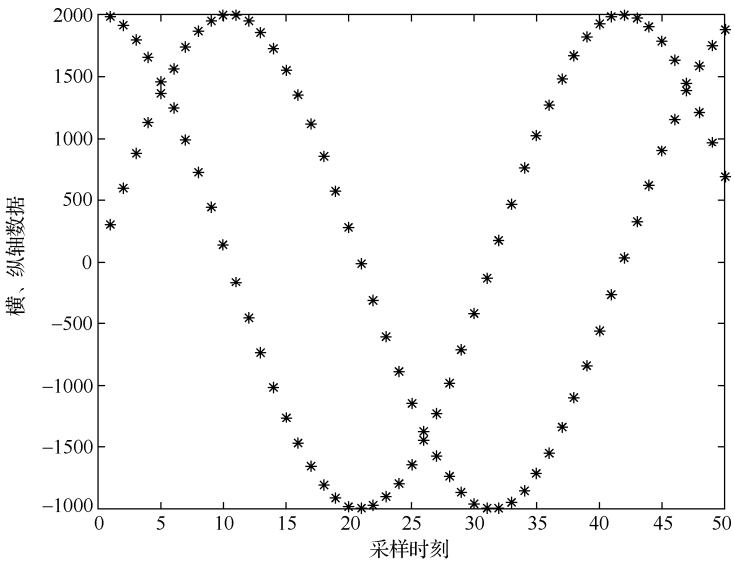


图 6-4　目标做圆周运动时的横、纵轴数据

例 6.4　匀速直线运动+圆周运动目标的真实轨迹模拟。

```
function [xt,y]=funtrackinglinecircle(T,Tt,Tz,R)
%T:采样周期
%Tt:直线运行时间
%Tz：圆周运行时间
```

```
%R：测量噪声方差

%本程序模拟匀速直线运动+圆周运动目标的真实轨迹。
%功能描述如下。
% 针对某机动目标运动的特点，假设某运动目标经历了两个航行阶段：
%初始匀速直线运动、匀速圆周转弯机动运动。将目标建立在二维坐标系中，初始位置为 XXX，目标线速度为 XXX;
% 以 XX 角速度向斜上方运动，目标运行 Tt 秒后，作向心 wv ms/s^2 的匀加速圆周运动，速度的大小 v 保持不变。
x0=[0 -2000]';%初始位置
v=200;%速度
vj=pi/4;%与水平方向所成角度

x=[];xt=[];
for t=1:T:Tt
    x2=[sin(vj)*v+x1(1);cos(vj)*v+x1(2)];
    x=[x x2];
    xt=[xt t];
    x1=x2;
end

% 圆周运动
x0=x2;
v=200;%线速度
r=2000;%半径
w=v/r;
op=[x0(1)-r x0(2)];%圆心
x1=x0;www=[];ww=0;
for tt=t:Tz+t
ww=ww+w
    x2=[cos(ww)*r+op(1);sin(ww)*r+op(2)];
    x=[x x2];
    xt=[xt tt];
    x1=x2;
      www=[www ww];
end
y=x+randn(size(x))*sqrt(R);
```

例 6.4 中 funtrackinglinecircle 函数的输入量有四个：采样周期 T、直线运动时间 Tt、圆周运动时间 Tz 和测量噪声的方差 R。从函数中可看出，有些变量是已经设置好的，如初始位置（0, –2000），还有线速度以及作直线运动时与水平方向产生的角度。程序先进行一段直线运动，运行到最后的点 x2。结束 for 循环之后的下一个起点就是

x2，设置了线速度以及运动的半径，给出圆心，又用 for 循环里的正、余弦进行圆周模拟运动。最后，对给出的目标真实运动轨迹 x 加上测量噪声，噪声方差是 R。

函数有两个输出：一个是 xt，另一个是 y。xt 是运动的时间，y 是加上测量噪声之后的传感器测量的输出。下面的程序调用上面的函数，程序调用的时候要设置四个输入量，如采样周期为 1s，直线运动时间为 20s，然后做圆周运动 50s，测量噪声方差为 1000，最终的结果如图 6-5～图 6-6 所示。图 6-5 模拟的是目标在平面上运动的情形，可以看出目标从起点开始先做直线运动，然后向左上方转弯，走了一个圈。因为数据带有一定的测量噪声，所以图形看起来不太规整。图 6-6 给出的测量数据是目标的真实轨迹和测量噪声的叠加，进行目标跟踪仿真研究的时候使用图 6-6 的数据。

```
T=1;Tt=20;Tz=50;R=1000;
 [xt,y]=funtrackinglinecircle(T,Tt,Tz,R);
 plot(y(1,:),y(2,:))
 xlabel('横轴模拟数据')
 ylabel('纵轴模拟数据')
 figure
 subplot(2,1,1),plot(xt,y(1,:))
 xlabel('采样时刻');ylabel('横轴模拟数据')
 subplot(2,1,2),plot(xt,y(2,:))
 xlabel('采样时刻');ylabel('纵轴模拟数据')
```

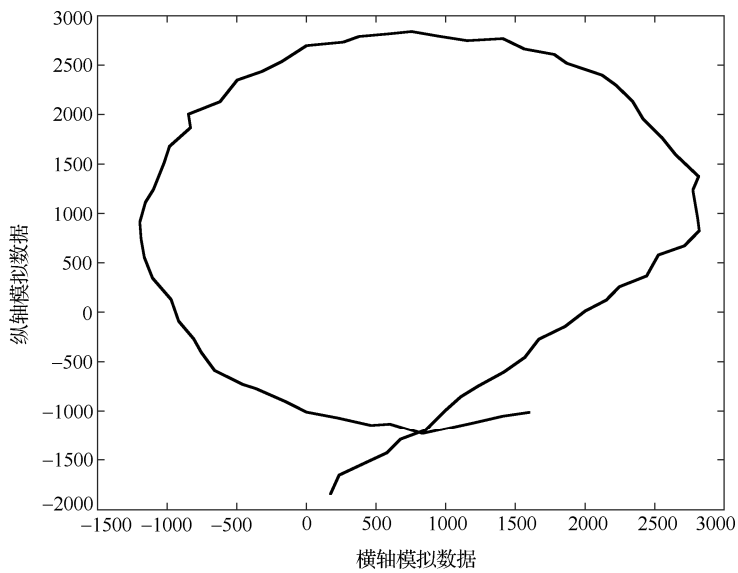


图 6-5　目标先做直线运动、再做圆周运动时的二维平面运动

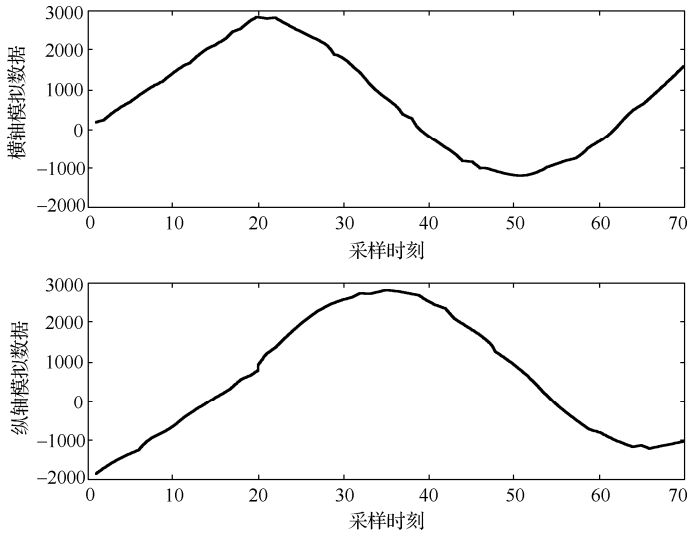


图 6-6　目标先做直线运动，再做圆周运动时的横、纵轴数据

6.3　GPS 跟踪系统的机动目标轨迹模拟

首先介绍一下 GPS 系统的数据采集特征，即传感器测量特征。GPS 系统可以测量二维空间，并且该测量与横、纵坐标不相关。一般情况下，横、纵坐标都含有噪声，方差已知，因此模拟的测量数据需要包含横、纵坐标，是 2 维数据。

例 6.5　下面给出在 2 维平面内目标做任意曲线运动的测量数据模拟程序。这段程序使用了函数 axis，这个函数可以设置坐标轴，即横、纵坐标的起点和终点，axis([0 100 0 100])表示横、纵坐标的起点和终点都是从 0 到 100，前两个数值表示横坐标的起点和终点，后两个数值表示纵坐标的起点和终点。hold on 函数的功能是保持目前图形窗口的状态，即接下来的图会画在目前的图形窗口上。

程序中还使用了两个 disp 函数，功能是给出说明，disp( )里面的任何字符都会显示在窗口上。这里显示的内容是本程序的使用说明：使用鼠标左键点出每一个点，点击右键表示点击该点后结束数据产生步骤。为什么这样提示呢？因为这段程序可以使用坐标在横、纵坐标轴从 0～100 的任意一点来画点，并记住每一个点的前后顺序，还能把这些点连成一条曲线。那么怎么实现记录每一个点呢？用 ginput 这个函数，用 for 循环来记录所有用左键点击的点。ginput 函数输入 1 表示点左键，输出是点击点的横、纵坐标值。每点击一个点，在程序中都会用圆圈画出来，不过注意在点击鼠标左键点时不要太快，否则会丢点。接下来把所有点用函数 spline 连成一条光滑曲线，使用 spline 函数需要设置插入的点数，程序中在每两个点之间插入 10 个点，这样能形成一条光滑曲线，来模拟在平面上机动目标走过的曲线。

最后使用 subplot，把横、纵坐标放在一张图上，其中实线是机动目标走过的真实的轨迹，不带测量噪声。点模拟的是带有测量噪声的传感器测量输出。

```
%使用说明：使用鼠标左键可以产生轨迹的各个点，鼠标右键为结束点。
axis([0 100 0 100])
hold on
% 程序变量初始化。
xy = [];
n = 0;
% 使用循环，得到鼠标点击左键是的坐标位置。
disp('Left mouse button picks points.')      %在主窗口上提示操作方法
disp('Right mouse button picks last point.')
but = 1;
while but == 1
    [xi,yi,but] = ginput(1);            %该函数可以得到鼠标左键的坐标点
    plot(xi,yi,'ro')
    n = n+1;
    xy(:,n) = [xi;yi];
end
% 利用插值函数获得光滑曲线，模拟目标运动曲线和测量数据。
t = 1:n;
ts = 1: 0.1: n;
xys = spline(t,xy,ts);
plot(xys(1,:),xys(2,:),'b-'); %用直线画出目标运动曲线
Rx=10;Ry=10;    %设置横轴、纵轴的测量噪声方差 Rx、Ry
plot(xys(1,:)+randn(size(xys(1,:)))*sqrt(Rx),xys(2,:)+randn(size(xys
(1,:)))*sqrt(Ry),'k.');   %在 2 维坐标下用黑色点画出每一个测量数据
xlabel('横轴模拟数据');ylabel('纵轴模拟数据')
hold off
%在另一张图上分别画出横、纵轴的目标运动数据和测量数据。
figure
subplot(2,1,1),plot(xys(1,:))
hold on
subplot(2,1,1),plot(xys(1,:)+randn(size(xys(1,:)))*sqrt(Rx),'.')
ylabel('横轴模拟数据');

subplot(2,1,2),plot(xys(2,:))
hold on
subplot(2,1,2),plot(xys(2,:)+randn(size(xys(1,:)))*sqrt(Rx),'.')
ylabel('纵轴模拟数据')
hold off
%存储数据。
save  mytarget1 xys ts
```

程序运行结果如图 6-7～图 6-8 所示。

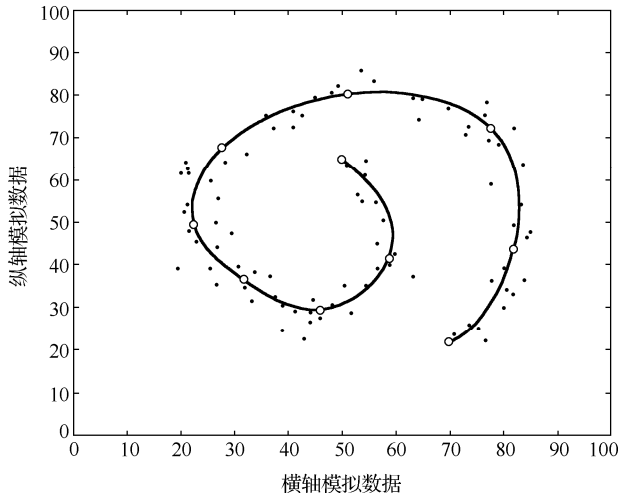


图 6-7　GPS 系统的机动目标轨迹模拟

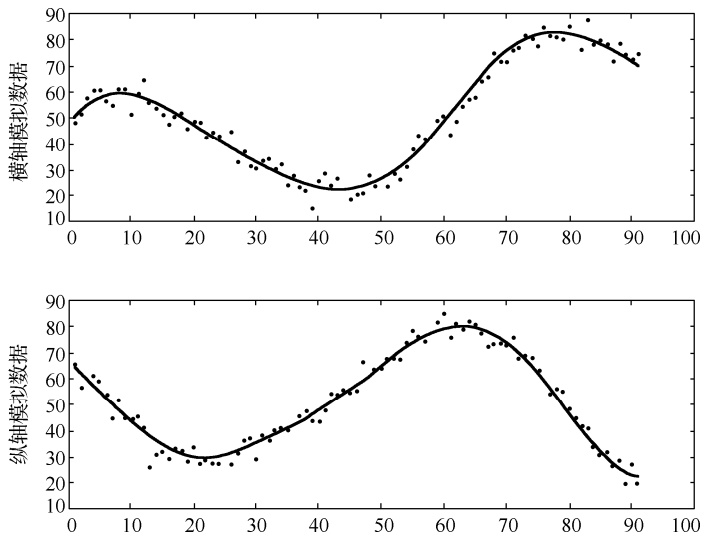


图 6-8　GPS 系统的机动目标轨迹横、纵坐标轴数据模拟

6.4　RFID 跟踪系统的机动目标轨迹模拟

下面来模拟实现 RFID 实现机动目标跟踪的系统。

6.4.1　RFID 系统测量模型

首先介绍一下 RFID 系统的测量特点。其测量需要两部分：一个是 RFID 阅读器；另一个是射频标签。阅读器和射频标签是通过射频来相互感应的。如果射频标签到达阅读器能够测量的范围，阅读器会给出两个信息：一个是射频标签内的标签码；另一个是标签到阅读器之间的距离。有一些低价位的 RFID 阅读器只能给出某标签到达其范围之内，并不能给出具体的距离，但是一些高档的 RFID 阅读器是可以给出一个具体的距离数值的。RFID 系统的测量模型也包含上述两部分，阅读器的位置是已知的，测量模型的输出是目标运动到阅读器的距离值。

设 $d_n(t_i)$ 是在 t_i 时刻目标到第 n 个阅读器的距离，则 $d_n(t_i)$ 可以表示为

$$d_n(t_i)=\sqrt{[x(t_i)-x_n(0)]^2+[y(t_i)-y_n(0)]^2} \tag{6-18}$$

式中，$x_n(0)$ 和 $y_n(0)$ 表示第 n 个 RFID 阅读器在 2 维空间横、纵坐标的位置，$x(t_i)$ 和 $y(t_i)$ 是目标在 2 维空间横、纵坐标的位置，$d_n(t_i)$ 的实际值其实是不知道的，而只能得到叠加噪声的测量输出数据：

$$z_n(t_i)=d_n(t_i)+v_n(t_i) \tag{6-19}$$

式中，$v_n(t_i)$ 是在 t_i 时刻第 n 个 RFID 阅读器的测量噪声，满足 $v_n(t_i)/d_n(t_i)\sim N\left(0,\left(\dfrac{0.2303\sigma_p}{\gamma}\right)^2\right)$，$\sigma_p$ 是标准偏差，γ 是系统误差参数，$d_n(t_i)$ 是在 t_i 时刻目标到第 n 个阅读器的距离。RFID 系统的测量误差随着目标到每一个阅读器距离的增加而改变，距离越大测量误差越大，距离与误差的比值是 $\dfrac{0.2303\sigma_p}{\gamma}$。实验表明，系统参数 γ 一般应取为 1.6～6.5。

6.4.2　RFID 室内跟踪系统仿真数据平台软件

根据 RFID 系统的测量模型的特点，编制了仿真数据模拟平台软件，该软件的功能包括两个部分：第一部分用于生成 RFID 室内跟踪系统仿真数据，即在坐标系中产生基于可设定测量半径和标注的跟踪系统仿真模型。第二部分用于完成仿真数据的显示、存储并以图像的形式对数据进行保存等功能。下面详细说明该平台软件的构成与实现。

系统设计的总体方案如图 6-9 所示。

图形对象包括 text、uicontrol 对象以及图形、坐标轴及其子对象，其对象层次结构如图 6-10 所示。

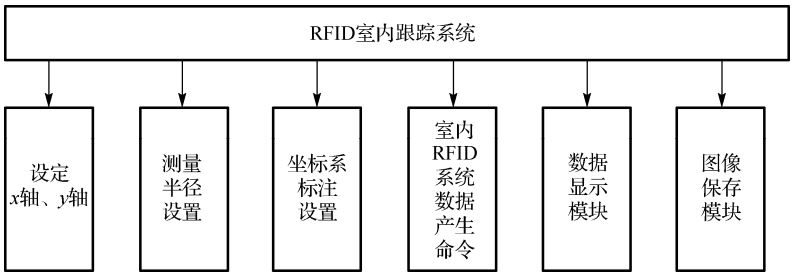


图 6-9　系统总体设计方案

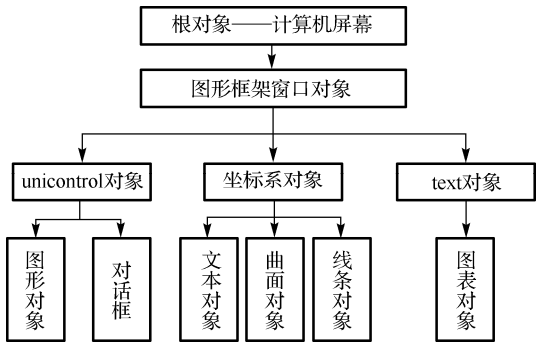


图 6-10　对象层次结构

软件系统使用的变量定义如下。

hFigure //创建的初始软件界面

hAxes //RFID 室内跟踪系统仿真坐标系

axes_1//坐标系标注

R_Width //测量半径

answer //设置半径对话框

answer1//设置标注对话框

d //真实轨迹数据

dmm //阅读器与可测量实际估计间的距离

readerlocation // RFID Reader 的位置

readermc //设定测量范围

f //数据显示窗口

sel //提问对话框

软件系统包括以下自定义函数。

RFID 室内跟踪系统仿真函数“开始”reader ( )

确定测量精度的“设置半径”函数 range ( )

介绍软件功能的“使用说明”函数 instructions ( )

保存参考图像 savepic ( )
显示数据函数 get_data_xls ( )
更改标注函数 change_tittle ( )
关闭软件函数 close( )

1. 系统函数介绍

图形用户界面的启动方法有两种：一种是点选 MATLAB 工具中“GUIDE”；另外一种是采用命令方式，在 MATLAB 的命令窗口中键入“GUIDE”（或者“guide”，不区分大小写）。本系统采用第二种启动方式，在键入主程序 RFID_GUI_1 并回车后，在屏幕正中央弹出一个窗口如图 6-11 所示。

本系统由一系列子函数组成：主函数、reader 函数、rang 函数和 savePic 回调函数等，部分函数的实现与功能介绍如下。

1）instruction 函数

软件为方便初次使用者的操作，特意设置了使用说明向导，通过 text 函数实现“说明书”的功能，当用户单击该控件时，其运行结果如图 6-12 所示。

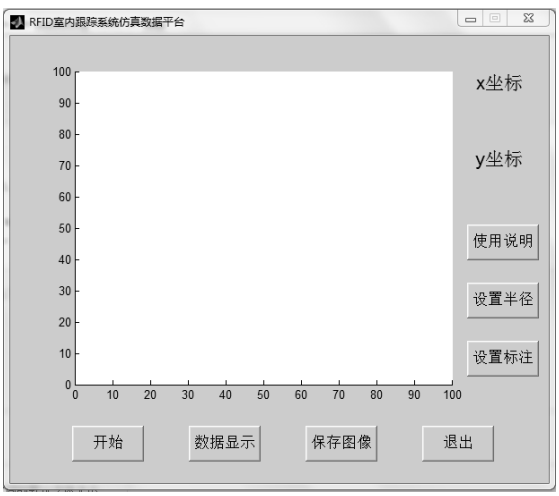


图 6-11　RFID 室内跟踪系统仿真数据初始软件

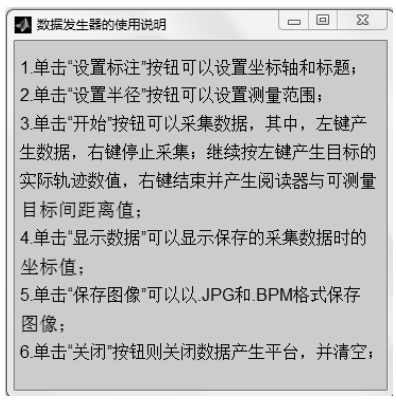


图 6-12　数据平台的使用说明

2）change_tittle 函数

该函数的功能是，用户可以在弹出的对话框中输入任意自定义的标注字符更改主题。该函数调用格式为

```
answer=inputdlg(prompt,name,numlines,defaultanswer);
```

其中，answer 为字符串单元数组，用于储存用户输入的字符串，prompt 为提示字符串，

name 为对话框的标题，numlines 为用户输入的最大数，defaultanswer 为默认的输入字符串，维数与 prompt 相同。实现标注设置对话框的代码如下：

```
prompt={'设置标题名称：','设置横坐标名称：','设置纵坐标名称：'};
                        name='设置标注';
                        numlines=1;
                defaultanswer={'RFID','x','y'};
```

运行结果如图 6-13 所示。

图 6-13　输入对话框

3）rang 函数

与 change_tittle 函数类似，设置半径的功能也是采用对话框的形式实现，其函数调用格式为

```
prompt='设置测量半径：';
name='半径设置';
answer=inputdlg(prompt,name);
```

这里需要注意的是 answer 里存储的只能是字符串或者字符数，想要得到数字，需添加代码：

```
If ～isempty(answer)
R_Width = floor(str2double(answer));
end
```

在上面的代码中，首先判断 answer 里存放的字符串或字符数是否为空，不为空则将 answer 里第一个位置的字符串或字符数转换为数字形式，然后赋值给 R_Width，获取用户输入的半径值，并取整。运行结果如图 6-14 所示。

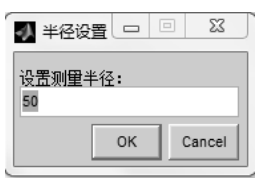


图 6-14　半径设置对话框

4）reader 函数

本模块通过 ginput 函数实现数据采集，并将 GUI 界面与 m 函数交互，仿真出 RFID 室内跟踪系统。其在 RFID 阅读器的测量默认半径为 50 时的运行结果如图 6-15 所示。

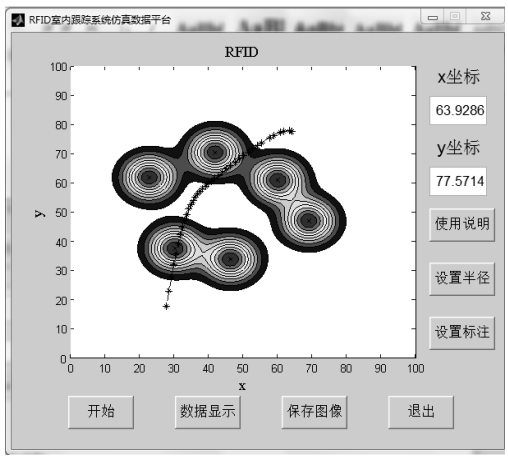


图 6-15　RFID 室内跟踪系统仿真数据软件

图中，圆圈的中心位置模拟 RFID 阅读器在室内的位置，其测量半径用圆圈表示，每个阅读器的监视区域，从中心向外逐渐减弱，星形线为 RFID 室内跟踪系统产生的目标参考轨迹。

本模块在运行中可以实时显示 RFID 阅读器及目标运动参考轨迹的坐标，并以表格的形式存储在 excel 文件中，方便后续调用数据。

当 RFID 阅读器测量半径分别为 10、150 时，所得的仿真情况分别如图 6-16～图 6-17 所示。

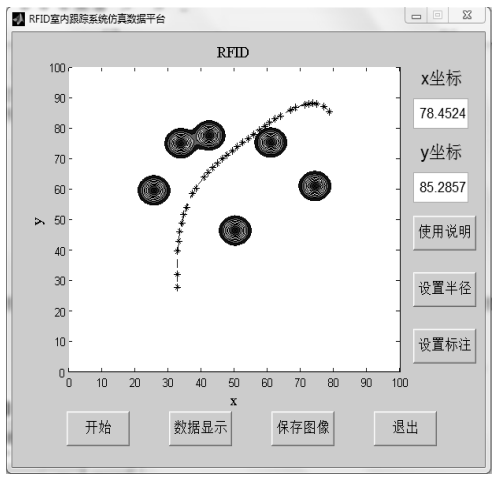


图 6-16　RFID 阅读器测量半径为 10 时平台的输出界面

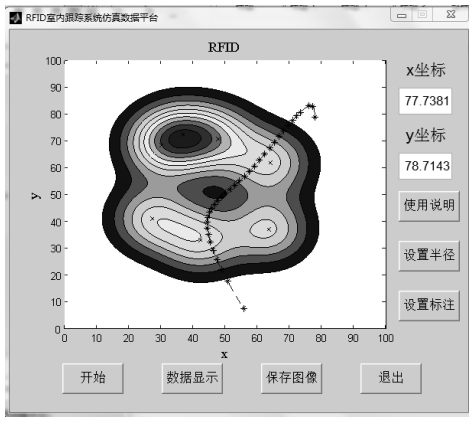


图 6-17　RFID 阅读器测量半径为 150 时平台的输出界面

对比图 6-16 和图 6-17 可以看出，当测量半径为 10 时，设置同样数目的阅读器，跟踪系统所能监测到的范围不同，并且检测率也不相同，半径越大，监测范围越大，检测率也越高，本软件可以实现 0～5000 的半径值，其精度范围能达到普通室内跟踪系统的仿真要求。

5）get_data_xls 函数

本模块实现显示及存储第一部分所产生数据的功能。在数据生成器 RFID 室内跟踪系统的阅读器、目标真实轨迹以及 RFID 阅读器对轨迹的测量数据分别存储在 data.xls、truedata.xls 以及 RFIDdata.xls 的表格中，通过调用 MATLAB 中 xlsread 函数可以对 excel 文件数据进行读取，为达到简洁、快速的目的，创建了数据显示窗口，以表格形式同时显示三类数据，保留了 figure 窗口的工具栏，数据显示窗口可以实现保存、编辑、复制等功能。如图 6-18～图 6-19 所示，分别为在测量半径为 60 时 RFID 室内跟踪系统仿真图及对应的数据显示窗口。

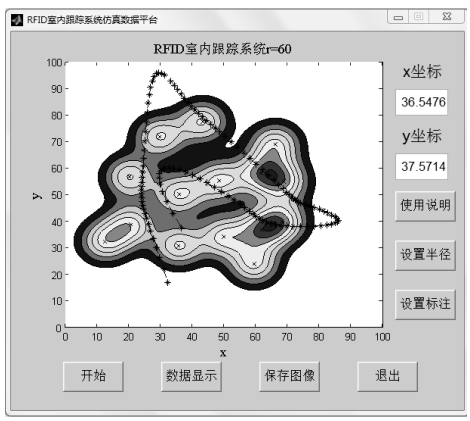


图 6-18　测量半径为 60 时 RFID 室内跟踪系统仿真图

图 6-19　数据显示窗口

如图 6-19 所示，“RFID 阅读器的位置”表格中保存了本次在 RFID 室内跟踪系统中仿真得到的 15 个 RFID 阅读器的位置（x 坐标，y 坐标）；“运动目标实际轨迹”表示在系统中存储的目标运动轨迹坐标，本次仿真放置的实际目标个数为 19 个；“阅读器与可测量实际目标间的距离”表示系统测得的阅读器与目标间的距离，行表示第 i 个实际目标，列表示第 j 个阅读器，如第 70 个实际目标到第 13 个阅读器间的距离为 24.6873，而到第 12、14、15 个阅读器的距离为 0。

6）savePic 函数

完整的 RFID 室内跟踪系统仿真图往往需要在其他文档中被引用，为方便获取图像，本软件设计了图像保存功能，可以实现 fig 及 bmp 的常用图像保存格式，运行后出现的图片保存地址选择对话框如图 6-20 所示。保存为 jpg、bmp 格式的图像分别如图 6-21～图 6-22 所示。

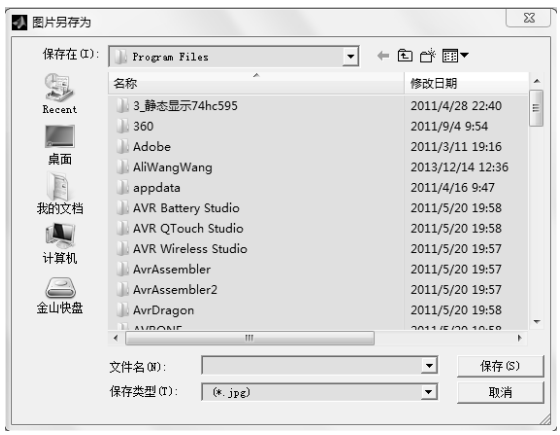


图 6-20　图片保存地址选择对话框

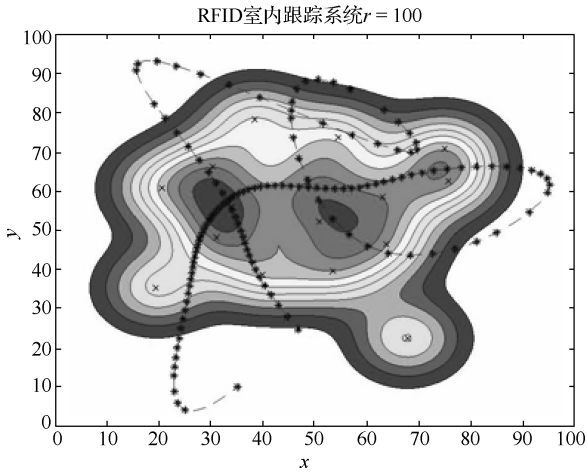


图 6-21　fig 格式保存的图像

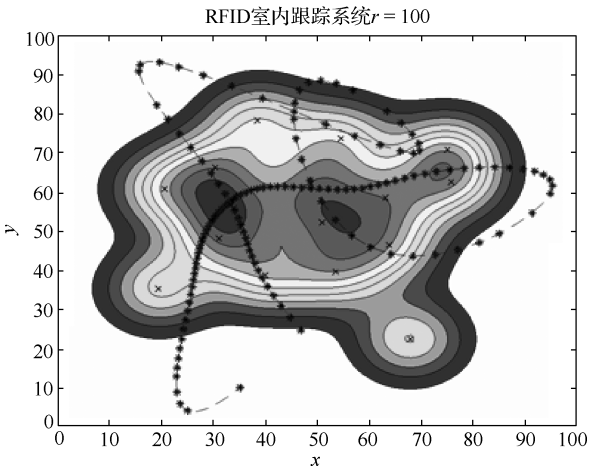


图 6-22　bmp 格式保存的图像

7）closeQuest 函数

本模块创建了一个提问对话框，进一步确认是否要关闭窗口，通过调用函数 questdlg('确认退出当前窗口？','退出确认','Yes','No','No'); 实现关闭 GUI 的对话框。运行结果如图 6-23 所示。

图 6-23　关闭窗口对话框

清除所有数据的函数如下：

```
clear all;
close all;
clc;
```

单击“Yes”则软件关闭，否则，本软件可继续工作。

2. 软件主要功能

软件的主要功能及其关系如图 6-24 所示，下面简单说一下各部分的作用。

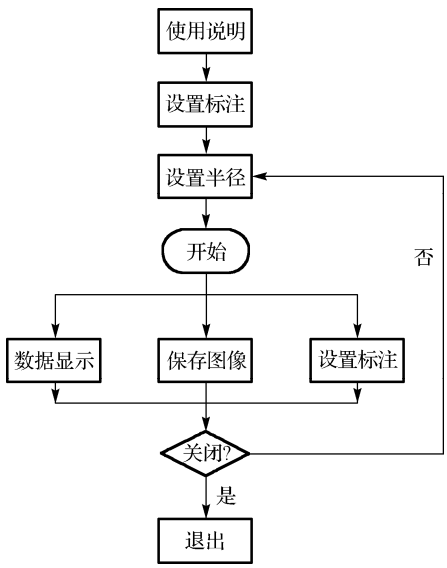


图 6-24　软件主要功能及其关系

1）使用说明

软件在 MATLAB 软件上运行后即可使用，对于初次使用者，本软件人性化地设计了使用向导，单击“使用说明”，用户即可通过弹出的文本图形对本软件的使用方法进行学习。

2）设置半径

用户可以对室内 RFID 跟踪系统的测量精度进行设置，本软件提供的可设置半径范围为 0～5000（默认半径为 50），并且在半径为 2400 时测量范围已覆盖整个坐标系，即室内目标均能跟踪，此时只需要在室内正中央设置一个跟踪器即可，同理，半径越小，测量范围越小，欲达到同样的跟踪效果，需要设置的跟踪器相对较多。

3）设置标注

本控件可以实现对坐标系的横、纵坐标及标题的更改，用户可设定参数进行相关技术的使用，默认的横坐标为 x，纵坐标为 y，标题为 RFID。

4）开始

单击“开始”控件，产生室内数据，并对位置信息进行实时显示，同时保存到 excel 文档中。在坐标系内单击左键，仿真安装室内 RFID 跟踪系统的阅读器位置，单击右键结束放置，每一个阅读器的高检测率的测量空间，只显示颜色（从红到蓝的等高线图）来表示产生的测量范围，继续单击左键产生目标的实际轨迹，单击右键得到阅读器与所能测量轨迹间的距离数据。

5）显示数据

本控件显示软件最新存储的二维空间数据，以 RFID 系统为例，包括阅读器、目标的实际轨迹，阅读器与所能测量到的轨迹间的距离数据。本模块保留了 MATLAB 传统图形的工具栏，可以实现对数据表格的保存、编辑、复制图像等功能。

6）保存图像

本控件主要生成室内 RFID 跟踪系统仿真图像，存储格式设置为 jpg 和 bmp 两种格式，二者特点见表 6-1。

表 6-1　jpg 和 bmp 格式图像的特点

图像文件格式	特点
bmp	无压缩、占用空间大，不合适网络传输及低速设备传输
jpg	文件大、传输快、流行、有损压缩、感兴趣区域

7）关闭

本控件实现对 GUI 界面的关闭，单击该按钮，关闭软件的所有图像，并清除所有数据。若不关闭界面，则可根据需要继续使用本软件。

RFID 室内跟踪系统仿真数据软件源代码见本章附录。

6.5　本 章 小 结

本章给出了系统仿真需要的一些数据，并给出两种典型系统，GPS 和 RFID 系统的测量模型及数据。在跟踪系统仿真研究中，这些数据是非常有用的。这些数据可以作为系统的实际测量值和目标的真实轨迹参考数据，来定量评估估计方法的性能。除了给出这些数据之外，本章更重要的内容是给出产生这些数据的 MATLAB 程序，熟悉这些程序的读者可以根据自己的需要设计参数，甚至修改程序产生不同的轨迹数据，这些都为目标跟踪仿真研究者提供了很大的便利。

附录　RFID 室内跟踪系统仿真数据软件源代码

```
%生成原始数据产生平台
function picprocess()
% 创建显示的窗口，并移动到屏幕中间。
clc;
```

```
clear;
hFigure = figure('Visible','on','Position',[0 0 600 500],'Resize',
'off',...
    'DockControls','off','Menubar','none','Name',...
    'RFID 室内跟踪系统仿真数据平台','NumberTitle', 'off',
'WindowButtonDownFcn',...
   @btnDown,'WindowButtonUpFcn',@btnUp,'CloseRequestFcn',
@closeQuest);
movegui(hFigure,'center');
hAxes = axes('Visible','on','Position',[0.12 0.22 0.7 0.7],'Drawmode',
'fast');
axis([0 100 0 100]);
setappdata(hFigure,'R_Width',50);
axes_1 = {'RFID','x','y'};
setappdata(hFigure,'axes_1',axes_1);
setappdata(hFigure,'xLim',get(hAxes,'xLim'));
setappdata(hFigure,'yLim',get(hAxes,'yLim'));
set(0,'DefaultUicontrolFontSize',13);
uicontrol('String','开始','Position',[70 25 80 40],'Callback',@reader);
uicontrol('String','设置半径','Position',[510 185 80 40],'Callback',
@Range);
uicontrol('String','使用说明','Position',[510 250 80 40],'Callback',
@Instructions);
uicontrol('String','保存图像','Position',[330 25 80 40],'Callback',
{@savePic,hAxes});
uicontrol('String','数据显示','Position',[200 25 80 40],'Callback',
@get_data_xls);
uicontrol('String','设置标注','Position',[510 120 80 40],'Callback',
@change_tittle);
uicontrol('String','退出','Position',[460 25 80 40],'Callback',
'close(gcbf)');
set(0,'DefaultuicontrolBackgroundColor',get(hFigure,'color'));
uicontrol('Style','text','string','x 坐标','fontsize',16,'Position',
[515 430 60 32]);
uicontrol('Style','text','string','y 坐标','fontsize',16,'Position',
[515 345 60 32]);
set(hFigure,'Visible','on');
end

%设置测量半径。

function Range(～,～)
```

```
prompt='设置测量半径：';
name='半径设置';
numlines=1;
default = {'50'};
answer=inputdlg(prompt,name,numlines,default);
if ～isempty(answer)
R_Width = floor(str2double(answer));
    if ～isnan(R_Width)&&R_Width>0&&R_Width<5000
        setappdata(gcf,'R_Width',R_Width);
   end
end
end

%产生 RFID 阅读器数据。

function reader(～,～)
cla;
axes_1 =getappdata(gcf,'axes_1');
title(axes_1{1},'Fontname','Times New Roman');
xlabel(axes_1{2},'Fontname','Times New Roman');
ylabel(axes_1{3},'Fontname','Times New Roman');
set(get(gca,'xlabel'),'fontsize',13);
set(get(gca,'ylabel'),'fontsize',13);
set(get(gca,'title'),'fontsize',13);
delete data.xls;
delete truedata.xls;
delete RFIDdata.xls;
n = 0;
R_Width = getappdata(gcf,'R_Width');
reader = 1;
while reader == 1
    axis([0 100 0 100]);
    title(axes_1{1},'Fontname','Times New Roman');
    xlabel(axes_1{2},'Fontname','Times New Roman');
    ylabel(axes_1{3},'Fontname','Times New Roman');
    set(get(gca,'xlabel'),'fontsize',13);
    set(get(gca,'ylabel'),'fontsize',13);
    set(get(gca,'title'),'fontsize',13);
    [rxi,ryi,reader] = ginput(1);
    plot(rxi,ryi,'kx')
    hold on;
    n = n+1;
```

```
    readerxy(:,n) = [rxi;ryi];
    text(rxi,ryi,'');
    uicontrol('style','edit','enable','inactive','BackgroundColor',
...
    'w','horizontal','right','position',[510 390 70 35],'string',...
    num2str(readerxy(1,n),6));
    uicontrol('style','edit','enable','inactive','BackgroundColor',
...
    'w','horizontal','right','position',[510 305 70 35],'string',...
    num2str(readerxy(2,n),6));
    xlswrite('data.xls',readerxy);
end
X1 = [0:1:100];
X2 = [0:1:100];
[X,Y] = meshgrid(X1,X2);
Z = zeros(size(X));
readerlocation=readerxy;
readermc=R_Width.*ones(size(readerxy(1,:)));
for i=1:length(readermc)
    Z = Z+exp(-((X-readerlocation(1,i)).*(X-readerlocation(1,i))+...
       (Y-readerlocation(2,i)).*(Y-readerlocation(2,i)))/readermc
(i));
end
[c,h]=contourf(X,Y,Z);
ch=get(h,'children');
set(min(ch),'FaceColor','w','FaceAlpha',0.1)
n=0;
but = 1;
while but == 1
    [xi,yi,but] = ginput(1);
    plot(xi,yi,'k*')
    n = n+1;
    xy(:,n) = [xi;yi];
    uicontrol('style','edit','enable','inactive','BackgroundColor',
...
      'w','horizontal','right','position',[510 390 70 35],'string',...
       num2str(xy(1,n),6));
    uicontrol('style','edit','enable','inactive','BackgroundColor',
...
      'w','horizontal','right','position',[510 305 70 35],'string',...
    num2str(xy(2,n),6));
    xlswrite('truedata.xls',xy);
```

```
end
t=1:n;
tss = 1: 0.1: n;
xyss = spline(t,xy,tss);
plot(xyss(1,:),xyss(2,:),'--');
hold on

ap=4;r=3;
for i=1:length(readerxy(1,:))
    for j=1:length(xyss(1,:))
   d(i,j)=sqrt((xyss(:,j)-readerxy(:,i))'*(xyss(:,j)-readerxy(:,i)));
   if rand(1)>exp(-d(i,j).^2/2/15/15);,
       d(i,j)=NaN;
   end
   dmm(i,j)= d(i,j)+randn(1)*(0.2303*ap/r)*d(i,j);
    end
end

ts=[];dm=[];xys=[];
for ii=1:length(tss);
    for jj=1:length(dmm(:,1));
        if isnan(dmm(jj,ii))

        else
            ts=[ts tss(ii)];
            dm=[dm dmm(:,ii)];
            xys=[xys xyss(:,ii)];

            break
        end
    end
end
plot(xys(1,:),xys(2,:),'k*');
xlswrite('RFIDdata.xls',dmm);

end

%设置鼠标按键操作。

function btnUp(hObject,～)
%窗口的 WindowButtonUpFcn 回调函数。
setappdata(hObject,'isPressed',false);
```

```
end

function btnDown(hObject,～)
%窗口的 WindowButtonDownFcn 回调函数
xLim = getappdata(hObject,'xLim');
yLim = getappdata(hObject,'yLim');
pos = get(gca,'CurrentPoint');
if(pos(1,1)>xLim(1))&&(pos(1,1)<xLim(2))&&(pos(1,2)>yLim(1))&&…
(pos(1,2)<yLim(2))
   set(hObject,'Pointer','hand');
else
   set(hObject,'Pointer','arrow');

end
end

%RFID 数据产生平台软件的使用说明。

function Instructions(～,～)
f = figure('Color',[0.8,0.8,0.8],'Position',[960 350 385 330],
'NumberTitle','off','Name',…
'数据发生器的使用说明');
set(f,'Toolbar','none','Menubar','none');
axis off;
text('string','1.单击“设置标注”可以设置坐标轴和标题；',...
    'fontsize',12,'position',[-0.15 1]);
text('string','2.单击“设置半径”可以设置测量范围；',...
    'fontsize',12,'position',[-0.15 0.9]);
text('string','3.单击“开始”可以采集数据，其中，左键产',...
    'fontsize',12,'position',[-0.15 0.8]);
text('string','生数据，右键停止采集；继续按左键产生目标的',...
    'fontsize',12,'position',[-0.15 0.7]);
text('string','实际轨迹数值，右键结束并产生阅读器与可测量',...
    'fontsize',12,'position',[-0.15 0.6]);
text('string','目标间距离值；', 'fontsize',13,'position',[-0.15 0.5]);
text('string','4.单击“显示数据”可以显示保存的采集数据时的',...
    'fontsize',12,'position',[-0.15 0.4]);
text('string','坐标值；','fontsize',13,'position',[-0.15 0.3]);
text('string','5.单击“保存图像”可以以.jpg 和.bpm 格式保存',...
    'fontsize',12,'position',[-0.15 0.2]);
text('string','图像；', 'fontsize',13,'position',[-0.15 0.1]);
text('string','6.单击“关闭”按钮则关闭数据产生平台，并清空；',...
```

```
'fontsize',12,'position',[-0.15 0]);
end

%保存 RFID 数据平台仿真图像。

function savePic(~,~,hAxes)
 [fName,pName,index] = uiputfile({'*.jpg';'*.bmp'},'图片另存为');
if index == 1||index==2
    hFig=figure('Visible','off');
    copyobj(hAxes,hFig);
    str = [pName fName];
    if index ==1
        print(hFig,'-djpeg',str);
    else
        print(hFig,'-dbmp',str);
    end
delete(hFig);
   hMsg = msgbox(['图片保存成功!'],'提示');
     pause(2);
     if ishandle(hMsg)
         delete(hMsg);
    end
end
end

%设置输出图像的标注。

function change_tittle(~,~)
prompt={'设置标题名称：','设置横坐标名称：','设置纵坐标名称：'};
name='设置标注';
numlines=1;
defaultanswer={'RFID','x','y'};
answer1=inputdlg(prompt,name,numlines,defaultanswer);
if ~isempty(answer1)
   title(answer1{1},'Fontname','Times New Roman');
   xlabel(answer1{2},'Fontname','Times New Roman');
   ylabel(answer1{3},'Fontname','Times New Roman');
   set(get(gca,'xlabel'),'fontsize',13);
   set(get(gca,'ylabel'),'fontsize',13);
   set(get(gca,'title'),'fontsize',13);
   axes_1= answer1;
   setappdata(gcf,'axes_1',axes_1);
```

```
end
end


%显示平台产生的数据。

function get_data_xls(～,～)
f = figure('Color',[0.8,0.8,0.8],'Position',[300 300 500 405], 'Number
Title','off','Name','data');
set(0,'DefaultuicontrolBackgroundColor',get(f,'color'));
uicontrol('Style','text','string','RFID 阅读器的位置','fontsize',13,...
    'Position',[38 375 150 25]);
uicontrol('Style','text','string','运动目标实际轨迹','fontsize',13,....
    'Position',[38 258 150 25]);
uicontrol('Style','text','string','阅读器与可测量实际目标间的距离',...
'fontsize',13, 'Position',[38 141 300 25]);
[～,～,raw] = xlsread('data.xls');
ColumnName = raw(1,:);
data= raw(1:end,:);
for i = 1:numel(data)
    if isnan(data{i})
        data{i} ='';
    end
end

rnames = {'X','Y'};
uitable('Parent',f,'Data',data,'RowName',rnames,'Position', [40 288
430 85],'Fontsize',10);
[～,～,raw] = xlsread('truedata.xls');
ColumnName = raw(1,:);
data1 = raw(1:end,:);
for i = 1:numel(data1)
    if isnan(data1{i})
        data1{i} ='';
    end
end
rnames1 = {'X','Y'};
uitable('Parent',f,'Data',data1,'RowName',rnames1,'Position',…
[40 171 43 85],'Fontsize',10);
[～,～,raw] = xlsread('RFIDdata.xls');
ColumnName = raw(1,:);
data2= raw(1:end,:);
for i = 1:numel(data2)
    if isnan(data2{i})
```

```
        data2{i} ='0';
    end
end

uitable('Parent',f,'Data',data2,'Position',[40 14 430 125],'Fontsize',10);
end

%关闭软件平台。

function closeQuest(hObject,～)
% 创建一个提问对话框，进一步确认是否要关闭窗口。
sel = questdlg('确认退出当前窗口？','退出确认','Yes','No','No');
 switch sel
    case'Yes'
        delete(hObject);
clear all;
close all;
clc;
    case'No'
      return;
 end

end
```

第 7 章　机动目标动力学模型

目标跟踪是 Kalman 滤波器的典型应用，通过使用传感器的测量数据来估计机动目标的位移、速度、加速度等运动特征。从第 6 章的估计方法可以看到，针对一个具体的跟踪问题，需要知道传感器的测量模型，还需要确定机动目标的运动过程，进而确定过程模型，这样再使用 Kalman 滤波器就可以估计运动的特征了。

利用 Kalman 滤波器获得准确估计需要准确的系统模型，例如，当系统做匀速运动或匀加速运动时，应该采用匀速（Constant Velocity，CV）模型和匀加速（Constant Acceleration，CA）模型。但在实际应用中经常出现系统加速度非常数的情况，例如，飞机雷达对地面目标或海上目标进行跟踪时，在目标运动过程中驾驶员的人为动作或者控制指令随时会使目标出现转弯、闪避等动作，因此，目标不可能一直做匀速或者匀加速运动，这种很“随意”的运动称为“机动”。为了保证较好的轨迹跟踪结果，研究机动目标的运动模型是十分必要的。

本章假设系统的测量传感器具有相同的采样周期，本章介绍了运动目标的连续模型、离散化方法及离散模型，给出 CV 模型、CA 模型、Singer 模型、当前模型、Jerk 模型、交互多模型等机动目标跟踪模型等常用方法。上述模型的共同特点是假设目标加速度具有已知的概率密度函数，如 Singer 模型假设加速度在一定区间服从均匀分布，但实际上这种假设得到的加速度参数和实际的目标机动运动参数并不一定相一致。因此，本章给出了不需要设置参数的自适应机动目标运动模型，并利用该模型进行估计，通过仿真的方法详细研究该模型的性能。

作为机动目标工作的基础，机动目标运动模型有着悠久的研究历史。自从 1970 年 Singer 提出 Singer 模型以来，研究者相继提出了“当前”统计模型、Jerk 模型及交互多模型等机动目标模型，很多著作对这些模型的性能进行了详细论述，例如，给出了 CV 模型、CA 模型、Singer 模型、“当前”统计模型、Jerk 模型及交互多模型的状态方程，并利用仿真分析了各模型的性能，读者若感兴趣，可参考相关文献。为保证本书的完整性，本章借鉴相关文献，将其研究成果进行罗列，并加以丰富，其目的在于向读者展现完整的机动模型研究基础内容。

未特殊说明，过程模型状态变量 $\boldsymbol{x}(t)$ 设为目标的位置、速度及加速度，表示为 $\boldsymbol{x}(t)=[x(t),\dot{x}(t),\ddot{x}(t)]^{\mathrm{T}}$，其中 $x(t)$ 为目标 t 时刻的位移，$\dot{x}(t)$ 为目标 t 时刻的速度，$\ddot{x}(t)$ 为目标 t 时刻的加速度。不同形式的机动目标模型通过假设目标速度及加速度满足不同的统计特性进行区别，作为目标跟踪的系统模型是一种数学关系，该模型与实际目标运动规律的一致程度越高，应用状态估计方法得到的跟踪性能越好。即目标机动模

型并非越复杂越好，而是与实际跟踪目标越一致越好。因此，研究者从最简单的 CV 模型开始，根据不同的目标特性逐渐丰富模型的描述方法，以期模型能和实际运动目标更加一致。

7.1　CV 模型

CV 模型假设目标做匀速直线运动，即目标加速度的理想值为 0，但由于干扰的存在，加速度不能维持在 0 值，而是零均值高斯白噪声 $w_c(t)$，即 $w_c(t) \sim N(0,\sigma^2)$，用公式表示加速度的变化为

$$\ddot{x}(t) = w_c(t)$$

将上式写为连续时间系统状态方程为

$$\begin{bmatrix} \dot{x}(t) \\ \ddot{x}(t) \end{bmatrix} = \begin{bmatrix} 0 & 1 \\ 0 & 0 \end{bmatrix} \begin{bmatrix} x(t) \\ \dot{x}(t) \end{bmatrix} + \begin{bmatrix} 0 \\ 1 \end{bmatrix} w_c(t) \tag{7-1}$$

即状态转移矩阵 $\boldsymbol{F}$ 和噪声转移矩阵 $\boldsymbol{G}$ 分别为

$$\boldsymbol{F} = \begin{bmatrix} 0 & 1 \\ 0 & 0 \end{bmatrix}, \quad \boldsymbol{G} = \begin{bmatrix} 0 \\ 1 \end{bmatrix} \tag{7-2}$$

由于根据机动模型进行目标跟踪一般使用离散的滤波器，所以有必要得到形如式（6-17）或式（6-18）的离散系统模型状态方程的参数。根据第 6 章介绍的离散化方法，利用式（6-10）得到

$$\boldsymbol{A} = \mathrm{e}^{\begin{bmatrix} 0 & T_0 \\ 0 & 0 \end{bmatrix}} \tag{7-3}$$

式（7-3）为指数函数，由于 $\boldsymbol{F}$ 为不满秩矩阵，不能使用特征值方法求指数函数，而是使用拉普拉斯的方法求得 $(s\boldsymbol{I} - \boldsymbol{F})^{-1}$ 为

$$(s\boldsymbol{I} - \boldsymbol{F})^{-1} = \begin{bmatrix} s & -1 \\ 0 & s \end{bmatrix}^{-1} = \frac{1}{s^2} \begin{bmatrix} s & 1 \\ 0 & s \end{bmatrix} = \begin{bmatrix} \dfrac{1}{s} & \dfrac{1}{s^2} \\ 0 & \dfrac{1}{s} \end{bmatrix} \tag{7-4}$$

对式（7-4）进行拉普拉斯逆变换，并取 $t = T_0$ 得到

$$\boldsymbol{A} = \begin{bmatrix} 1 & T_0 \\ 0 & 1 \end{bmatrix} \tag{7-5}$$

下面分别求两类离散模型的过程噪声方差，由于处理噪声的方式不同，得到的离散模型过程噪声方差也会有所不同，这里省略 $\delta(k-u)$。

对于 DCM，根据式（6-12）得到

$$\begin{aligned}\boldsymbol{Q}&=E[\boldsymbol{w}(k)\boldsymbol{w}^{\mathrm{T}}(u)]=\int_0^{T_0}\mathrm{e}^{\boldsymbol{F}(T_0-\lambda)}\boldsymbol{G}\cdot\sigma^2\cdot\boldsymbol{G}^{\mathrm{T}}[\mathrm{e}^{\boldsymbol{F}(T_0-\lambda)}]^{\mathrm{T}}\mathrm{d}\lambda\\&=\int_0^{T_0}\begin{bmatrix}1&T_0-\lambda\\0&1\end{bmatrix}\begin{bmatrix}0\\1\end{bmatrix}\cdot\sigma^2\cdot[0\quad 1]\begin{bmatrix}1&0\\T_0-\lambda&1\end{bmatrix}\mathrm{d}\lambda\\&=\int_0^{T_0}\begin{bmatrix}(T_0-\lambda)^2&T_0-\lambda\\T_0-\lambda&1\end{bmatrix}\sigma^2\mathrm{d}\lambda\\&=\begin{bmatrix}\frac{1}{3}T_0^3&\frac{1}{2}T_0^2\\\frac{1}{2}T_0^2&T_0\end{bmatrix}\sigma^2\end{aligned}\tag{7-6}$$

对于 DDM，根据式（6-14）求离散时间模型的过程噪声方差，这里先求 $\boldsymbol{B}=\int_0^{T_0}\mathrm{e}^{\boldsymbol{F}(T_0-\lambda)}\boldsymbol{G}\mathrm{d}\lambda$，得

$$\boldsymbol{B}=\int_0^{T_0}\mathrm{e}^{\boldsymbol{F}(T_0-\lambda)}\boldsymbol{G}\mathrm{d}\lambda=\int_0^{T_0}\begin{bmatrix}1&T_0-\lambda\\0&1\end{bmatrix}\begin{bmatrix}0\\1\end{bmatrix}\mathrm{d}\lambda=\int_0^{T_0}\begin{bmatrix}T_0-\lambda\\1\end{bmatrix}\mathrm{d}\lambda=\begin{bmatrix}\frac{1}{2}T_0^2\\T_0\end{bmatrix}\tag{7-7}$$

则求得离散时间 DDM 的过程噪声方差 $\boldsymbol{Q}_s$ 为

$$\boldsymbol{Q}_s=\boldsymbol{B}\cdot\sigma^2\cdot\boldsymbol{B}^{\mathrm{T}}=\begin{bmatrix}\frac{1}{4}T_0^4&\frac{1}{2}T_0^3\\\frac{1}{2}T_0^3&T_0^2\end{bmatrix}\sigma^2\tag{7-8}$$

发现式（7-6）和式（7-8）得出的离散模型过程噪声方差并不相同，虽然都是基于同一个连续模型离散化得到，第 6 章曾经提到，由于这两类模型对噪声的处理并不相同，DCM 对连续过程协方差求积分，计算的是过程噪声累积效果；DDM 假设连续时间噪声为分段时常噪声，使用增益矩阵 $\boldsymbol{B}$ 来体现分段噪声的累积效果，因此导致过程噪声方差计算结果不同。从理论上讲，DCM 更为科学、合理，因为过程噪声在整个采样区间并不一定满足 DDM 的假设，所以会更符合一般的噪声过程。但从计算角度讲，DCM 计算更复杂一些，这从前面的计算过程就可以知道。

下面给出 CV 的 DCM 的 MATLAB 程序，这是一个函数，输入变量 T 表示采样周期 T_0，qq 表示 σ^2，输出为离散过程模型矩阵 $\boldsymbol{A}$ 及过程噪声方差 $\boldsymbol{Q}$，分别用 A 和 Q 表示。

```
function [A,Q]=CVmodel(T,qq)
A=[1 T;0 1];
Q=[T^3/3 T^2/2;T^2/2 T]*qq;
```

7.2　CA 模型

CA 模型假设目标做匀加速直线运动，即理想的目标加速度值没有变化，其导数为 0，但由于干扰的存在，加速度导数变化也不能维持在 0 值，而是满足零均值高斯白噪声 $w_c(t)$[$w_c(t)$ 假设如 7.1 节，即 $w_c(t) \sim N(0,\sigma^2)$]作用的结果，用公式表示加速度导数的变化为

$$\dddot{x}(t) = w_c(t) \tag{7-9}$$

可以得到连续系统模型为

$$\begin{bmatrix} \dot{x}(t) \\ \ddot{x}(t) \\ \dddot{x}(t) \end{bmatrix} = \begin{bmatrix} 0 & 1 & 0 \\ 0 & 0 & 1 \\ 0 & 0 & 0 \end{bmatrix} \begin{bmatrix} x(t) \\ \dot{x}(t) \\ \ddot{x}(t) \end{bmatrix} + \begin{bmatrix} 0 \\ 0 \\ 1 \end{bmatrix} w_c(t) \tag{7-10}$$

即

$$\boldsymbol{F} = \begin{bmatrix} 0 & 1 & 0 \\ 0 & 0 & 1 \\ 0 & 0 & 0 \end{bmatrix}, \quad \boldsymbol{G} = \begin{bmatrix} 0 \\ 0 \\ 1 \end{bmatrix} \tag{7-11}$$

下面利用与 7.1 节相似的方法来计算离散模型中的状态转移 $\boldsymbol{A}$，先利用拉普拉斯变换得到

$$(s\boldsymbol{I} - \boldsymbol{F})^{-1} = \begin{bmatrix} s & -1 & 0 \\ 0 & s & -1 \\ 0 & 0 & s \end{bmatrix}^{-1} = \begin{bmatrix} \frac{1}{s} & \frac{1}{s^2} & \frac{1}{s^3} \\ 0 & \frac{1}{s} & \frac{1}{s^2} \\ 0 & 0 & \frac{1}{s} \end{bmatrix} \tag{7-12}$$

经过逆变换并取 $t = T_0$ 得到

$$\boldsymbol{A} = \mathrm{e}^{\boldsymbol{F}T_0} = \begin{bmatrix} 1 & T_0 & \frac{T_0^2}{2} \\ 0 & 1 & T_0 \\ 0 & 0 & 1 \end{bmatrix} \tag{7-13}$$

DCM 的过程噪声方差 $\boldsymbol{Q}$ 为

$$\boldsymbol{Q}=E[\boldsymbol{w}(k)\boldsymbol{w}^{\mathrm{T}}(u)]=\int_0^{T_0}\mathrm{e}^{\boldsymbol{F}(T_0-\lambda)}\boldsymbol{G}\cdot\sigma^2\cdot\boldsymbol{G}^{\mathrm{T}}[\mathrm{e}^{\boldsymbol{F}(T_0-\lambda)}]^{\mathrm{T}}\mathrm{d}\lambda$$

$$=\int_0^{T_0}\begin{bmatrix}1 & T_0-\lambda & \dfrac{(T_0-\lambda)^2}{2}\\ 0 & 1 & T_0-\lambda\\ 0 & 0 & 1\end{bmatrix}\begin{bmatrix}0\\0\\1\end{bmatrix}\cdot\sigma^2\cdot\begin{bmatrix}0 & 0 & 1\end{bmatrix}\begin{bmatrix}1 & 0 & 0\\ T_0-\lambda & 1 & 0\\ \dfrac{(T_0-\lambda)^2}{2} & T_0-\lambda & 1\end{bmatrix}\mathrm{d}\lambda$$

$$=\int_0^{T_0}\begin{bmatrix}\dfrac{(T_0-\lambda)^4}{4} & \dfrac{(T_0-\lambda)^3}{2} & \dfrac{(T_0-\lambda)^2}{2}\\ \dfrac{(T_0-\lambda)^3}{2} & (T_0-\lambda)^2 & T_0-\lambda\\ \dfrac{(T_0-\lambda)^2}{2} & T_0-\lambda & 1\end{bmatrix}\sigma^2\mathrm{d}\lambda$$

$$=\begin{bmatrix}\dfrac{T_0^5}{20} & \dfrac{T_0^4}{8} & \dfrac{T_0^3}{6}\\ \dfrac{T_0^4}{8} & \dfrac{T_0^3}{3} & \dfrac{T_0^2}{2}\\ \dfrac{T_0^3}{6} & \dfrac{T_0^2}{2} & T_0\end{bmatrix}\sigma^2 \tag{7-14}$$

DDM 的过程噪声方差为

$$\boldsymbol{B}=\int_0^{T_0}\mathrm{e}^{\boldsymbol{F}(T_0-\lambda)}\boldsymbol{G}\mathrm{d}\lambda=\int_0^{T_0}\begin{bmatrix}1 & T_0-\lambda & \dfrac{(T_0-\lambda)^2}{2}\\ 0 & 1 & T_0-\lambda\\ 0 & 0 & 1\end{bmatrix}\begin{bmatrix}0\\0\\1\end{bmatrix}\mathrm{d}\lambda$$

$$=\int_0^{T_0}\begin{bmatrix}\dfrac{(T_0-\lambda)^2}{2}\\ T_0-\lambda\\ 1\end{bmatrix}\mathrm{d}\lambda=\begin{bmatrix}\dfrac{T_0^3}{6}\\ \dfrac{T_0^2}{2}\\ T_0\end{bmatrix} \tag{7-15}$$

则仿照 CV 的方法，求得离散时间 DDM 的过程噪声方差 $\boldsymbol{Q}_s$ 为

$$\boldsymbol{Q}_s=\boldsymbol{B}\cdot\sigma^2\cdot\boldsymbol{B}^{\mathrm{T}}=\begin{bmatrix}\dfrac{T_0^6}{36} & \dfrac{T_0^5}{12} & \dfrac{T_0^4}{6}\\ \dfrac{T_0^5}{12} & \dfrac{T_0^4}{4} & \dfrac{T_0^3}{2}\\ \dfrac{T_0^4}{6} & \dfrac{T_0^3}{2} & T_0^2\end{bmatrix}\sigma^2 \tag{7-16}$$

DCM 形式 CA 模型的 MATLAB 函数程序如下，输入量 T、qq，以及输出量 A、Q 和 CV 模型一样。注意到，CA 模型的输出为 3 阶，这是和 CV 模型不同的地方，因为 CA 模型的状态变量扩展到了 3 阶，与 CV 模型描述的状态变量相比，多了加速度这个量。

CA 模型的 MATLAB 程序如下，输入、输出变量和函数 CVmodel 一样。

```
function [A,Q]=CAmodel(T,qq)
A=[1 T T^2/2;0 1 T;0 0 1];
Q=[T^5/20 T^4/8 T^3/6; T^4/8 T^3/3 T^2/2;T^3/6 T^2/2 T]*qq;
```

为了详细说明过程运动模型在跟踪系统中的使用，给出三个程序，分别说明如何利用 CV 模型及 CA 模型跟踪直线+圆形机动、蛇形机动以及任意曲线机动的轨迹。每个程序分为四部分，先产生测量数据，进而计算目标的运动模型，然后利用 Kalman 滤波器进行估计，最后画出结果图。由于程序较长，范例将分部分给出每一段程序，解释程序的功能，并给出结果。先利用 CV 模型及 CA 模型跟踪直线+圆形机动为例对程序进行详细说明，程序中系统采样周期设置为 1s，假设运动目标先进行 31s 的直线运动，再进行 57s 的圆周运动。程序的第一部分调用第 6 章介绍的函数 funtrackinglinecircle 来产生仿真测量数据。

```
%本程序名为 C7-1.m
clc
clear
%测量参数：周期和测量方差的设置。
T=1;
R=40^2;
%圆形机动的测量值。
Tt=31;Tz=57;[t,y]=funtrackinglinecircle(T,Tt,Tz,R);
```

接下来选择并计算系统参数，可以选择 CV 模型或 CA 模型，程序中使用 MATLAB 的注释符号%来简单地设置，先给出 CV 模型和 CA 模型的两个函数，然后使用“%”来注释掉一个，只使用其中的一个模型。由于不知道过程模型中的参数“方差”，因此仿真中设定不同的值进行计算，进而分析不同的参数值得到的估计结果有什么不同。因为仿真中轨迹的真值是已知的，可以用估计结果的方差来定量分析估计结果的好坏。一般这样认为，与最小方差相对应的过程模型参数是最合适的。σ^2（程序中用 qq 表示该变量）可以设置为不同的值，下面的程序中将其设置为 1，读者可以自行修改该语句来改变σ^2 的值。

```
%选择模型。
qq=1;%此值的选择对性能影响很大，可以取不同的值，如取 0.1、1、10 等，直接修改程序
     %中的值即可。
```

```
 [A,Q]=CVmodel(T,qq);C=[1 0];
% [A,Q]=CAmodel(T,qq);C=[1 0 0]; %这两个模型在执行的时候选其中的一个
```

接下来，需要利用 Kalman 滤波器估计状态，并对横、纵轴分别估计。在使用 Kalman 滤波器之前，先对初值进行设置，将状态的初值设置为全 0 向量，状态估计方差矩阵为单位矩阵的 1000 倍。并且，将每一个采样点的状态估计保存至一个向量，每一个新产生的估计状态都依次放在前一个状态的后一列，因此在设置初值时设置了一个空向量，横轴分别是 xx1 和 xx2，语句为 xx1=[]和 xx2=[]。使用 Kalman 滤波器分别估计横轴和纵轴，这两部分几乎是完全一样的，在估计纵轴的时候，需要重新设置状态的初值、估计方差的初值等，但是系统参数 A、C、R、Q 是一样的。为了区别横纵轴估计状态的输出量，分别使用 xx1 和 xx2 来加以区别，也就是说横轴的状态估计量，位移、速度和加速度是用 xx1 向量来表示的，纵轴的状态估计量，即纵轴的位移、速度和加速度是用 xx2 向量来表示的。然后把传感器的输出以及估计位移的结果画在二维空间当中，CV 模型在 $\sigma^2=1$ 时的估计结果如图 7-1 所示，CV 模型在 $\sigma^2=10$ 时的估计结果如图 7-2 所示，而 CA 模型在 $\sigma^2=1$ 和 $\sigma^2=10$ 的估计结果分别如图 7-3～图 7-4 所示。

```
%使用 Kalman 进行滤波,对于圆形机动，需要横、纵轴分别估计。

%估计横轴。
xe=zeros(length(Q),1);p=1000*eye(size(A));xx1=[];
for i=1:length(t)
[xe,p]=kalmanfun(A,C,Q,R,xe,y(1,i),p);
xx1=[xx1 xe];
end
%估计纵轴。
xe=zeros(length(Q),1);p=1000*eye(size(A));xx2=[];
for i=1:length(t)
[xe,p]=kalmanfun(A,C,Q,R,xe,y(2,i),p)
xx2=[xx2 xe];
end
plot(y(1,:),y(2,:),'*');hold on,plot(C*xx1,C*xx2);hold off
figure
```

虽然从理论上讲，目标运动的位移应该和实际运动的轨迹进行比较，也就是说，$\boldsymbol{P}(k|k)=E[(\boldsymbol{x}(k)-\hat{\boldsymbol{x}}(k|k))(\boldsymbol{x}(k)-\hat{\boldsymbol{x}}(k|k))^{\mathrm{T}}]$，但图中两条线的接近程度也能定性说明哪一个结果会更好。可以明显看出，使用 CA 模型的结果要优于 CV 模型，这是由于目标运动的轨迹包含圆形运动，其加速度不为 0，这不符合 CV 模型的假设前提，所以 CV 估计的效果并不好。而 CA 模型考虑了目标运动中的加速度成分，在拐弯的时候，跟踪的结果明显好于 CV 模型。下面来看一下 CA 模型跟踪蛇形机动的结果。先将采样周期设为 0.3s，因为太大的采样周期会使蛇形机动的轨迹变得不光滑，取采样点数为 30，并设蛇形机动

的幅值为 10，角频率为 $\pi/3$，调用 funtrackingsnake 首先产生测量数据 ym，然后调用 Kalman 滤波器使用 CA 模型进行估计，最后，蛇形机动的真实轨迹与估计结果如图 7-5 所示。下面这一段程序是放在前例程序后面的，因此，不涉及模型方面的内容。

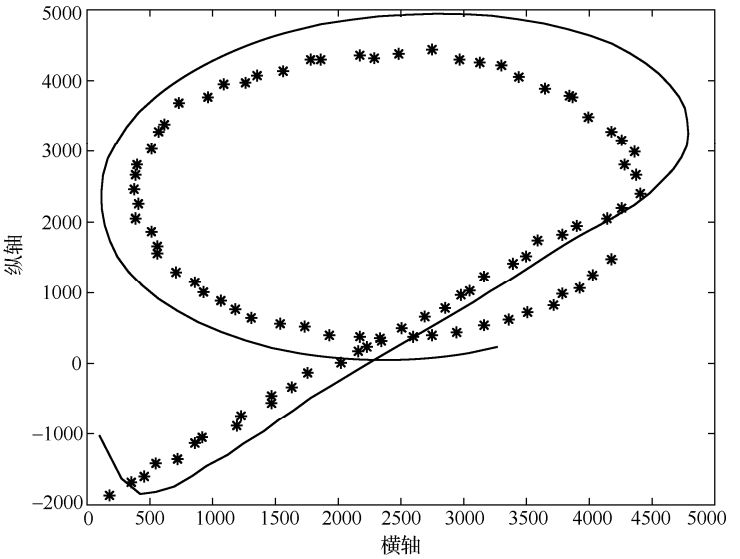


图 7-1　CV 模型在 $\sigma^2=1$ 时的估计结果

*表示带有噪声的测量数据，—表示估计结果

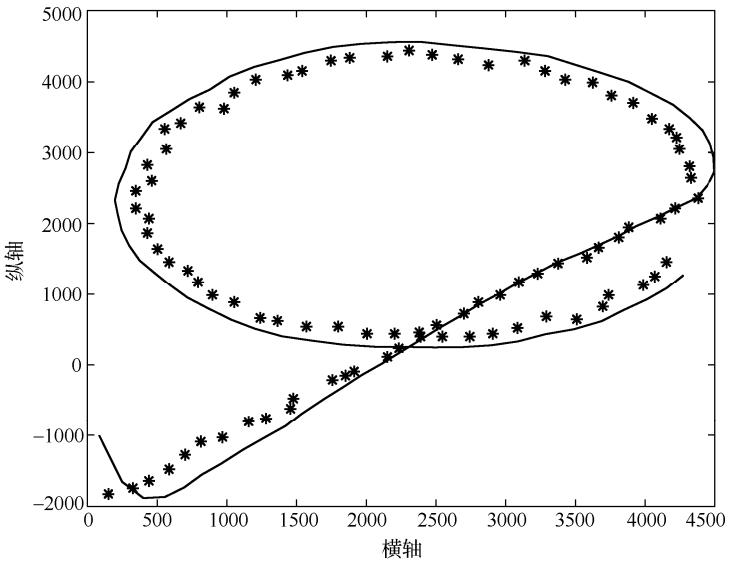


图 7-2　CV 模型在 $\sigma^2=10$ 时的估计结果

*表示带有噪声的测量数据，—表示估计结果

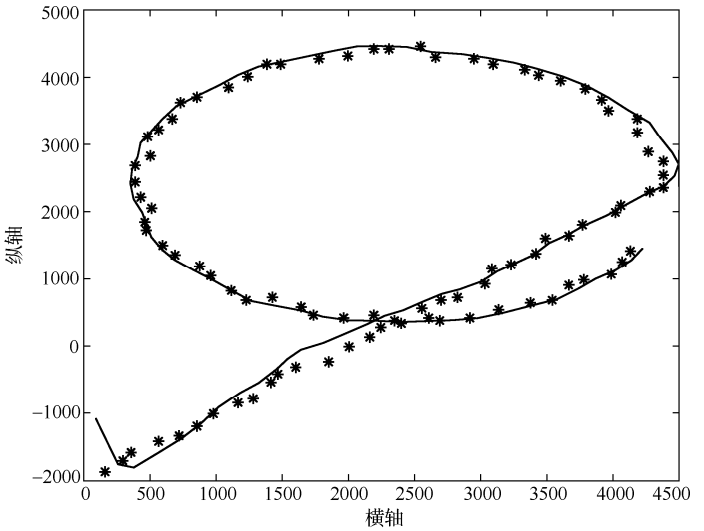


图 7-3　CA 模型在 $\sigma^2=1$ 时的估计结果

*表示带有噪声的测量数据，—表示估计结果

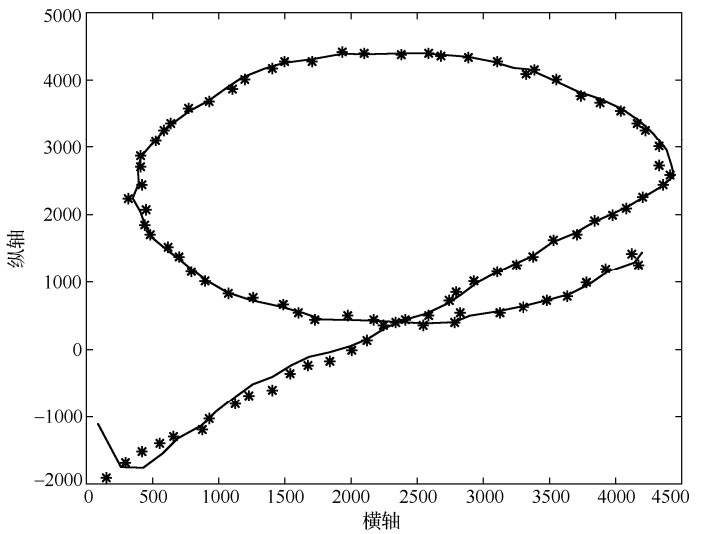


图 7-4　CA 模型在 $\sigma^2=10$ 时的估计结果

*表示带有噪声的测量数据，—表示估计结果

```
%使用 Kalman 滤波器对蛇形机动进行估计。
%产生蛇形机动的测量值。
T=0.3;
t=1:T:30;a=10;omig=pi/3;R=2;
[yreal,ym]=funtrackingsnake(a,omig,t,R);
%估计。
```

```
xe=zeros(length(Q),1);p=1000*eye(size(A));xx1=[];
for i=1:length(t)
[xe,p]=kalmanfun(A,C,Q,R,xe,y(i),p)
xx1=[xx1 xe];
end
plot(t,y,t,C*xx1)
figure
```

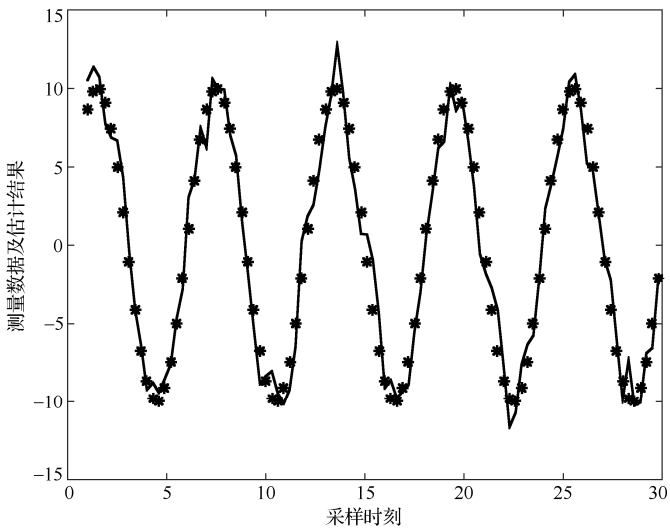


图 7-5　CA 模型在跟踪蛇形机动（参数为 $\sigma^2=10$ ）时的估计结果
*表示带有噪声的测量数据，—表示估计结果

从图 7-5 可以看出，在跟踪蛇形机动拐弯处的时候误差比较大，这说明在这些点上，加速度的噪声分布已经不再满足 CA 模型的假设条件。读者可以尝试使用 CV 模型看一下这两者的结果有何不同。

下面再来看一下使用 CA 模型跟踪任意曲线的结果。在程序中需要先产生任务曲线的待跟踪数据，使用第 6 章产生任意曲线的程序得到目标运动轨迹，然后分别进行横、纵轴的估计，最后给出估计结果图和具体的估计方差数据。

下面这一段程序也放在前面这些程序的后面，因此也不需要再产生 CA 模型，只把必要的参数重写一遍，如 R=20。

```
%对任意曲线进行跟踪。
R=20;
%使用说明：使用鼠标左键可以产生轨迹的各个点，鼠标右键为结束点。
axis([0 100 0 100])
hold on
% 初始化，将存贮测量数据序列 xy 置为空矩阵。
xy = [];
```

```
n = 0;
% 循环，采集鼠标的坐标点。
disp('单击鼠标左键进行产点。')
disp('单击鼠标右键表示产点过程结束。')
but = 1;
while but == 1
    [xi,yi,but] = ginput(1);
    plot(xi,yi,'ro')
    n = n+1;
    xy(:,n) = [xi;yi];
end
% 使用曲线值方法使曲线更光滑。
t = 1:n;
ts = 1: 0.1: n;
xys = spline(t,xy,ts);
y=xys+sqrt(R)*randn(size(xys));
plot(xys(1,:),xys(2,:),'b-');
hold off
figure

%估计横轴。
xe=zeros(length(Q),1);p=10*eye(size(A));xx1=[];
for i=1:length(y(1,:))
[xe,p]=kalmanfun(A,C,Q,R,xe,y(1,i),p);
xx1=[xx1 xe];
end
%估计纵轴。
xe=zeros(length(Q),1);p=10*eye(size(A));xx2=[];
for i=1:length(y(2,:))
[xe,p]=kalmanfun(A,C,Q,R,xe,y(2,i),p)
xx2=[xx2 xe];
end
plot(xys(1,:),xys(2,:),'b-');hold on
plot(y(1,:),y(2,:),'*');hold on,
plot(C*xx1,C*xx2,'o');hold off

diag(cov(xys'-[C*xx1;C*xx2]'))
```

这段程序产生三个结果图，目标在二维平面中的真实轨迹如图 7-6 所示，其估计结果如图 7-7 所示，“○”给出了每一个采样点的估计结果，在程序的最后一行还计算了估计方差，横轴为 22.2743，纵轴为 20.7571。

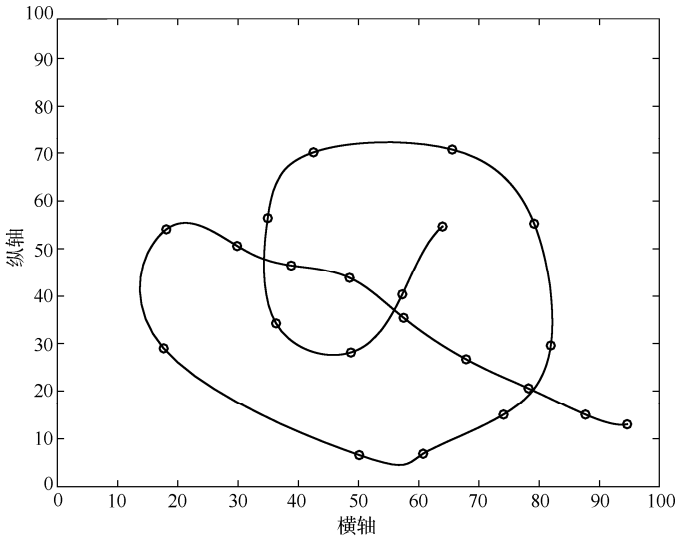


图 7-6　目标在 2 维平面内做任意运动

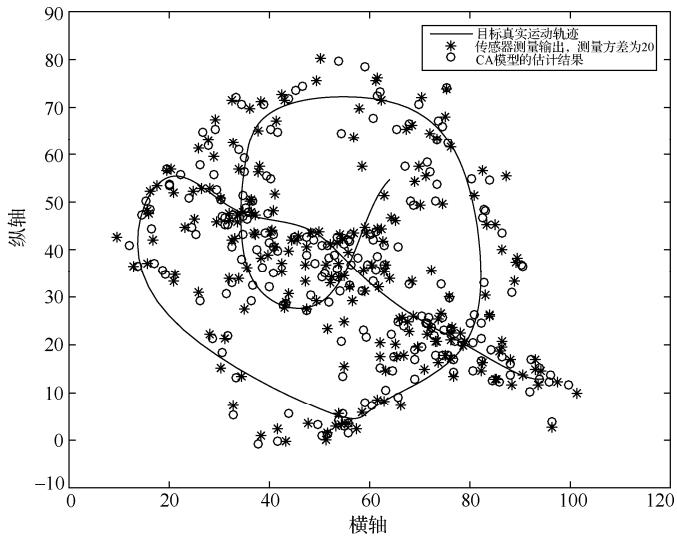


图 7-7　基于 CA 模型在 2 维平面内的跟踪结果

7.3　Singer 模型

CV 模型和 CA 模型的区别在于前者假设目标机动的“加速度”为噪声过程，而后者假设目标机动的加速度导数为噪声过程，两者的共同点为都假设该噪声过程为零均值高斯白噪声。已知白噪声是一种比较理想化的模型，实际噪声一般更符合有色噪声过程。认识到这一点后，Singer 于 1970 年提出了著名的 Singer 模型。

Singer 模型假设目标加速度为指数自相关零均值随机噪声过程，下面为了与 CV 模型及 CA 模型相区别，使用 $a(t)$ 描述目标的加速度，即 $a(t)=\ddot{x}(t)$。$a(t)$ 的时间相关函数为指数形式，即

$$R_a(\tau)=E[a(t)a(t+\tau)]=\sigma_a^2\mathrm{e}^{-\alpha|\tau|} \tag{7-17}$$

式中，σ_a^2 和 α 是待定参数，在区间 $[t,t+\tau]$ 表达了目标的机动特性，σ_a^2 为目标机动加速度方差，α 是机动时间常数的倒数，即机动频率。α 的确切值只有通过实测才能知道，在实际应用中，可以根据前人的研究结果取经验值：如目标机动形式是飞机慢速转弯，一般 α 取为 1/60s；逃避机动时 α 取为 1/20s；大气扰动 α 取为 1。

对于机动加速度方差 σ_a^2，可以根据机动目标加速度的概率密度函数来计算。那么如何得到目标机动加速度的概率密度函数的确切形式呢？答案是：利用经验进行假设。正如 CV 模型假设机动速度为常值、其加速度满足零均值高斯白噪声分布一样，这一次仍然需要假设机动目标加速度满足某种数学关系。

Singer 提出了这样的假设：①机动加速度等于极大值 a_{M} 的概率为 P_{M}，等于极小值 $-a_{\mathrm{M}}$ 的概率也为 P_{M}；②机动加速度等于 0 的概率为 P_0（即非机动概率）；③机动加速度在区间 $[-a_{\mathrm{M}},a_{\mathrm{M}}]$ 为均匀分布。由以上假设可得到如下概率密度函数：

$$P(a)=[\delta(a-a_{\mathrm{M}})+\delta(a+a_{\mathrm{M}})]P_{\mathrm{M}}+\delta(a)P_0+[1(a-a_{\mathrm{M}})+1(a+a_{\mathrm{M}})]\frac{1-P_0-2P_{\mathrm{M}}}{2a_{\mathrm{M}}} \tag{7-18}$$

式中，$1(\cdot)$ 为单位阶跃函数，$\delta(\cdot)$ 为狄拉克脉冲函数。利用概率论知识求得如式（7-18）所示的概率密度函数的方差为

$$\begin{aligned}\sigma_a^2&=\int_{-\infty}^{+\infty}a^2p(a)\mathrm{d}a\\&=(-a_{\mathrm{M}})^2P_{\mathrm{M}}+(a_{\mathrm{M}})^2P_{\mathrm{M}}+\int_{-\infty}^{+\infty}a^2\frac{1-(P_0+2P_{\mathrm{M}})}{2a_{\mathrm{M}}}\mathrm{d}a=\frac{a_{\mathrm{M}}^2}{3}(1+4P_{\mathrm{M}}-P_0)\end{aligned} \tag{7-19}$$

利用式（7-19）可以求出 Singer 模型满足上述假设的机动加速度方差 σ_a^2。注意，利用式（7-19）需要知道 a_{M}、P_{M} 及 P_0，那么这三个值又该如何确定呢？答案还是：实测或假设。也就是在获得某一类跟踪目标的机动数据后经过统计得到这些值，或者假设为某一个值，这是在仿真研究中经常用到的方法。对于一个好的模型，应该对这些值不是十分敏感，也就是说，即使这些值与未知的真值有些偏差，也希望利用“不是完全准确的”系统模型得到“还可以接受”的跟踪结果。这就又涉及另一个问题，如何使估计结果在模型存在一定误差的情况下还能保持基本的跟踪性能，已知好的估计方法可以忍受一定的模型误差，不至于使估计结果发散。可见，这是一个模型与估计方法相配合的问题，模型很不准确，对估计方法要求就高一些，反之，模型准确度高，则估计方法可以进行一定的简化以减少计算量，也会得到较好的跟踪结果。

Singer 模型最大的贡献在于其突破了 CV 模型与 CA 模型的高斯白噪声的假设，

并且对如何处理有色噪声得到相关系统参数给出了具体的解决方法，如系统模型的噪声方差、状态转移矩阵等，这给以后的研究者很大的启发，以后的研究者沿用这些处理方法根据不同的噪声假设丰富、改进了 Singer 模型，得到了性能更完善的机动目标模型。

下面来介绍 Singer 模型如何利用有色噪声得到连续模型的状态转移矩阵，需要用到的知识是 Wiener-Kolmogoron 有色噪声白化处理方法。将 $a(t)$ 的时间相关函数 $R_a(\tau)$ 进行拉普拉斯变换得到

$$R_a(s)=\frac{-2\alpha\sigma_a^2}{(s-\alpha)(s+\alpha)}=H(s)H(-s)W_c(s) \tag{7-20}$$

式中，$H(s)$ 为传递函数，$W_c(s)$ 为输入噪声向量的拉普拉斯变换，得到

$$\begin{cases}H(s)=\dfrac{1}{s+a}\\ W_c(s)=2\alpha\sigma_a^2\end{cases} \tag{7-21}$$

则经白化后的有色噪声可用输入为白噪声的一阶时间相关模型表示（该模型为一阶马尔可夫过程）：

$$\dot{a}(t)=-\alpha\cdot a(t)+w_c(t) \tag{7-22}$$

式中，$w_c(t)$ 为均值为 0、方差为 σ_w^2（其中 $\sigma_w^2=2\alpha\sigma_a^2$）的高斯白噪声，即

$$E[w_c(t)w_c^{\mathrm{T}}(\tau)]=2\alpha\sigma_a^2\delta(t-\tau) \tag{7-23}$$

仍然取系统状态向量 $\boldsymbol{x}(t)=[x(t),\dot{x}(t),\ddot{x}(t)]^{\mathrm{T}}$，得到连续模型状态方程为

$$\begin{bmatrix}\dot{x}(t)\\ \ddot{x}(t)\\ \dddot{x}(t)\end{bmatrix}=\begin{bmatrix}0&1&0\\0&0&1\\0&0&-\alpha\end{bmatrix}\begin{bmatrix}x(t)\\ \dot{x}(t)\\ \ddot{x}(t)\end{bmatrix}+\begin{bmatrix}0\\0\\1\end{bmatrix}w_c(t) \tag{7-24}$$

注意到，式（7-24）与 CA 模型的状态方程（7-10）的区别在于在状态转移矩阵中多了 $-\alpha$ 项，即加速度的导数不再等于高斯白噪声。因此，相应的离散模型也有了很大的区别，下面来推导离散模型的参数 $\boldsymbol{A}$。由式（7-24）得到连续模型中的状态转移矩阵 $\boldsymbol{F}$ 为

$$\boldsymbol{F}=\begin{bmatrix}0&1&0\\0&0&1\\0&0&-\alpha\end{bmatrix} \tag{7-25}$$

还是利用前面使用的拉普拉斯变换方法来计算离散模型中的状态转移矩阵 $\boldsymbol{A}$，得到

$$(s\boldsymbol{I}-\boldsymbol{F})^{-1}=\begin{bmatrix}s & -1 & 0\\ 0 & s & -1\\ 0 & 0 & s+\alpha\end{bmatrix}^{-1}=\begin{bmatrix}\dfrac{1}{s} & \dfrac{1}{s^2} & \dfrac{1}{s^2(s+a)}\\ 0 & \dfrac{1}{s} & \dfrac{1}{s(s+a)}\\ 0 & 0 & \dfrac{1}{(s+a)}\end{bmatrix} \tag{7-26}$$

即

$$\boldsymbol{A}=\mathrm{e}^{\boldsymbol{F}T_0}=\begin{bmatrix}1 & T_0 & \dfrac{\alpha T_0-1+\mathrm{e}^{-\alpha T_0}}{\alpha^2}\\ 0 & 1 & \dfrac{1-\mathrm{e}^{-\alpha T_0}}{\alpha}\\ 0 & 0 & \mathrm{e}^{-\alpha T_0}\end{bmatrix} \tag{7-27}$$

利用与 7.2 节相似的方法计算得到 DCM 的过程噪声方差为

$$\boldsymbol{Q}=\sigma_w^2\begin{bmatrix}q_{11} & q_{12} & q_{13}\\ q_{12} & q_{22} & q_{23}\\ q_{13} & q_{23} & q_{33}\end{bmatrix} \tag{7-28}$$

式中

$$\begin{cases}q_{11}=\dfrac{1}{2\alpha^5}\left[1-\mathrm{e}^{-2\alpha T_0}+2\alpha T_0+\dfrac{2\alpha^3T_0^3}{3}-2\alpha^2T_0^2-4\alpha T_0\mathrm{e}^{-\alpha T_0}\right]\\ q_{12}=\dfrac{1}{2\alpha^4}[\mathrm{e}^{-2\alpha T_0}+1-2\mathrm{e}^{-\alpha T_0}+2\alpha T_0\mathrm{e}^{-\alpha T_0}-2\alpha T_0+\alpha^2T_0^2]\\ q_{13}=\dfrac{1}{2\alpha^3}[1-\mathrm{e}^{-2\alpha T_0}-2\alpha T_0\mathrm{e}^{-\alpha T_0}]\\ q_{22}=\dfrac{1}{2\alpha^3}[4\mathrm{e}^{-\alpha T_0}-3-\mathrm{e}^{-2\alpha T_0}+2\alpha T_0]\\ q_{23}=\dfrac{1}{2\alpha^2}[\mathrm{e}^{-2\alpha T_0}+1-2\alpha T_0]\\ q_{33}=\dfrac{1}{2\alpha}[1-\mathrm{e}^{-2\alpha T_0}]\end{cases} \tag{7-29}$$

至此，得到了 Singer 模型状态方程的系统参数。Singer 模型和 CV 模型的区别在于式（7-22）中的α，注意到当$\alpha=0$时，Singer 模型中的连续模型状态转移矩阵 $\boldsymbol{F}$（式（7-25））、离散模型状态转移矩阵 $\boldsymbol{A}$（式（7-27））及离散模型过程噪声方差矩阵 $\boldsymbol{Q}$（式（7-28）和式（7-29））退化成 CA 模型中的式（7-11）、式（7-13）及式（7-14），由此可见，CA 模型只是 Singer 模型的一个特例。

正如前所说，Singer 模型算法的提出及其解决有色噪声的方法为以后的各种机动模型奠定了基础，因此在跟踪模型研究中具有十分重要的意义。但不难看出，Singer 模型假设目标机动加速度在区间[$-a_M$，a_M]为均匀分布，从而得到加速度σ_a^2，这种方法其实并不能很好地描述任何实际机动目标的运动，假设同样具有很大的先验知识特征。

根据式（7-27）和式（7-29），Singer 模型的 MATLAB 程序如下，其中 T 是采样周期，qq 和前面一样，是机动目标运动过程中的噪声方差，a 为机动频率，可以根据经验进行设置，输出为系统参数 A 和过程噪声方差矩阵。

```
function [A,Q]=Singermodel(T,qq,a)
A=[1 T (a*T-1+exp(-a*T))/a/a;0 1 (1-exp(-a*T))/a;0 0 exp(-a*T)];
q11=(1-exp(-2*a*T)+2*a*T+2*(a^3)*(T^3)/3-2*a^2*T^2-4*a*T*exp(-a*T))/
(a^4);
q12=(exp(-2*a*T)+1-2*exp(-a*T)+2*a*T*exp(-a*T)-2*a*T+a^2*T^2)/(a^3);
q13=(1-exp(-2*a*T)-2*a*T*exp(-a*T))/(a^2);
q22=(4*exp(-a*T)-3-exp(-2*a*T)+2*a*T)/(a^2);
q23=(exp(-2*a*T)+1-2*exp(-a*T))/a;
q33=(1-exp(-2*a*T));
Q=qq*[q11 q12 q13;q12 q22 q23;q13 q23 q33];
```

7.4　当前统计模型

当前统计模型是由周宏仁于 1983 年提出来的，其基本思想在于，当目标正以某一加速度机动时，下一时刻的加速度取值是有限的，且只能在“当前”加速度的邻域内。从本质上讲，该模型是 Singer 模型的改进版本，改进的方面有如下两项：①加速度的均值为非零值$\bar{a}$，并取为当前加速度的预测值，即$\bar{a}=\hat{\dot{x}}(k+1|k)$，且机动加速度仍符合一阶时间相关过程；②假设加速度的统计特性满足修正的瑞利分布，而不再是 Singer 模型假设的均匀分布。这两项改进的优势在于，均值非零更符合实际机动目标的运动特性，瑞利分布的方差由均值决定，在估计目标状态的同时得到了实时变化的机动加速度均值，从而实时地修正加速度的分布，并通过方差反馈到下一时刻的滤波增益之中，实现了闭环自适应跟踪。

根据上述第一项假设，可以知道加速度的模型为

$$\ddot{x}(t)=\bar{a}(t)+a(t) \tag{7-30}$$

$$\dot{a}(t)=-\alpha\cdot a(t)+w_c(t) \tag{7-31}$$

式中，$\bar{a}(t)$为机动加速度“当前”均值，在每一采样周期内为常数。若令$a_1(t)=\bar{a}(t)+a(t)$，代入式（7-30）和式（7-31），得到

$$\begin{cases}\ddot{x}(t)=a_1(t)\\ \dot{a}_1(t)=-\alpha a_1(t)+\alpha\bar{a}(t)+w_c(t)\end{cases} \tag{7-32}$$

进一步得到如下连续系统模型：

$$\begin{bmatrix}\dot{x}(t)\\ \ddot{x}(t)\\ \dddot{x}(t)\end{bmatrix}=\begin{bmatrix}0 & 1 & 0\\ 0 & 0 & 1\\ 0 & 0 & -\alpha\end{bmatrix}\begin{bmatrix}x(t)\\ \dot{x}(t)\\ \ddot{x}(t)\end{bmatrix}+\begin{bmatrix}0\\ 0\\ \alpha\end{bmatrix}\bar{a}(t)+\begin{bmatrix}0\\ 0\\ 1\end{bmatrix}w_c(t) \tag{7-33}$$

即

$$\dot{\boldsymbol{x}}(t)=\boldsymbol{F}\boldsymbol{x}(t)+\boldsymbol{E}\bar{a}(t)+\boldsymbol{G}w_c(t) \tag{7-34}$$

式中

$$\boldsymbol{F}=\begin{bmatrix}0 & 1 & 0\\ 0 & 0 & 1\\ 0 & 0 & -\alpha\end{bmatrix},\quad \boldsymbol{E}=\begin{bmatrix}0\\ 0\\ \alpha\end{bmatrix},\quad \boldsymbol{G}=\begin{bmatrix}0\\ 0\\ 1\end{bmatrix} \tag{7-35}$$

比较式（7-33）与式（7-24），发现多了输入项 $\bar{a}(t)$，因此，有必要先讨论一下式（7-34）中输入矩阵项的离散化方法。利用 6.1 节解状态方程的方法得到

$$\boldsymbol{x}(t)=\mathrm{e}^{\boldsymbol{F}(t-t_0)}\boldsymbol{x}(t_0)+\int_{t_0}^{t}\mathrm{e}^{\boldsymbol{F}(t-\lambda)}\boldsymbol{E}\bar{a}(\lambda)\mathrm{d}\lambda+\int_{t_0}^{t}\mathrm{e}^{\boldsymbol{F}(t-\lambda)}\boldsymbol{G}w_c(\lambda)\mathrm{d}\lambda \tag{7-36}$$

又已知 $\bar{a}(t)$ 在每一采样周期内为常数 $\bar{a}(t_0)$（以下将其简写为 $\bar{a}$），得到

$$\boldsymbol{x}(t)=\mathrm{e}^{\boldsymbol{F}(t-t_0)}\boldsymbol{x}(t_0)+\int_{t_0}^{t}\mathrm{e}^{\boldsymbol{F}(t-\lambda)}\boldsymbol{E}\mathrm{d}\lambda\cdot\bar{a}+\int_{t_0}^{t}\mathrm{e}^{\boldsymbol{F}(t-\lambda)}\boldsymbol{G}w_c(\lambda)\mathrm{d}\lambda \tag{7-37}$$

即同样取 $T_0=t-t_0$，即在一个采样周期内进行积分，得到如下离散形式的动力学模型：

$$\boldsymbol{x}(k+1)=\boldsymbol{A}\boldsymbol{x}(k)+\boldsymbol{U}\bar{a}+w(k) \tag{7-38}$$

式中，$\boldsymbol{A}$ 与 $w(k)$ 的方差如式（7-27）和式（7-38）所示，进一步利用式（7-35）得到的输入项转移矩阵如下：

$$\begin{aligned}\boldsymbol{U}&=\int_0^{T_0}\mathrm{e}^{\boldsymbol{F}(T_0-\lambda)}E\mathrm{d}\lambda\\ &=\int_0^{T_0}\begin{bmatrix}1 & T_0-\lambda & \dfrac{\alpha(T_0-\lambda)-1+\mathrm{e}^{-\alpha(T_0-\lambda)}}{\alpha^2}\\ 0 & 1 & \dfrac{1-\mathrm{e}^{-\alpha(T_0-\lambda)}}{\alpha}\\ 0 & 0 & \mathrm{e}^{-\alpha(T_0-\lambda)}\end{bmatrix}\mathrm{d}\lambda\cdot\begin{bmatrix}0\\ 0\\ -\alpha\end{bmatrix}\\ &=\begin{bmatrix}\dfrac{1}{\alpha}\left(-T_0+\dfrac{\alpha T_0^2}{2}+\dfrac{1-\mathrm{e}^{-\alpha T_0}}{\alpha}\right)\\ T_0-\dfrac{1-\mathrm{e}^{-\alpha T_0}}{\alpha}\\ 1-\mathrm{e}^{-\alpha T_0}\end{bmatrix}\end{aligned} \tag{7-39}$$

观察当前统计模型的参数，如 $\boldsymbol{A}$、$\boldsymbol{U}$ 及方差 $\boldsymbol{Q}$ 可知，除了系统的采样周期 T_0 外，还需要机动频率 α 及目标机动加速度方差 σ_a^2 才可以求出。当前统计模型中的机动频率 α 和 Singer 模型的求法一致，需要进行实测或根据先验知识进行假设。至于 σ_a^2，前面提到，当前统计模型将其做了改进，不再像 Singer 模型那样假设加速度噪声满足某一区间的均匀分布，而是假设其满足瑞利分布，接下来讨论如何利用修正瑞利分布求得加速度方差 σ_a^2。

当目标“当前”加速度为正时，修正瑞利分布的概率密度函数为

$$P(a)=\begin{cases}\dfrac{(a_{\mathrm{M}}-a)}{\mu^2}\exp\left\{-\dfrac{(a_{\mathrm{M}}-a)^2}{2\mu^2}\right\}, & 0<a<a_{\mathrm{M}}\\ 0, & a\geqslant a_{\mathrm{M}}\end{cases} \tag{7-40}$$

式中，$a_{\mathrm{M}}>0$ 为已知的目标加速度最大值，a 为目标机动加速度，$\mu>0$ 为一常数。a 的均值和方差分别为

$$E[a]=a_{\mathrm{M}}-\sqrt{\frac{\pi}{2}}\mu \tag{7-41}$$

$$\sigma_a^2=\frac{4-\pi}{2}\mu^2 \tag{7-42}$$

当目标“当前”加速度为负时，概率密度函数为

$$P(a)=\begin{cases}\dfrac{(a-a_{-\mathrm{M}})}{\mu^2}\exp\left\{-\dfrac{(a-a_{-\mathrm{M}})^2}{2\mu^2}\right\}, & 0>a>a_{-\mathrm{M}}\\ 0, & a\leqslant a_{-\mathrm{M}}\end{cases} \tag{7-43}$$

式中，$a_{-\mathrm{M}}>0$ 为已知的目标加速度最小值。a 的均值和方差分别为

$$E[a]=a_{-\mathrm{M}}+\sqrt{\frac{\pi}{2}}\mu \tag{7-44}$$

$$\sigma_a^2=\frac{4-\pi}{2}\mu^2 \tag{7-45}$$

当目标“当前”加速度为零时，概率密度函数为

$$p(a)=\delta(a) \tag{7-46}$$

式中，$\delta(\cdot)$ 为狄拉克函数。

当前统计模型假设 a 的均值 $\bar{a}=E[a]$ 在每一采样周期内为常数，且取为当前加速度的预测值，即 $\bar{a}=\hat{x}(k+1|k)$，也就是说在实际滤波过程中 $\bar{a}$ 为已知的。利用式（7-41）和式（7-42）可以消去中间变量 μ，得到修正瑞利分布关于均值与方差的关系。

当目标“当前”加速度为正时：

$$\sigma_a^2 = \frac{4-\pi}{\pi}[a_{\mathrm{M}} - \bar{a}]^2 \tag{7-47}$$

当目标“当前”加速度为负时：

$$\sigma_a^2 = \frac{4-\pi}{\pi}[\bar{a} - a_{-\mathrm{M}}]^2 \tag{7-48}$$

当目标“当前”加速度为零时，σ_a^2 可取任意小的正数。

应用式（7-47）和式（7-48）可以利用每一步的加速度估计值得到修正瑞利分布的加速度噪声方差，从而形成了“估计—模型”参数的闭环结构，也就是说，当前统计模型利用“当前”的估计值调整系统参数并期望该参数能够更准确地反映目标机动的当前特性，进而提高估计性能。进一步良好的估计结果又得到了更为准确的系统参数，从而形成了良好的闭环递推过程。但必须指出，当前统计模型利用的是“修正瑞利分布”，其特点是可以由噪声均值计算方差，而方差正是求取系统参数不可缺少的。但加速度噪声是否一定符合修正瑞利分布呢？就像 Singer 模型假设其满足均匀分布，但认为这样的假设不会完全符合实际情况一样，修正瑞利分布也不可能包揽所有的机动目标噪声情况，之所以选择瑞利分布，很大原因在于可以根据瑞利分布噪声过程的均值求得其方差。

同 Singer 模型一样，在机动模型研究中，当前统计模型也具有重要的意义，其贡献在于：①将机动模型与滤波过程形成了一个闭环结构，系统模型可以根据滤波结果进行实时调整，与 Singer 模型的开环结构相比，具有很大的进步；②给出了闭环结构中关键部分——由 $\bar{a}$ 求取 σ_a^2 的一种解法，即将噪声方差假设为修正瑞利分布，这为研究者提供了一个开放的平台，即如果修正瑞利分布不能满足实际应用的假设，应该利用何种方法求出更为合理的 σ_a^2？

回到当前统计模型，下面给出了当前统计模型的 MATLAB 程序，模型的输入为采样周期 T，当前的状态估计值 $\hat{\hat{x}}(k+1|k)$、机动频率 α、参数 a_{M}，分别用 xa、a、xamax 表示。从程序中可以看出，Singer 模型不再需要设置目标运动过程中噪声方差 σ_a^2，而是利用当前的状态估计值以及参数 a_{M} 利用瑞利分布计算得到。

```
function [A1,A,Q,U,qa]=Starmodel(T,xa,a,xamax)
if xa>0
   qa=(xamax-xa)^2*(4-pi)/pi
else
   qa=(xamax+xa)^2*(4-pi)/pi
end

A=[1 T (a*T-1+exp(-a*T))/a/a;0 1 (1-exp(-a*T))/a;0 0 exp(-a*T)];
q11=(1-exp(-2*a*T)+2*a*T+2*(a^3)*(T^3)/3-2*a^2*T^2-4*a*T*exp(-a*T))/
(a^4);
```

```
q12=(exp(-2*a*T)+1-2*exp(-a*T)+2*a*T*exp(-a*T)-2*a*T+a^2*T^2)/(a^3);
q13=(1-exp(-2*a*T)-2*a*T*exp(-a*T))/(a^2);
q22=(4*exp(-a*T)-3-exp(-2*a*T)+2*a*T)/(a^2);
q23=(exp(-2*a*T)+1-2*exp(-a*T))/a;
q33=(1-exp(-2*a*T));
Q=qa*[q11 q12 q13;q12 q22 q23;q13 q23 q33];
A1=[1 T T^2/2;0 1 T;0 0 1];
U=[(-T+a*T^2/2+(1-exp(-a*T))/a)/a;T-(1-exp(-a*T))/a;1-exp(-a*T)];
```

下面看一下仿真练习。

在 2 维空间中考虑状态 $\boldsymbol{x}(k)=[x(k)\quad \dot{x}(k)\quad \ddot{x}(k)\quad y(k)\quad \dot{y}(k)\quad \ddot{y}(k)]^{\mathrm{T}}$，其中 $x(k)$、$y(k)$ 分别是目标在横、纵轴的位移，$\dot{x}(k)$、$\dot{y}(k)$ 分别是目标在横、纵轴移动的速度，单位是 m/s，$\ddot{x}(k)$、$\ddot{y}(k)$ 则分别是目标在横、纵轴移动的加速度，单位是 m/s^2。设状态的初始值为 $\boldsymbol{x}_0=[2000\quad 0\quad 0\quad 10000\quad -15\quad 0]^{\mathrm{T}}$，采样周期为 10s，即 T=10s。设目标在第 40 到第 60 的采样周期期间做了一个慢拐弯，横轴的加速度为 0.075m/s^2，纵轴的加速度也为 0.075m/s^2。在第 61 到第 66 采样周期，目标做了一个急速拐弯，横轴的加速度为 0.3m/s^2，纵轴的加速度为–0.3m/s^2，在其余时间目标做匀速直线运动。首先利用下面这个程序跟踪目标运动的轨迹，给出了横轴坐标的目标运动数据如图 7-8 所示，目标在 2 维空间中的运动轨迹如图 7-9 所示。然后把轨迹数据，包括目标在横轴和纵轴的位移数据以及相应的采样时间保存起来。

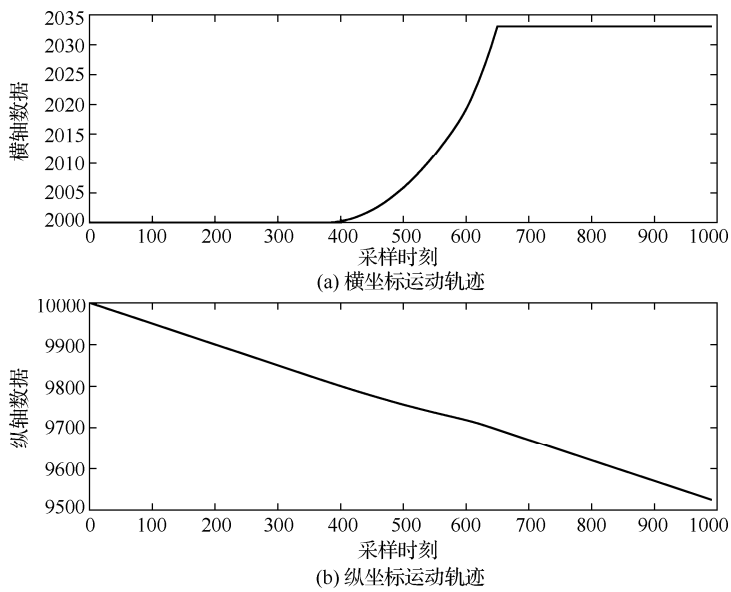


图 7-8　目标运动的实际轨迹

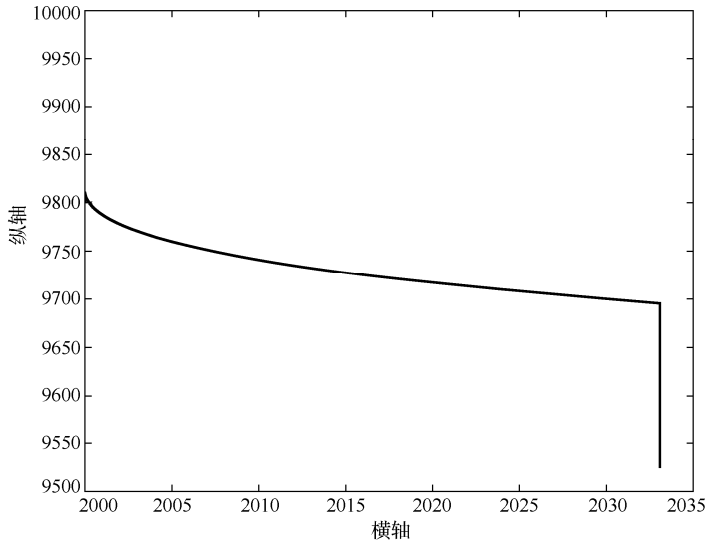


图 7-9　目标在 2 维空间的运动轨迹

```
clc
clear
number=100;
x=zeros(1,100);y=zeros(1,number);
ax1=0.075;ay1=0.075;
ax2=0.3;ay2=-0.3;
T=10;
t=0:T:T*(number-1);
for i=1:number
    if i==1
        x(i)=2000;
           y(i)=10000;
    elseif i<=39
        x(i)=x(i-1);
         y(i)=y(i-1)-5;
    elseif i<=60;
        x(i)=2*x(i-1)-x(i-2)+ax1;
         y(i)=2*y(i-1)-y(i-2)+ay1;
    elseif i<=66;
        x(i)=2*x(i-1)-x(i-2)+ax2;
         y(i)=2*y(i-1)-y(i-2)+ay2;
    else
        x(i)=x(i-1);
         y(i)=y(i-1)-5;
    end
```

```
end
ts=t;
xys=[x;y];
subplot(2,1,1);plot(t,x)
subplot(2,1,2);plot(t,y)
figure
plot(x,y)

%将数据 ts、xys 写入 Hotarget 数据文件。
save Hotarget ts xys
```

下面使用当前统计模型来跟踪上述目标的运动。

```
clc
clear
summ=0;N=10;
for n=1:N

%读入 Hotarget 数据文件。
load Hotarget
T=10;
R=10000 ts xys;

y=xys+sqrt(R)*randn(size(xys));
qqa=[];
%选择模型参数。
 a=1/20;
 xamax=30;
 C=[1 0 0];
%估计横轴。
xe=zeros(3,1);p=10*eye(3);xx1=[];
for i=1:length(y(1,:))
xa=xe(3);
[A1,A,Q,U,qa]=Starmodel(T,xa,a,xamax);
[xe,p]=kalmanadfun(A1,A,U,C,Q,R,xe,y(1,i),p);
xa=xe(3);
xx1=[xx1 xe];
qqa=[qqa qa];
end
%估计纵轴。
xe=zeros(3,1);p=10*eye(3);xx2=[];
for i=1:length(y(2,:))
```

```
xa=xe(3);
[A1,A,Q,U,qa]=Starmodel(T,xa,a,xamax);
[xe,p]=kalmanadfun(A1,A,U,C,Q,R,xe,y(2,i),p);
xx2=[xx2 xe];
end

covv=diag(cov(xys'-[C*xx1;C*xx2]'))

summ=summ+covv;
end
summ/N
 plot(xys(1,:),xys(2,:),'--');hold on
plot(C*xx1,C*xx2,'r*');hold off
figure
subplot(2,1,1),plot(ts,xys(1,:),ts,C*xx1,'-.')
subplot(2,1,2),plot(ts,xys(2,:),ts,C*xx2,'-.')
figure
subplot(2,1,1),plot(ts,xys(1,:)-C*xx1)
subplot(2,1,2),plot(ts,xys(2,:)-C*xx2)
```

先来说明上述程序的组成，N 表示要多次执行程序，基于 Monte Carlo 的思想，这样做在程序最后计算方差的时候会更准确。程序调用了产生 2 维空间轨迹的运动数据 Hotarget，并设置采样周期为 10s，测量方差为 10000，进而给出了传感器采样的测量数据 y，该数据和目标真实轨迹的差别如图 7-10 所示，其中目标的真实估计为虚线所示，“*”点是测量数据点。程序接下来选择模型参数，包括 a、xamax，再分别跟踪横轴和纵轴的状态，最后给出估计方差横、纵轴分别为 8993、17226，目标的真实轨迹和估计轨迹如图 7-11 所示，其中实线是目标的真实轨迹，虚线是根据传感器测量数据得到的估计轨迹。这两者还是有很大差别的，由于测量数据的方差很大（10000），估计结果的方差也就相对比较大。图 7-11 中估计轨迹和目标的真实轨迹之差如图 7-12 所示。

下面来看另外一组数据，使用第 6 章介绍的在 2 维平面内能够生成任意移动轨迹的程序，来产生 mytarget 数据文件。先调用数据文件，将采样周期设为 0.1s，并对测量方差进行设置，设为 25。使用下面的程序：

```
load mytarget
T=0.1;
R=25;
```

来代替原来程序中的以下三行：

```
load Hotarget
T=10;
R=10000;
```

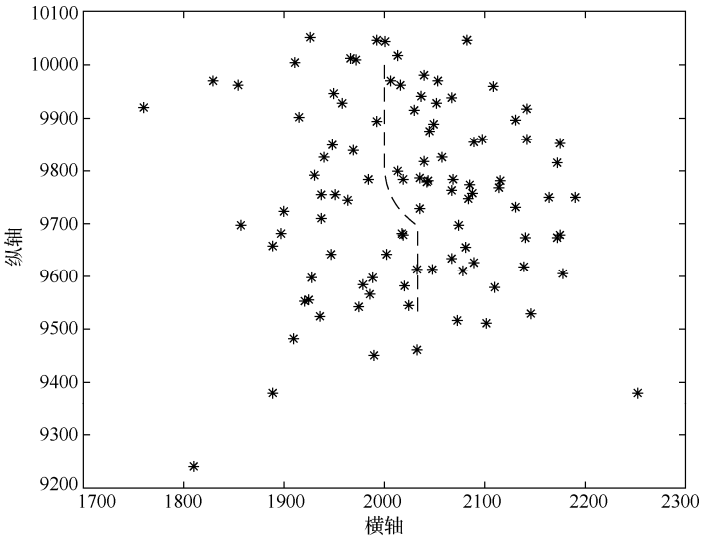


图 7-10　目标在 2 维空间的真实轨迹和传感器测量数据

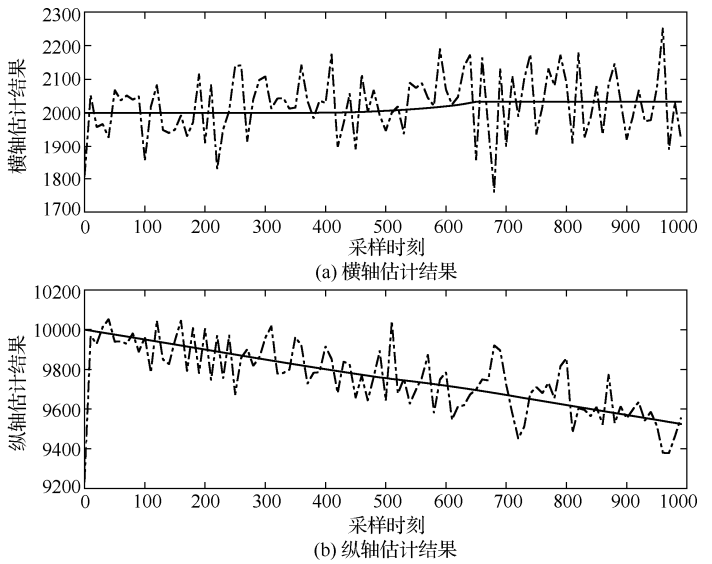


图 7-11　跟踪轨迹的结果

实线表示真实轨迹，虚线表示利用传感器测量数据估计的结果

其余程序不变，得到的结果图分别如图 7-13～图 7-15 所示，横、纵轴的估计方差分别为 7.3985、5.4062。发现使用多次仿真（如 N 取为 10）时，每次计算出的跟踪方差结果不相同，但相差不大。分析认为，在每一次叠加测量噪声的时候，产生的随机数并不相同，因此每一次的仿真计算结果也略有不同，读者可以试着增加仿真次数来使这个数据更为稳定。

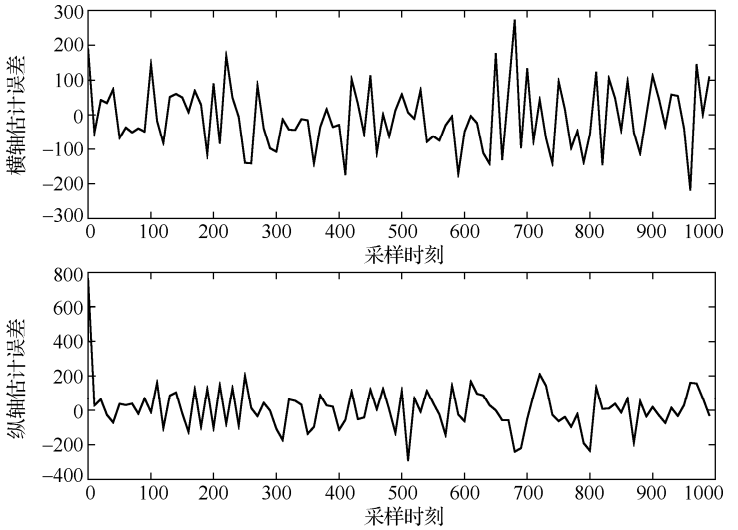


图 7-12　估计结果与真实估计之差

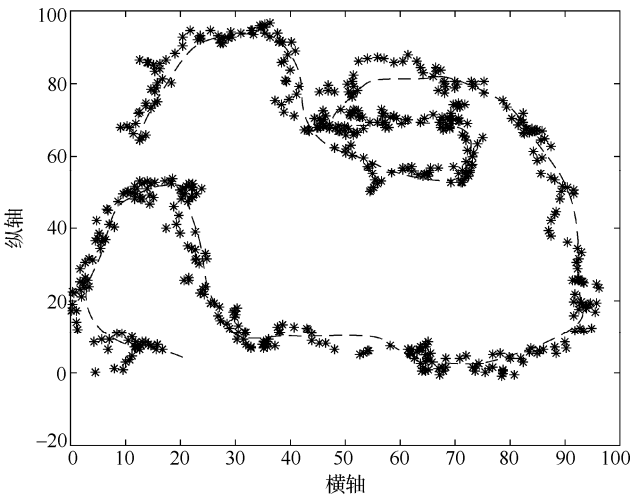
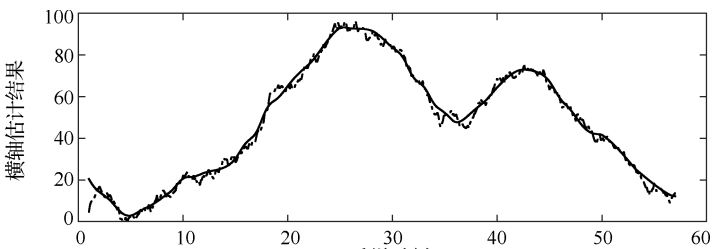
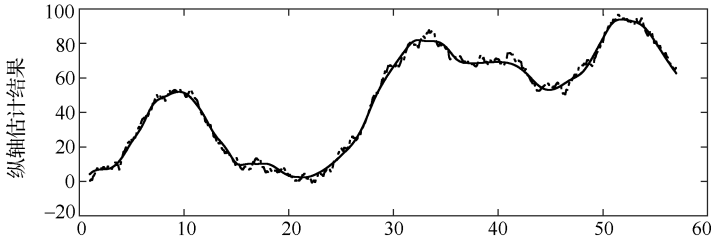


图 7-13　数据 mytarget 中，目标在 2 维空间的真实轨迹和传感器测量数据



(a) 横轴估计结果



(b) 纵轴估计结果

图 7-14　数据 mytarget 中，跟踪轨迹的结果

实线表示真实轨迹，虚线表示利用传感器测量数据估计的结果

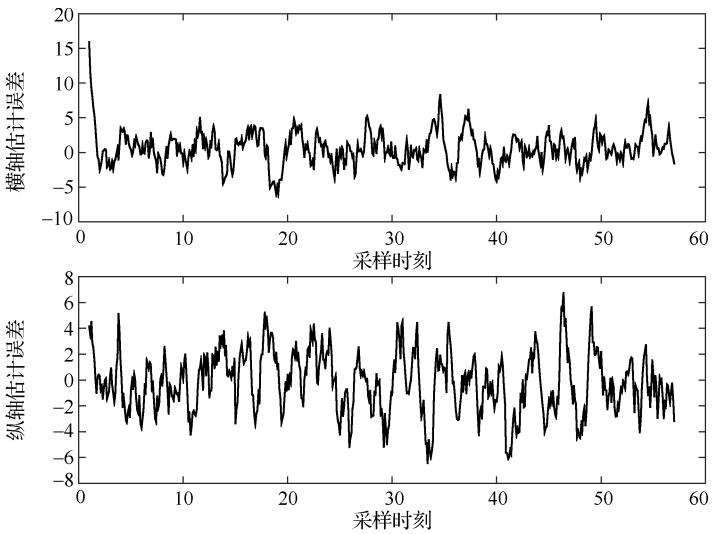


图 7-15　数据 mytarget 中，估计结果与真实估计之差

7.5　Jerk 模型

算法跟踪复杂机动目标的性能不佳，其原因之一在于状态向量的阶数不足，因此 Jerk 模型在加速度模型的基础上又增加了一维——加速度导数，从而得到对加速度估计更加准确。

类似于 Singer 模型，目标加速度导数（用 $j(t)$ 表示）的指数自相关函数为

$$R_j(\tau) = E[j(t)\,j(t+\tau)] = \sigma_j^2 \mathrm{e}^{-\alpha|\tau|} \tag{7-49}$$

即加速度导数满足如下微分方程：

$$\dot{j}(t) = -\alpha \cdot j(t) + w_c(t) \tag{7-50}$$

式中，$w_c(t)$ 为均值为 0、方差为 σ_w^2（其中 $\sigma_w^2 = 2\alpha\sigma_j^2$）的高斯白噪声。

令系统状态向量为 $\boldsymbol{x}(t) = [x(t), \dot{x}(t), \ddot{x}(t), \dddot{x}(t)]^{\mathrm{T}}$，得到连续模型状态方程为

$$\begin{bmatrix} \dot{x}(t) \\ \ddot{x}(t) \\ \dddot{x}(t) \\ \ddddot{x}(t) \end{bmatrix} = \begin{bmatrix} 0 & 1 & 0 & 0 \\ 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 1 \\ 0 & 0 & 0 & -\alpha \end{bmatrix} \begin{bmatrix} x(t) \\ \dot{x}(t) \\ \ddot{x}(t) \\ \dddot{x}(t) \end{bmatrix} + \begin{bmatrix} 0 \\ 0 \\ 0 \\ 1 \end{bmatrix} w_c(t) \tag{7-51}$$

利用前面 Singer 模型使用的方法，可以得到 Jerk 模型的系统参数：

$$\boldsymbol{A} = \begin{bmatrix} 1 & T_0 & \dfrac{T_0^2}{2} & p_1 \\ 0 & 1 & T_0 & q_1 \\ 0 & 0 & 1 & r_1 \\ 0 & 0 & 0 & s_1 \end{bmatrix} \tag{7-52}$$

式中

$$\begin{cases} p_1 = \dfrac{2 - 2\alpha T_0 + \alpha^2 T_0^2 - 2\mathrm{e}^{-\alpha T_0}}{2\alpha^3} \\ q_1 = \dfrac{\mathrm{e}^{-\alpha T_0} - 1 + 2\alpha T_0}{\alpha^2} \\ r_1 = \dfrac{1 - \mathrm{e}^{-\alpha T_0}}{\alpha} \\ s_1 = \mathrm{e}^{-\alpha T_0} \end{cases} \tag{7-53}$$

过程噪声方差矩阵为

$$\boldsymbol{Q} = 2\alpha\sigma_j^2 \begin{bmatrix} q_{11} & q_{12} & q_{13} & q_{14} \\ q_{21} & q_{22} & q_{23} & q_{24} \\ q_{31} & q_{32} & q_{33} & q_{34} \\ q_{41} & q_{42} & q_{43} & q_{44} \end{bmatrix} \tag{7-54}$$

式中，对称矩阵 $\boldsymbol{Q}$ 的具体表达式为

$$
\begin{cases}
q_{11}=\dfrac{1}{2\alpha^7}\left[\dfrac{\alpha^5T_0^5}{10}-\dfrac{\alpha^4T_0^4}{2}+\dfrac{4\alpha^3T_0^3}{3}-2\alpha^2T_0^2+2\alpha T_0-3+4\mathrm{e}^{-\alpha T_0}+2\alpha^2T_0^2\mathrm{e}^{-\alpha T_0}-\mathrm{e}^{-2\alpha T_0}\right] \\
q_{12}=\dfrac{1}{2\alpha^4}\left[1-2\alpha T_0+2\alpha^2T_0^2-\alpha^3T_0^3+\dfrac{\alpha^4T_0^4}{4}+\mathrm{e}^{-2\alpha T_0}+2\alpha T_0-2\mathrm{e}^{-\alpha T_0}-\alpha^2T_0^2\mathrm{e}^{-\alpha T_0}\right] \\
q_{13}=\dfrac{1}{2\alpha^3}\left[2\alpha T_0-\alpha^2T_0^2-\dfrac{\alpha^3T_0^3}{3}-3-2\mathrm{e}^{-2\alpha T_0}+4\mathrm{e}^{-\alpha T_0}+\alpha^2T_0^2\mathrm{e}^{-\alpha T_0}\right] \\
q_{14}=\dfrac{1}{2\alpha^3}[1+\mathrm{e}^{-2\alpha T_0}-2\mathrm{e}^{-\alpha T_0}-2\alpha^2T_0^2\mathrm{e}^{-\alpha T_0}] \\
q_{22}=\dfrac{1}{2\alpha^3}\left[1-\mathrm{e}^{-2\alpha T_0}+\dfrac{4\alpha^3T_0^3}{3}+2\alpha T_0-2\alpha^2T_0^2-4\alpha T_0\mathrm{e}^{-\alpha T_0}\right] \\
q_{23}=\dfrac{1}{2\alpha^2}[1+\alpha^2T_0^2-2\alpha T_0+2\alpha T_0\mathrm{e}^{-\alpha T_0}+\mathrm{e}^{-2\alpha T_0}-2\mathrm{e}^{-\alpha T_0}] \\
q_{24}=\dfrac{1}{2\alpha^2}[1-\mathrm{e}^{-2\alpha T_0}-2\alpha T_0\mathrm{e}^{-2\alpha T_0}] \\
q_{33}=\dfrac{1}{2\alpha^3}[4\mathrm{e}^{-\alpha T_0}-\mathrm{e}^{-2\alpha T_0}+2\alpha T_0-3] \\
q_{34}=\dfrac{1}{2\alpha^2}[\mathrm{e}^{-2\alpha T_0}+1-2\alpha T_0] \\
q_{44}=\dfrac{1}{2\alpha}[1-\mathrm{e}^{-2\alpha T_0}]
\end{cases}
\tag{7-55}
$$

Jerk 模型和前面的 Singer 模型非常相似，在这里就不给出 MATLAB 程序了，读者可以根据自己的需要，在 Singer 模型的基础上进行修改设计出 Jerk 模型的程序。

7.6　交互式多模型算法

无论是 Singer 模型、当前统计模型还是 Jerk 模型都会碰到一个问题，就是使假设目标机动特性满足某一特定规律，但是，因为实际机动目标运动过程中的噪声特性会有所变化，在某一时间段内跟踪效果较好的模型在下一个时间段内可能就不再保持良好的性能，于是人们提出了同时使用多个假设，并在跟踪中不断地判断当前目标机动特性来改变所用模型的想法。交互式多模型算法（Interacting Multiple Model Algorithm，IMM）[22]较好地贯彻了这种思想，是一种具有实用水平的跟踪模型，也是在跟踪领域中应用最为广泛的机动模型。

IMM 包括多个基本模型，根据滤波器结果对每一个模型与当前机动目标的一致性进行估算，利用估算结果对各个模型产生的滤波结果进行加权计算，从而得到比单一模型性能更优的跟踪效果。

下面为具体实现过程。

7.6.1　初始量的假设

IMM 假设模型概率切换是在 Markov 链下进行的，设系统的先验模型数为 N，即 N 个明显的集合为 $M(k)=\{m_i(k)\}(i=1,\cdots,N)$。从 k 时刻模型 i 切换跳转到 $k+1$ 时刻模型 j（$j=1,\cdots,N$）的转移概率为 P_{ij}，即 $P_{ij}=\boldsymbol{P}\{m(k+1)=m_j \mid m(k)=m_i\}$，模型转移概率矩阵为

$$\boldsymbol{P}=\begin{bmatrix} P_{11} & P_{12} & \cdots & P_{1N} \\ P_{21} & P_{22} & \cdots & P_{2N} \\ \vdots & \vdots & & \vdots \\ P_{N1} & P_{N2} & \cdots & P_{NN} \end{bmatrix} \tag{7-56}$$

式中，每一行表示 k 时刻的当前模型，每一列表示 $k+1$ 时刻的模型。需要注意的是，$\boldsymbol{P}$ 是给定的矩阵，也是根据先验知识给定的。另外，$k=0$ 步时还需要给出当前模型 i 的概率初值为 $\mu_i(0)$（$i=1,\cdots,N$）。

7.6.2　状态估计的交互式作用

需要预测当前模型 i 变化到模型 j 的输入交互概率，首先计算联合概率矩阵：

$$\boldsymbol{P}_{\mu}(k+1\mid k)=\begin{bmatrix} P_{11}\mu_1(k) & P_{12}\mu_1(k) & \cdots & P_{1N}\mu_1(k) \\ P_{21}\mu_2(k) & P_{22}\mu_2(k) & \cdots & P_{2N}\mu_2(k) \\ \vdots & \vdots & & \vdots \\ P_{N1}\mu_N(k) & P_{N2}\mu_N(k) & \cdots & P_{NN}\mu_N(k) \end{bmatrix} \tag{7-57}$$

式中，$P_{ij}\mu_i(k)$ 表示模型 i 转移到模型 j 的预测联合概率，将式（7-57）中的每一列相加，即可得到在 $k+1$ 时刻的模型 j 的预测概率：

$$\mu_j(k+1\mid k)=\boldsymbol{P}\{m_j(k+1)\mid \boldsymbol{Z}^k\}=\sum_{i=1}^{N}P_{ij}\mu_i(k) \tag{7-58}$$

则输入交互概率为

$$\mu_{j/i}(k)=\boldsymbol{P}\{m_i(k)\mid m_j(k+1),\boldsymbol{Z}^k\}=\frac{P_{ij}\mu_i(k)}{\mu_j(k+1\mid k)} \tag{7-59}$$

输入交互概率意味着如果在未来 $k+1$ 时刻为模型 j，则在当前时刻 k 为模型 i 的概率，有点像后验概率的意思。用数值举例计算如下。

假设模型为 3 个，其先验概率相同，即为 $\boldsymbol{\mu}(0)=[1/3,\ 1/3,\ 1/3]$，Markov 模型概率转移矩阵为

$$\boldsymbol{P}=\begin{bmatrix}0.8 & 0.15 & 0.05\\ 0.3 & 0.4 & 0.3\\ 0.05 & 0.15 & 0.8\end{bmatrix} \tag{7-60}$$

则得到

$$\boldsymbol{P}_{\mu}(k+1\,|\,k)=\begin{bmatrix}\dfrac{0.8}{3} & \dfrac{0.15}{3} & \dfrac{0.05}{3}\\ \dfrac{0.3}{3} & \dfrac{0.4}{3} & \dfrac{0.3}{3}\\ \dfrac{0.05}{3} & \dfrac{0.15}{3} & \dfrac{0.8}{3}\end{bmatrix}$$

$$\begin{cases}\mu_1(1\,|\,0)=(0.8+0.3+0.05)/3=1.15/3\\ \mu_2(1\,|\,0)=(0.15+0.4+0.15)/3=0.7/3\\ \mu_3(1\,|\,0)=(0.05+0.3+0.8)/3=1.15/3\end{cases} \tag{7-61}$$

可以看到，得到的式（7-61）显示在 $k=1$ 时刻各模型的概率不再相同。在 $k+1$ 时刻为模型 j，则在当前时刻 k 为模型 i 的输入交互概率矩阵为

$$\boldsymbol{P}_{\mu_{j/i}}(k)=\begin{bmatrix}\dfrac{0.8}{1.15} & \dfrac{0.15}{0.7} & \dfrac{0.05}{1.15}\\ \dfrac{0.3}{1.15} & \dfrac{0.4}{0.7} & \dfrac{0.3}{1.15}\\ \dfrac{0.05}{1.15} & \dfrac{0.15}{0.7} & \dfrac{0.8}{1.15}\end{bmatrix} \tag{7-62}$$

根据输入交互概率对当前状态和估计方差进行交互计算，方法如下：

$$\hat{\boldsymbol{x}}_j^0(k\,|\,k)=E[\boldsymbol{x}(k)\,|\,m_j(k+1),\boldsymbol{Z}^k]=\sum_{i=1}^{N}\hat{\boldsymbol{x}}_i(k\,|\,k)\mu_{j/i}(k) \tag{7-63}$$

$$\begin{aligned}\boldsymbol{P}_j^0(k\,|\,k)&=\mathrm{cov}[\hat{\boldsymbol{x}}_j^0(k\,|\,k)\,|\,m_j(k+1),\boldsymbol{Z}^k]\\ &=\sum_{i=1}^{N}\left\{\boldsymbol{P}_i(k\,|\,k)+[\hat{\boldsymbol{x}}_j^0(k\,|\,k)-\hat{\boldsymbol{x}}_i(k\,|\,k)][\hat{\boldsymbol{x}}_j^0(k\,|\,k)-\hat{\boldsymbol{x}}_i(k\,|\,k)]^{\mathrm{T}}\right\}\mu_{j/i}(k)\end{aligned} \tag{7-64}$$

7.6.3　模型并行滤波

将 $\hat{\boldsymbol{x}}_j^0(k\,|\,k)$、$\boldsymbol{P}_j^0(k\,|\,k)$ 作为 k 时刻模型 j 的输入，可以应用第 4 章介绍 Kalman 滤波器得到相应的滤波输出 $\hat{\boldsymbol{x}}_j(k+1\,|\,k+1)$、$\boldsymbol{P}_j(k+1\,|\,k+1)$。

7.6.4　模型概率更新

根据 $k+1$ 时刻的滤波输出再次计算 $k+1$ 时刻的模型概率。若模型 j 滤波残差为 $\boldsymbol{v}_j(k+1)$，相应的协方差为 $\boldsymbol{S}_j(k+1)$，并假定其服从高斯分布，那么模型 j 的可能性为

$$\boldsymbol{\Lambda}_j(k+1)=\frac{1}{\sqrt{\left|2\pi\boldsymbol{S}_j(k+1)\right|}}\exp\left[-\frac{1}{2}\boldsymbol{v}_j^{\mathrm{T}}(k+1)\boldsymbol{S}_j^{-1}(k+1)\boldsymbol{v}_j(k+1)\right] \tag{7-65}$$

根据 Kalman 滤波器可知

$$\begin{cases}\boldsymbol{v}_j(k+1)=\boldsymbol{z}(k+1)-\boldsymbol{H}_j(k+1)\hat{\boldsymbol{x}}_j(k+1|k)\\ \boldsymbol{S}_j(k+1)=\boldsymbol{H}_j(k+1)\boldsymbol{P}_j(k+1|k)\boldsymbol{H}_j^{\mathrm{T}}(k+1)+\boldsymbol{R}_j(k+1)\end{cases} \tag{7-66}$$

式（7-66）表明残差 $\boldsymbol{v}_j(k+1)$ 越小，则模型 j 的可能性 $\boldsymbol{\Lambda}_j(k+1)$ 越大。模型 j 的更新概率为

$$\mu_j(k+1)=\frac{\boldsymbol{\Lambda}_j(k+1)\mu_j(k+1|k)}{\sum_{r=1}^{N}\boldsymbol{\Lambda}_r(k+1)\mu_r(k+1|k)} \tag{7-67}$$

7.6.5　模型输出

根据模型 j 的更新概率计算系统的状态输出估计为

$$\hat{\boldsymbol{x}}(k+1|k+1)=E[\boldsymbol{x}(k+1)|\boldsymbol{Z}^{k+1}]=\sum_{j=1}^{N}\hat{\boldsymbol{x}}_j(k+1|k+1)\mu_j(k+1) \tag{7-68}$$

状态估计方差为

$$\begin{aligned}\boldsymbol{P}(k+1|k+1)&=E\left\{[\boldsymbol{x}(k+1)-\hat{\boldsymbol{x}}(k+1|k+1)][\boldsymbol{x}(k+1)-\hat{\boldsymbol{x}}(k+1|k+1)]^{\mathrm{T}}|\boldsymbol{Z}^{k+1}\right\}\\&=\sum_{j=1}^{N}\boldsymbol{P}_j(k+1|k+1)+[\hat{\boldsymbol{x}}(k+1|k+1)-\hat{\boldsymbol{x}}_j(k+1|k+1)]\\&\quad\cdot[\hat{\boldsymbol{x}}(k+1|k+1)-\hat{\boldsymbol{x}}_j(k+1|k+1)]^{\mathrm{T}}\mu_j(k+1)\end{aligned} \tag{7-69}$$

IMM 算法的特点如下所示。

（1）滤波估计的结果反映在模型概率中，通过模型概率的变化达到自适应调整模型的作用。当前统计模型也利用滤波结果使系统结构呈现出闭环状态，但不同的是当前统计模型调整的是模型参数，而 IMM 调整的是多个模型的组合方式。

（2）算法具有模块化、并行化的计算特点，依据不同应用环境，滤波模型可以选择各种线性和非线性滤波算法，并行计算的各模块可以提高效率。

目前，IMM 算法被应用于各种跟踪系统中，但其性能在很大程度上依赖于所使用的模型集，这也就产生了一个难以调和的矛盾——为了提高估计性能就需要更多的模型来匹配目标，但这些增加的模型会大大增加系统运算量，甚至在某些情况下降低跟踪器的性能。另外，对于弱机动或非机动目标采用 IMM 算法也会造成资源的浪费。

IMM 算法给出了与前面提到的模型不同的建模与跟踪思路，即既然一个模型很难很好地描述实际系统的特性，那就使用多个模型来描述，期望利用多个模型的组合来

更好地描述系统，从而得到更为有效的跟踪。在此基础上，很多文献给出了其他通过求取模型之间转移概率来适应不同应用背景的方法。

基于 MATLAB 程序的 IMM 模型函数如下，模型假设有三个基本模型，分别是（A1,Q1）、（A2,Q2）、（A3,Q3），测量模型中的测量矩阵为 C，测量噪声方差为 R，输入交互概率矩阵 model0。从前面的介绍可以看出，多模型 IMM 和估计方法是“搅”在一起的，需要根据估计的结果来判断和哪一个模型更“相像”，从而来计算估计输出的结果，所以 IMM 的函数输入变量中还包括传感器的测量数据 y 及估计向量初值 x0，测量数据是一个向量，包含所有的测量数据，而输出则是根据这组测量数据得到的状态估计结果，在函数中用 x 表示。

```
function [x]=IMM(A1,Q1,A2,Q2,A3,Q3,C,R,Pij,model0,y,x0)

%设置跟踪初值。
 xei1=x0;pi1=10*eye(3);xxi1=[];
 xei2=x0;pi2=10*eye(3);xxi2=[];
 xei3=x0;pi3=10*eye(3);xxi3=[];
 x=[];pa=[];
%利用 IMM 进行跟踪。
 [ii,jj]=size(Pij);

for i=1:length(y)
    for PPi=1:ii
 Pmodel(PPi,:)=Pij(PPi,:).*model0(PPi);
 end
 sumPm=sum(Pmodel);        %计算式（7-58）
 for PPj=1:jj
     model(:,PPj)=Pmodel(:,PPj)/sumPm(PPj);    %计算式（7-59）
 end

%接下来 3 行计算式（7-63）。
xej1=model(1,1)*xei1+model(2,1)*xei2+model(3,1)*xei3;
xej2=model(1,2)*xei1+model(2,2)*xei2+model(3,2)*xei3;
xej3=model(1,3)*xei1+model(2,3)*xei2+model(3,1)*xei3;

%接下来 3 行计算式（7-64）。
pj1=model(1,1)*(pi1+(xej1-xei1)*(xej1-xei1)')+model(2,1)*(pi2+(xej1-
xei2)*(xej1-xei2)')+model(3,1)*(pi3+(xej1-xei3)*(xej1-xei3)');
pj2=model(1,2)*(pi1+(xej2-xei1)*(xej2-xei1)')+model(2,2)*(pi2+(xej2-
xei2)*(xej2-xei2)')+model(3,2)*(pi3+(xej2-xei3)*(xej2-xei3)');
pj3=model(1,3)*(pi1+(xej3-xei1)*(xej3-xei1)')+model(2,3)*(pi2+(xej3-
xei2)*(xej3-xei2)')+model(3,3)*(pi3+(xej3-xei3)*(xej3-xei3)');
```

```
%下面 3 行利用 Kalman 滤波器进行并行计算。
[xej1,pkj1,v1,S1]=kalmanfun(A1,C,Q1,R,xej1,y(i),pj1);
[xej2,pkj2,v2,S2]=kalmanfun(A2,C,Q2,R,xej2,y(i),pj2);
[xej3,pkj3,v3,S3]=kalmanfun(A3,C,Q3,R,xej3,y(i),pj3);

% 下面 3 行计算式（7-65）。
V1=exp(-v1*inv(S1)*v1)/sqrt(abs(2*pi*S1));
V2=exp(-v2*inv(S2)*v2)/sqrt(abs(2*pi*S2));
V3=exp(-v3*inv(S3)*v3)/sqrt(abs(2*pi*S3));

Vsum=sumPm*[V1,V2,V3]';

uj=sumPm.*[V1,V2,V3]/Vsum;  %计算式（7-68）

xe=xej1*uj(1)+xej2*uj(2)+xej3*uj(3);

%计算式（7-68）。
p=pkj1+(xe-xej1)*(xe-xej1)'*uj(1)+pkj2+(xe-xej2)*(xe-xej2)'*uj(2)+pk
j3+(xe-xej3)*(xe-xej3)'*uj(3);

model0=uj;
xei1=xej1;xei2=xej2;xei3=xej3;
pi1=pkj1;pi2=pkj2;pi3=pkj3;

x=[x xe];
xxi1=[xxi1 xei1];
xxi2=[xxi2 xei2];
xxi3=[xxi3 xei3];
pa=[pa pkj1(1,1)];
end
```

还是利用前面所述的当前统计模型使用的模拟轨迹数据，使用 IMM 方法，看看估计结果有何差别。和前一个程序一样，这段程序先用 load 函数调用所要跟踪的轨迹数据，并设置采样周期（T=10）和测量噪声的方差（R=10000），在产生具有传感器测量噪声的测量数据之后，需要先设置 IMM 模型中使用的三个基本模型，将其选为 CA 模型，但是过程噪声方差并不相同，分别是 0.1、1、10。模型转移概率矩阵式（7-56）设置为

$$\boldsymbol{P}=\begin{bmatrix}0.95 & 0.05 & 0\\ 0.33 & 0.34 & 0.33\\ 0 & 0.5 & 0.95\end{bmatrix}$$

并且 3 个模型的先验概率相同，即为 $\mu(0)=[1/3,\ 1/3,\ 1/3]$，程序中分别使用 Pij 和 model0 来表示模型转移概率矩阵和先验概率。接下来分别调用 IMM 函数来估计横轴、纵轴的运动状态，然后将结果输出。但是，IMM 并没有给出预想得到的估计结果，使用 IMM 方法，在纵轴的估计结果是发散的。原因如下:给出的这 3 个基础模型可能根本无法抓住目标运动的基本特征，因此这样的组合也不能够得到很好的估计。读者可以试着去改变基础模型，看看能否使用 IMM 获得更好的估计。从这个例子可以看出，IMM 方法有时并不很好用，所以很多研究者试图来改进 IMM 模型，相应的参考文献非常多，这里不再一一赘述。

```
clc
clear
summ=0;N=1;
for n=1:N


load Hotarget
T=10;
R=10000;

y=xys+sqrt(R)*randn(size(xys));
pa=[];

%IMM 模型中使用的 3 个基本模型。
 [A1,Q1]=CAmodel(T,0.1);
 [A2,Q2]=CAmodel(T,1);
 [A3,Q3]=CAmodel(T,10);
 C=[1 0 0];

%模型转移参数。
 Pij=[0.95 0.05 0;0.33 0.34 0.33;0 0.05 0.95];
 model0=[1/3,1/3,1/3];

 %估计横轴。
 %设置状态初值。
 x0=[0 0 0]';
 [xx1]=IMM(A1,Q1,A2,Q2,A3,Q3,C,R,Pij,model0,y(1,:),x0);

 %估计纵轴。
[xx2]=IMM(A1,Q1,A2,Q2,A3,Q3,C,R,Pij,model0,y(2,:),x0);
end

plot(xys(1,:),xys(2,:),'b-');hold on
plot(C*xx1,C*xx2,'--');hold off
```

```
figure
subplot(2,1,1),plot(ts,xys(1,:),ts,C*xx1)
subplot(2,1,2),plot(ts,xys(2,:),ts,C*xx2)
figure
subplot(2,1,1),plot(ts,xys(1,:)-C*xx1)
subplot(2,1,2),plot(ts,xys(2,:)-C*xx2)
 covv=diag(cov(xys'-[C*xx1;C*xx2]'))
%
summ=summ+covv;

 summ/N
```

下面来看一下使用轨迹 mytarget 用 IMM 进行估计。和 7.4 节一样，用下面一小段程序：

```
load mytarget
T=0.1;
R=25;
```

来代替原程序中：

```
load Hotarget
T=10;
R=10000;
```

其余程序不变，得到的结果分别如图 7-16～图 7-17 所示，横轴、纵轴的估计方差分别为 8.1097、11.2429。从估计方差的结果来讲，使用 IMM 方法针对数据 mytarget 来说，要比使用当前统计模型结果更差一些。

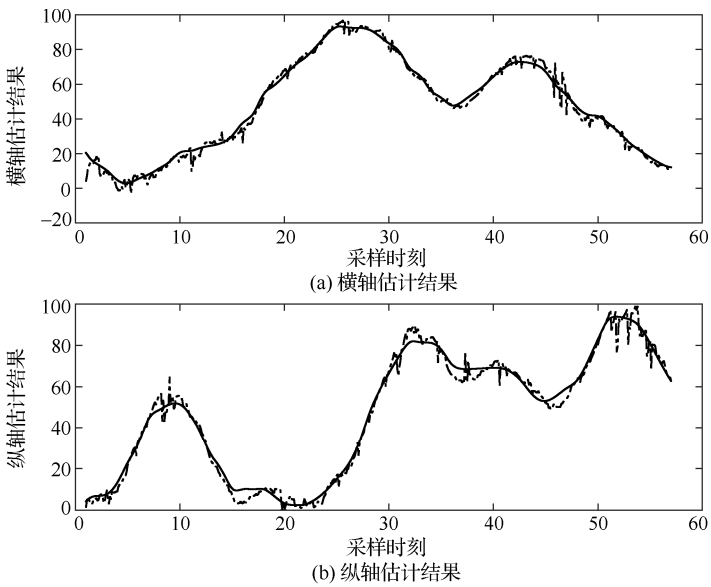


图 7-16　使用数据 mytarget，IMM 跟踪轨迹的结果
实线表示真实轨迹，虚线表示利用传感器测量数据估计的结果

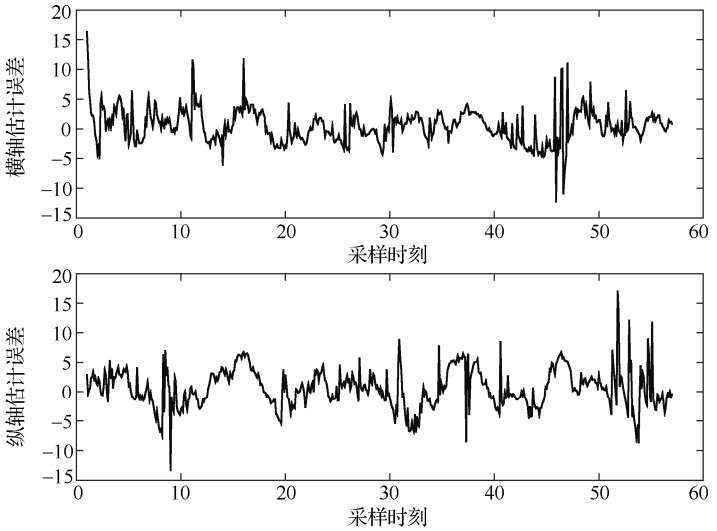


图 7-17　使用数据 mytarget，IMM 估计结果与真实估计之差

7.7　数据驱动模型数学基础

到目前为止，介绍了 CV 模型、CA 模型、Singer 模型、Jerk 模型及 IMM 模型，这些模型的共同特点是需要对目标的机动特性进行先验假设，这其实是离线模型辨识的问题。随着目标跟踪应用背景的不断扩大，面对大量多种一无所知的目标，由于先验知识的缺乏，这些基于假设的模型一般很难得到较好的跟踪效果。在 7.1 节机动模型研究的基础上，先讨论以下几个问题。

既然已经有许多机动模型研究成果，研究另外的建模方法还有必要吗？作为以武器打击为研究目的的军用背景从 20 世纪 70 年代以来极大地推进了机动目标的研究，利用雷达跟踪战机的机动，迅速而准确地打击要求机动目标研究必须立足于准确的目标模型和能够快速跟踪的方法上，由此，人们研究了战机等目标的机动特征并得到了相关动力学描述模型，研究成果最大的成就在于特定类型目标的相关模型参数的先验值方面。近年来机动目标跟踪背景发生了很大的变化，部分民用背景跟踪目标的出现大大增加了机动目标的类型，目标的机动特性与原军事背景广泛研究的战机运动有很大区别，如视频跟踪中汽车、人及物联网中被跟踪物品的运动特性等。对于大规模出现的民用目标，研究其准确的先验知识几乎是不可能的。因此，研究不再基于目标先验知识假设的机动目标建模方法是十分有必要的。

既然不再刻意于研究目标的先验知识，那么还有什么可以利用来得到目标的运动模型呢？在只能得到测量数据的条件下，有效利用测量数据是首先想到的方法。因此，基于数据驱动的建模与跟踪方法应运而生，近年来已成为一个热点研究方向。基于“数

据中含有模型而模型融于数据”的基本思想，这种方法理论上只要求存在机动目标的运动测量数据就可以得到目标模型并进行跟踪。根据第 6 章的知识可以知道，得到的数据中含有系统噪声，需要利用准确的系统模型，才能在存在这些噪声的情况下得到状态的最优估计，但目前系统模型参数也需要进行估计，因此，整个系统同时存在着两个估计过程，如何保证整个系统在两个估计过程下收敛是这种数据驱动建模方法的关键。

下面给出基于数据驱动的机动模型所要研究的问题描述。

式（7-31）的离散化模型为

$$a(k+1)=\beta a(k)+w^a(k) \tag{7-70}$$

可知 $w^a(k)$ 是零均值白噪声序列，方差为 σ_{aw}^2，根据第 6 章介绍的离散化方法，有如下关系：

$$\sigma_{aw}^2=\sigma_a^2(1-\beta^2) \tag{7-71}$$

$$\beta=\mathrm{e}^{-\alpha T_0} \tag{7-72}$$

从式（7-70）可知，机动加速度序列 $a(k)$ 为一阶自回归（Auto-Regressive，AR）模型，若已知 $a(k)(k=1,2,3,\cdots,K)$，则可以根据 $a(k)$ 序列估计式（7-70）中的参数 β 及方差 σ_{aw}^2，进而根据式（7-71）和式（7-72）得到式（7-31）模型中的机动频率 α 和目标机动加速度 $a(t)$ 的方差 σ_a^2。

另外，当机动频率 α 为零时，即目标没有机动，而是做匀加速运动时 $\beta=1$，有机动即 $\alpha>0$ 时，$1>\beta>0$。从系统分析的角度来讲，这保证了单输入单输出离散系统（7-70）的稳定。分析上述基于数据驱动的机动模型问题的描述可知，与 Singer 模型和当前统计模型等不同的是，将要利用加速度数据对相关参数进行估计，而不再通过假定加速度噪声满足某种分布来求取机动频率 α 和方差 σ_a^2。

基于上述目的，本节先研究随机过程中的参数估计问题。为了与前面讲述的加速度噪声相区别，本节使用 $p(k)$ 表示一个具有普遍意义的随机过程，写成自回归模型的形式为

$$p(k)=\sum_{m=1}^{M}\alpha_m p(k-m)+w_p(k) \tag{7-73}$$

式中，$w_p(k)$ 为零均值白噪声，其方差为 σ_{pw}^2。

下面介绍几种方法来估计式（7-73）中的参数 α_m 及 $w_p(k)$ 的方差 σ_{pw}^2，其中 $m=0,1,2,\cdots,M$。

7.7.1　最小二乘估计方法

本节介绍如何使用最小二乘（Least Squares，LS）估计方法来得到想要获得的参

数。设 $\boldsymbol{\theta}=[\hat{\alpha}_1 \quad \hat{\alpha}_2 \quad \cdots \quad \hat{\alpha}_M]^{\mathrm{T}}$ 为参数的估计，对于式（7-73），将待估计参数 $\boldsymbol{\theta}$ 与测量数据 $p(k)$ 写成如下线性关系：

$$p(k)=\theta \boldsymbol{h}(k)+w_p(k) \tag{7-74}$$

式中

$$\boldsymbol{h}(k)=[p(k-1) \quad p(k-2) \quad \cdots \quad p(k-M)] \tag{7-75}$$

下面利用递推的方法在线对参数进行估计：

$$\hat{\boldsymbol{\theta}}(k)=\hat{\boldsymbol{\theta}}(k-1)+\boldsymbol{K}(k)[p(k)-\boldsymbol{h}^{\mathrm{T}}(k)\hat{\boldsymbol{\theta}}(k-1)] \tag{7-76}$$

式中

$$\boldsymbol{K}(k)=\frac{\boldsymbol{\Sigma}(k-1)\boldsymbol{h}(k)}{\hat{\delta}_{pw}^2(k)+\boldsymbol{h}^{\mathrm{T}}(k)\boldsymbol{\Sigma}(k-1)\boldsymbol{h}(k)} \tag{7-77}$$

$$\boldsymbol{\Sigma}(k)=[\boldsymbol{I}-\boldsymbol{K}(k)\boldsymbol{h}^{\mathrm{T}}(k)]\boldsymbol{\Sigma}(k-1) \tag{7-78}$$

但是并不知道零均值白噪声 $w_p(k)$ 的方差 δ_{pw}^2。其实方差 δ_{pw}^2 可以根据估计参数 $\hat{\boldsymbol{\theta}}$ 求出，从理论上讲，如果知道了参数 $\hat{\boldsymbol{\theta}}$ 的值，则将 $\hat{\boldsymbol{\theta}}$ 代入式（7-74）得到 $s(k)$ 后可以按照式（7-79）进行计算：

$$\delta_{pw}^2=E\left[(p(k)-\boldsymbol{\theta}\boldsymbol{h}(k))(p(k)-\boldsymbol{\theta}\boldsymbol{h}(k))^{\mathrm{T}}\right]=\frac{1}{\boldsymbol{K}}\sum_{k=1}^{K}[p(k)-\boldsymbol{\theta}\boldsymbol{h}(k)]^2 \tag{7-79}$$

进而得到方差 δ_{pw}^2 的递推关系：

$$\hat{\delta}_{pw}^2(k)=\frac{1}{k}\left\{(k-1)\hat{\delta}_{pw}^2(k-1)+[p(k)-\boldsymbol{h}^{\mathrm{T}}(k)\hat{\boldsymbol{\theta}}(k-1)][p(k)-\boldsymbol{h}^{\mathrm{T}}(k)\hat{\boldsymbol{\theta}}(k-1)]^{\mathrm{T}}\right\} \tag{7-80}$$

利用式（7-76）～式（7-78）、式（7-80）的递推方法就可以求出式（7-73）中参数 α_m 及 $w_p(k)$ 的方差 σ_{pw}^2。注意到，上述方法是在递推求解 α_m 时需要用到 σ_{pw}^2 的估计，而递推求解 σ_{pw}^2 时也需要用到 α_m 量，在两者都不确切知道时进行递推。这样的方法其实非常“危险”，会引起所谓的累积误差，试想两个人 *A* 和 *B* 在谈话，*A* 的谈话内容是根据 *B* 得来的，而 *B* 的谈话内容是根据 *A* 得来的，这样在其中一个有一点错误的时候，两个人的谈话内容就会可能越来越“离谱”，上面的这个方法也很容易引起这样的问题。但是，在既需要估计参数，也需要估计状态的时候，这样的方法也不失为一种可行的方法。要想完全消除累积误差也是不太可能的，因此需要注意两点：一是承认会有累积误差的存在，并且想办法定量计算该方法的累积误差；二是想办法将累积误差控制在一定的范围之内。

7.7.2　Yule-Walker 估计方法

Yule-Walker 方法可以直接估计出如式（7-73）的方程系数及噪声方差。设 $r(m)$ $(m=1,2,\cdots,M)$，是 $p(k)$ 的自相关函数。对于式（7-73），Yule-Walker 估计方程为

$$\begin{bmatrix} r(0) & r(1) & \cdots & r(M-1) & r(M) \\ r(1) & r(0) & \cdots & r(M-2) & r(M-1) \\ \vdots & \vdots & & \vdots & \vdots \\ r(M) & \cdots & \cdots & r(1) & r(0) \end{bmatrix} \begin{bmatrix} 1 \\ -\alpha_1 \\ \vdots \\ -\alpha_M \end{bmatrix} = \begin{bmatrix} \sigma_w^2 \\ 0 \\ \vdots \\ 0 \end{bmatrix} \tag{7-81}$$

式中，从第二行到最后一行（用 $j=1,2,\cdots,M$ 表示）可以表示为如下差分方程：

$$r(j)-\sum_{m=1}^{M} a_m r(j-m)=0, \quad j=1,2,\cdots,M \tag{7-82}$$

则利用线性方程组求解方法，可得求解式（7-82）的 M 个未知数 $\hat{a}_m$ $(j=1,2,\cdots,M)$，再利用式（7-81）中的第一行得到

$$\hat{\delta}_w^2 = r(0)-\sum_{m=1}^{M} \hat{a}_m r(m) \tag{7-83}$$

在理论上计算 $r(m)$ 比较复杂，可以利用已知序列对自相关函数进行估算，方法如下：

$$\hat{r}(m)=\begin{cases} \dfrac{1}{K}\displaystyle\sum_{k=m}^{K-1} p(k)p(k-m), & 0 \leqslant m \leqslant K-1 \\ \hat{r}(-m), & -(K-1) \leqslant m < 0 \\ 0, & |m| \geqslant K \end{cases} \tag{7-84}$$

这里为进行在线估计，利用已知随机序列 $p(k)$ 递推计算 $\hat{r}(m)$：

$$\hat{r}_k(m)=\hat{r}_{k-1}(m)+\frac{1}{k}[p(k)p(k-m)-\hat{r}_{k-1}(m)]$$

$$\hat{r}_k(0)=\hat{r}_{k-1}(0)+\frac{1}{k}[p(k)p(k)-\hat{r}_{k-1}(0)]$$

一般假设 $k \leqslant m$ 时 $p(k)$ 为零。

利用式（7-84）代替式（7-82）及式（7-83）中的自相关函数 $r(m)$ 可以得到 $\hat{a}_m$ 及 $\hat{\delta}_w^2$。由于序列有限，$\hat{r}(m)$ 与其理论值 $r(m)$ 并不完全相等，因此，用 $\hat{r}(m)$ 来代替式（7-82）中的 $r(m)$ 时等式不再成立，有

$$\hat{r}(j)-\sum_{m=1}^{M} a_m \hat{r}(j-m)=\varepsilon(j), \quad j=1,2,\cdots,M \tag{7-85}$$

式中，$\varepsilon(j)(j=1,2,\cdots,M)$ 表示利用序列计算方法式（7-84）引入的估计误差。式（7-85）也可写为

$$\hat{r}(j)=\sum_{m=1}^{M} a_m \hat{r}(j-m)+\varepsilon(j), \quad j=1,2,\cdots,M \tag{7-86}$$

利用线性代数求解的方法显然已经不再适用于式（7-86）的求解，可以使用最小二乘估计的方法得到 $\hat{a}_m$。

设

$$\boldsymbol{x}=\begin{bmatrix}\hat{r}(1)\\ \hat{r}(2)\\ \vdots\\ \hat{r}(M)\end{bmatrix},\quad \boldsymbol{\theta}=\begin{bmatrix}a_1\\ a_2\\ \vdots\\ a_M\end{bmatrix},\quad \boldsymbol{H}=\begin{pmatrix}\hat{r}(0) & \cdots & \hat{r}(M-1)\\ \vdots & & \vdots\\ \hat{r}(M-1) & \cdots & \hat{r}(0)\end{pmatrix} \tag{7-87}$$

式中，$\boldsymbol{\theta}$ 最小二乘估计为 $(\boldsymbol{H}^{\mathrm{T}}\boldsymbol{H})^{-1}\boldsymbol{H}^{\mathrm{T}}\boldsymbol{x}$，该方法被称为最小二乘估计修正的 Yule-Walker 方法。为说明最小二乘方法、Yule-Walker 方法的参数估计效果，给出以下仿真研究实例。设系统模型为一阶 AR 模型为 $p(k)=\alpha_1 p(k-1)+w_p(k)$，在 $a_1=0.8$、$w_p(k)$ 的方差 $\sigma_{pw}^2=2$ 的参数下仿真得到 4000 个数据。利用 500 次 Monte Carlo 仿真，使用递推的最小二乘算法式（7-76）～式（7-80）及 Yule-Walker 方法得到的估计结果 $\hat{a}_1$ 及 $\hat{\sigma}_{pw}^2$ 如表 7-1 所示。递推最小二乘方法、Yule-Walker 方法的参数收敛情况如图 7-18 和图 7-19 所示。可以看出，Yule-Walker 方法得到的稳态估计误差较大，且在递推开始时的振荡比最小二乘估计方法大。

表 7-1　两种估计方法的比较

估计方法	估计量的收敛值 $\hat{a}_1$	估计误差 $\dfrac{\lvert\hat{a}_1-a_1\rvert}{a_1}\times100\%$	估计量 $\hat{\delta}_{pw}^2$	$\dfrac{\lvert\hat{\delta}_{pw}^2-\delta_{pw}^2\rvert}{\delta_{pw}^2}\times100\%$
递推的最小二乘算法式(7-76)～式(7-80)	0.7917	1.0375%	1.9255	3.725%
修正的 Yule- Walker 方法	0.7916	1.05%	1.9250	3.75%

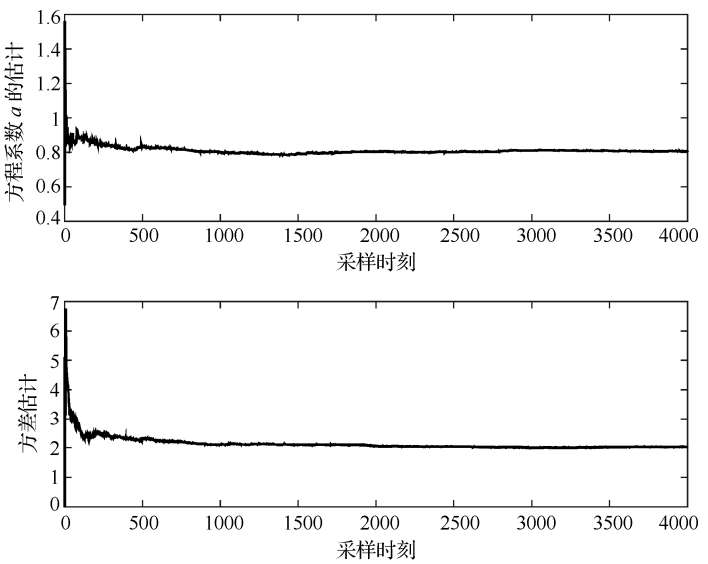


图 7-18　递推最小二乘估计方法的参数收敛情况

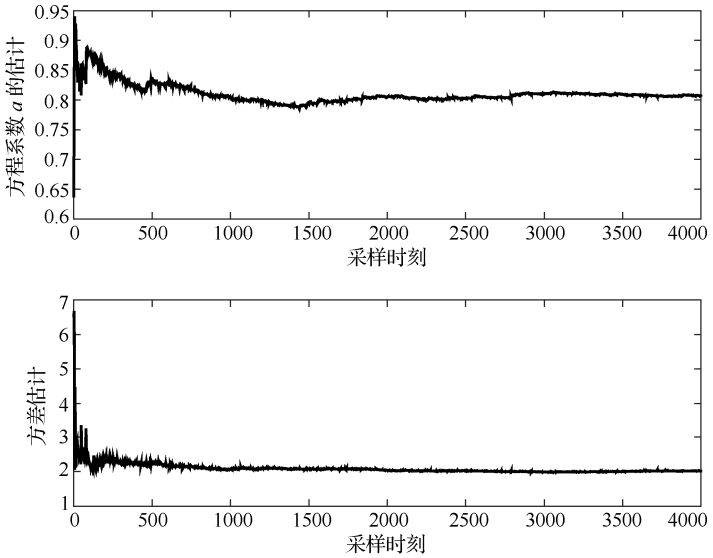


图 7-19　Yule-Walker 估计方法参数的收敛性

下面给出 MATLAB 仿真实现程序。所要估计的是一阶 AR 模型，形如式（7-70），其中参数 β 及方差 σ_{aw}^2 的真实值分别为 0.8 和 2（在程序中这两个量分别用 a 和 q 表示）。然后使用 for 循环产生了待估计序列的 4000 个数据，为了让估计结果评价更为客观，执行 500 次程序，再进行平均求估计结果，即采用所谓的 Monte Carlo 仿真的方法。程序中分别调用最小二乘递推方法、Yule-Walker 估计方法进行估计，因为这两个方法都是递推的，所以采用最后一个估计值作为估计的最终结果，把最后一次的估计递推过程化出来，用黑色线，并且加粗画出来，这样看起来更清楚。

程序中先给出了参数 a 和 q 的真值，然后产生带有待估计参数的系列 p，再分别使用最小二乘的递推方法和 Yule-Walker 方法（函数 LSRecursive 和 YuleWalker）对参数进行估计。

```
clc
clear
%产生待估计序列 p(k+1)=a*p(k)+w(t)。
a=0.8;
q=2;    %w(t)的方差
p0=3;p=[];
for i=1:4000
    p=[p p0];
    p0=a*p0+sqrt(q)*randn(1);
end

sum_ar=0;
```

```
sum_qr=0;
sum_aY=0;
sum_qY=0;

J=500;
for j=1:J  %利用 500 次估计来计算平均结果
%利用最小二乘递推方法得到的估计参数。
[ar,qr]=LSRecursive(p);
arlast=ar(max(length(ar)));
qrlast=qr(max(length(qr)));

%利用 Yule-Walker 方法得到的估计参数。
[aY,qY]=YuleWalker(p);
aYlast=aY(max(length(aY)));
qYlast=qY(max(length(qY)));

sum_ar=sum_ar+arlast;
sum_qr=sum_qr+qrlast;
sum_aY=sum_aY+aYlast;
sum_qY=sum_qY+qYlast;
end

arl=sum_ar/J%求平均值
qrl=sum_qr/J
aYl=sum_aY/J
qYl=sum_qY/J

subplot(2,1,1),plot(ar,'k','LineWidth',2)
xlabel('采样时刻'),ylabel('方程系数 a 的估计')
subplot(2,1,2),plot(qr,'k','LineWidth',2)
xlabel('采样时刻'),ylabel('方差估计')
figure
subplot(2,1,1),plot(aY,'k','LineWidth',2)
xlabel('采样时刻'),ylabel('方程系数 a 的估计')
subplot(2,1,2),plot(qY,'k','LineWidth',2)
xlabel('采样时刻'),ylabel('方差估计')
```

下面程序是实现递推最小二乘估计方法的函数。

```
function [a,q]=LSRecursive(p)
a0=0.5;
Q0=100;
q0=0;
```

```
a=[];q=[];
a=[a a0];
q=[q q0];
for k=2:length(p)
    q0=((k-1)*q0+(p(k)-a0*p(k-1))^2)/k;
    K=Q0*p(k-1)/(q0+p(k-1)*Q0*p(k-1));
    a0=a0+K*(p(k)-a0*p(k-1));
    Q0=(1-K*p(k-1))*Q0;
    a=[a a0];
    q=[q q0];
end
```

下面程序是实现 Yule-Walker 方法的函数。

```
function [a,q]=YuleWalker(p)
a=[];q=[];
r0=0;
r1=0;
for k=2:length(p)
    r0=r0+(p(k)*p(k)-r0)/k;
    r1=r1+(p(k)*p(k-1)-r1)/k;
    P=r1;
H=r0;
a0=inv(H'*H)*H'*P;
q0=r0-a0*r1;
a=[a a0];
q=[q q0];
end
```

7.8　自适应参数机动目标模型估计方法

本节将在 7.7 节讨论数据模型基本思想的基础上，给出一种自适应机动目标模型，该模型首先根据目标实际运动中加速度满足一阶 AR 模型的关系，建立含有系统自适应参数的目标运动模型；然后，根据建立的目标运动模型，对目标运动特性进行估计，计算目标当前状态估计值，最后，根据目标加速度估计值修正系统自适应参数，再利用修正的系统自适应参数更新目标的运动模型，利用更新的目标运动模型进行下一次预测与估计。

下面详细给出该算法的运行步骤。

步骤 1：目标运动状态和系统自适应参数初始化。

（1）设置状态初值 $\hat{\boldsymbol{x}}(0|0)=\boldsymbol{x}_0$，一般将其设为 3 维全 0 列向量，维数为系统模型中状态向量的维数，选择位移、速度以及加速度为状态向量。

（2）系统自适应参数初值 $\alpha=\alpha_0$ 和 $\delta_a^2=\delta_{a0}^2$ 取任意正数，例如，α_0 取值 $\frac{1}{3}$，δ_{a0}^2 取值 3。

（3）自相关函数初值 $r_0(0)$ 和 $r_0(1)$ 的初值取为 $r_0(0)=\ddot{x}_0\ddot{x}_0=0$，$r_0(1)=\ddot{x}_0=0$。

（4）系统加速度分量初值 $\overline{a}(0)=\overline{a}_0$，一般 $\overline{a}_0$ 取 0。

步骤 2：建立具有系统自适应参数的运动模型。

（1）描述目标的运动特征。

和 Singer 模型类似，设目标加速度为非零均值的时间相关随机过程 $\ddot{x}(t)=\overline{a}+a(t)$，其中 $\overline{a}$ 为加速度均值，$a(t)$ 为零均值指数相关有色噪声模型，其相关函数为

$$R_a(\tau)=E[a(t)a(t+\tau)]=\delta_a^2\mathrm{e}^{-\alpha|\tau|}$$

式中，$R_a(\tau)$ 表示相关函数，δ_a^2 表示加速度方差，α 为机动频率，反映目标的机动随机特性。

对有色噪声 $a(t)$ 做白化处理，得到

$$\dot{a}(t)=-\alpha a(t)+w(t)$$

式中，$w(t)$ 为零均值白噪声，方差为 $\delta_w^2=2\alpha\delta_a^2$。

由 $\ddot{x}(t)=\overline{a}+a(t)$ 和 $\dot{a}(t)=-\alpha a(t)+w(t)$ 得到目标运动的连续状态方程：

$$\dot{\boldsymbol{x}}(t)=\begin{bmatrix}0&1&0\\0&0&1\\0&0&-\alpha\end{bmatrix}\boldsymbol{x}(t)+\begin{bmatrix}0\\0\\\alpha\end{bmatrix}\overline{a}(t)+\begin{bmatrix}0\\0\\1\end{bmatrix}w(t)$$

以周期 T 采样，离散化后系统目标运动满足以下方程：

$$\boldsymbol{x}(k+1)=\boldsymbol{A}(k+1,k)\boldsymbol{x}(k)+\boldsymbol{U}(k)\overline{a}(k)+\boldsymbol{w}(k)\tag{7-88}$$

式中，$\boldsymbol{x}=[x,\dot{x},\ddot{x}]^{\mathrm{T}}$ 为 3 维状态列向量，$x,\dot{x},\ddot{x}$ 分别是位移、速度和加速度，$\boldsymbol{x}(k+1)$ 为 $k+1$ 时刻目标的状态向量，k 为采样时刻，$\boldsymbol{A}(k+1,k)$ 为状态转移矩阵，$\boldsymbol{x}(k)$ 为 k 时刻目标的状态向量；$\boldsymbol{U}(k)$ 为控制矩阵；$\overline{a}(k)$ 为 0 时刻开始至 k 时刻目标的加速度均值；$\boldsymbol{w}(k)$ 为过程噪声，其均值为 0，方差为 $\boldsymbol{Q}(k)$；所述 $\boldsymbol{A}(k+1,k)$、$\boldsymbol{U}(k)$ 及 $\boldsymbol{Q}(k)$ 中含有目标机动频率 α 和目标机动加速度方差 δ_a^2，随着系统自适应参数的变化而变化；状态转移矩阵 $\boldsymbol{A}(k+1,k)$ 的表达式如下：

$$\boldsymbol{A}(k+1,k)=\begin{bmatrix}1&T&\dfrac{\alpha T-1+\mathrm{e}^{-\alpha T}}{\alpha^2}\\0&1&\dfrac{1-\mathrm{e}^{-\alpha T}}{\alpha}\\0&0&\mathrm{e}^{-\alpha T}\end{bmatrix}\tag{7-89}$$

控制矩阵 $\boldsymbol{U}(k)$ 的表达式如下：

$$\boldsymbol{U}(k)=\begin{bmatrix}\dfrac{1}{\alpha}\left(-T+\dfrac{\alpha T^2}{2}+\dfrac{1-\mathrm{e}^{-\alpha T}}{\alpha}\right)\\ T-\dfrac{1-\mathrm{e}^{-\alpha T}}{\alpha}\\ 1-\mathrm{e}^{-\alpha T}\end{bmatrix} \tag{7-90}$$

过程噪声 $w(k)$ 的方差 $\boldsymbol{Q}(k)$ 的表达式如下：

$$\boldsymbol{Q}(k)=E[\boldsymbol{w}(k)\boldsymbol{w}^{\mathrm{T}}(k)]=2\alpha\delta_a^2\begin{bmatrix}q_{11}&q_{12}&q_{13}\\ q_{12}&q_{22}&q_{23}\\ q_{13}&q_{23}&q_{33}\end{bmatrix}$$

式中

$$\begin{cases}q_{11}=\dfrac{1}{2\alpha^5}\left[1-\mathrm{e}^{-2\alpha T}+2\alpha T+\dfrac{2\alpha^3T^3}{3}-2\alpha^2T^2-4\alpha T\mathrm{e}^{-\alpha T}\right]\\ q_{12}=\dfrac{1}{2\alpha^4}[\mathrm{e}^{-2\alpha T}+1-2\mathrm{e}^{-\alpha T}+2\alpha T\mathrm{e}^{-\alpha T}-2\alpha T+\alpha^2T^2]\\ q_{13}=\dfrac{1}{2\alpha^3}[1-\mathrm{e}^{-2\alpha T}-2\alpha T\mathrm{e}^{-\alpha T}]\\ q_{22}=\dfrac{1}{2\alpha^3}[4\mathrm{e}^{-\alpha T}-3-\mathrm{e}^{-2\alpha T}+2\alpha T]\\ q_{23}=\dfrac{1}{2\alpha^2}[\mathrm{e}^{-2\alpha T}+1-2\alpha T]\\ q_{33}=\dfrac{1}{2\alpha}[1-\mathrm{e}^{-2\alpha T}]\end{cases} \tag{7-91}$$

（2）考虑目标运动的测量方程：

$$\boldsymbol{y}(k)=\boldsymbol{H}(k)\boldsymbol{x}(k)+\boldsymbol{v}(k) \tag{7-92}$$

式中，k 为采样时刻，$\boldsymbol{y}(k)$ 为目标在 k 时刻的观测值，$\boldsymbol{H}(k)$ 为测量矩阵，$\boldsymbol{x}(k)$ 为 k 时刻目标的状态向量，$\boldsymbol{v}(k)$ 为高斯测量白噪声，其方差为 $\boldsymbol{R}$，且与过程噪声 $w(k)$ 相互独立。

步骤 3：根据建立的具有系统自适应参数的运动模型对目标状态进行预测。

（1）根据建立的具有系统自适应参数的运动模型和初始值完成目标状态的一步预测，预测方程式如下：

$$\hat{\boldsymbol{x}}(k\,|\,k-1)=\boldsymbol{A}(k,k-1)\hat{\boldsymbol{x}}(k-1\,|\,k-1)+\boldsymbol{U}(k-1)\bar{a}(k-1) \tag{7-93}$$

式中，$\hat{\boldsymbol{x}}(k\,|\,k-1)$ 表示 $k-1$ 时刻预测目标在 k 时刻的状态，k 为采样时刻，$\boldsymbol{A}(k,k-1)$ 为状态转移矩阵，$\hat{\boldsymbol{x}}(k-1\,|\,k-1)$ 表示目标 $k-1$ 时刻目标的状态估计值，$\boldsymbol{U}(k-1)$ 为控制矩阵，$\bar{a}(k-1)$ 为从 0 时刻开始至 $k-1$ 的加速度均值。

（2）按照式（7-94）完成目标状态协方差的一步预测：

$$\boldsymbol{P}(k|k-1)=\boldsymbol{A}(k,k-1)\boldsymbol{P}(k-1|k-1)\boldsymbol{A}^{\mathrm{T}}(k,k-1)+\boldsymbol{Q}(k-1) \tag{7-94}$$

式中，$\boldsymbol{P}(k|k-1)$ 表示 $k-1$ 时刻预测目标在 k 时刻的状态协方差，k 为采样时刻，|表示条件操作符，$\boldsymbol{P}(k-1|k-1)$ 表示 $k-1$ 时刻目标的状态协方差的估计值，$\boldsymbol{A}(k,k-1)$ 为状态转移矩阵，$\boldsymbol{Q}(k-1)$ 为过程噪声协方差。

步骤 4：根据目标的状态预测值、观测数据值和目标的状态协方差预测值对目标状态进行更新。

（1）根据目标状态协方差预测值、测量矩阵及测量噪声方差按照式（7-95）计算滤波器增益：

$$\boldsymbol{K}(k)=\boldsymbol{P}(k|k-1)\boldsymbol{H}^{\mathrm{T}}(k)[\boldsymbol{H}(k)\boldsymbol{P}(k|k-1)\boldsymbol{H}^{\mathrm{T}}(k)+\boldsymbol{R}]^{\mathrm{T}} \tag{7-95}$$

式中，$\boldsymbol{K}(k)$ 为滤波器增益，k 为采样时刻，$\boldsymbol{P}(k|k-1)$ 表示 $k-1$ 时刻预测目标在 k 时刻的状态协方差，$\boldsymbol{H}(k)$ 为 k 时刻的测量矩阵，$\boldsymbol{R}$ 为高斯测量白噪声的方差，$\boldsymbol{H}^{\mathrm{T}}(k)$ 为 k 时刻测量矩阵的转置。

（2）利用目标状态预测值和观测数据值计算目标当前状态估计值：

$$\hat{\boldsymbol{x}}(k|k)=\hat{\boldsymbol{x}}(k|k-1)+\boldsymbol{K}(k)[\boldsymbol{y}(k)-\boldsymbol{H}(k)\hat{\boldsymbol{x}}(k|k-1)] \tag{7-96}$$

式中，$\hat{\boldsymbol{x}}(k|k)$ 表示 k 时刻目标的状态估计值，$\hat{\boldsymbol{x}}(k|k-1)$ 表示 $k-1$ 时刻预测目标在 k 时刻的状态，k 为采样时刻，$\boldsymbol{K}(k)$ 为 k 时刻滤波器增益，$\boldsymbol{y}(k)$ 为雷达接收数据在 k 时刻的目标观测值，$\boldsymbol{H}(k)$ 为 k 时刻的测量矩阵。

（3）计算目标状态协方差的估计值：

$$\boldsymbol{P}(k|k)=[\boldsymbol{I}-\boldsymbol{K}(k)\boldsymbol{H}(k)]\boldsymbol{P}(k|k-1) \tag{7-97}$$

式中，$\boldsymbol{I}$ 是 3 维单位矩阵，$\boldsymbol{P}(k|k)$ 表示 k 时刻的目标状态协方差的估计值，k 为采样时刻，$\boldsymbol{K}(k)$ 为 k 时刻滤波器增益，$\boldsymbol{H}(k)$ 为 k 时刻的测量矩阵，$\boldsymbol{P}(k|k-1)$ 表示 $k-1$ 时刻预测目标在 k 时刻的状态协方差。

步骤 5：根据目标状态估计值计算目标加速度均值及目标加速度估计值。

（1）计算目标加速度均值：

$$\bar{a}(k)=\frac{1}{k}\sum_{k=0}^{k-1}\hat{\dot{x}}(k|k) \tag{7-98}$$

式中，$\bar{a}(k)$ 为 0 至 k 时刻加速度均值，$\hat{\dot{x}}(k|k)$ 为 k 时刻目标的状态估计值 $\hat{\boldsymbol{x}}(k|k)$ 的第三行值，k 为采样时刻。

（2）获取系统 $k-1$ 时刻和 k 时刻的加速度估计值 $\hat{a}(k-1)$、$\hat{a}(k)$：

$$\hat{a}(k-1)=\hat{\dot{x}}(k-1|k-1) \tag{7-99}$$

$$\hat{a}(k)=\hat{\dot{x}}(k|k) \tag{7-100}$$

式中，$\hat{\dot{x}}(k-1|k-1)$ 为 $k-1$ 时刻状态估计 $\hat{x}(k-1|k-1)$ 的第三行值，$\hat{\dot{x}}(k|k)$ 为 k 时刻状态估计 $\hat{x}(k|k)$ 的第三行值。

步骤 6：根据目标加速度估计值对系统自适应参数进行修正。

根据采样时刻 k 值的大小，选择修正系统自适应参数 α 和 δ_a^2 的方法，若 k 小于等于 4 进入步骤（1），若 k 大于 4 进入步骤（2）。

（1）当采样时刻 k 小于等于 4 时，因为采样数据较少，采用当前统计模型的参数取值方法，按下式计算系统自适应参数 α 和 δ_a^2。

$\alpha=\alpha_0$，其中 α_0 为系统自适应参数 α 的初值。

若 $\hat{a}(k)>0$，则取 $\sigma_a^2=\dfrac{4-\pi}{\pi}[a_{\mathrm{M}}-\hat{a}(k)]^2$；

若 $\hat{a}(k)<0$，则取 $\sigma_a^2=\dfrac{4-\pi}{\pi}[\hat{a}(k)-a_{-\mathrm{M}}]^2$；

若 $\hat{a}(k)=0$，则 σ_a^2 取 (0,10] 之间的任意数。

其中，$\hat{a}(k)$ 为 k 时刻目标加速度估计值，π 为圆周率，取为 3.14，a_{M} 为正的常数，取为 3，$a_{-\mathrm{M}}$ 为与 a_{M} 绝对值相等的负常数，取为–3。

（2）当采样时刻 k 大于 4 时，利用 Yule-Walker 方法，按式（7-101）计算系统自适应参数 α 和 δ_a^2：

$$\begin{cases} r_k(1)=r_{k-1}(1)+\dfrac{1}{k}[b\cdot\hat{a}(k)\hat{a}(k-1)-r_{k-1}(1)] \\ r_k(0)=r_{k-1}(0)+\dfrac{1}{k}[b\cdot\hat{a}(k)\hat{a}(k)-r_{k-1}(0)] \end{cases} \tag{7-101}$$

式中，b 是大于 1 的常数，$r_k(1)$ 为 k 时刻目标加速度向前一步相关函数，$r_{k-1}(1)$ 为 $k-1$ 时刻目标加速度向前一步相关函数，$\hat{a}(k-1)$ 和 $\hat{a}(k)$ 分别为 $k-1$ 时刻和 k 时刻目标加速度估计值，$r_k(0)$ 为 k 时刻目标加速度自相关函数，$r_{k-1}(0)$ 为 $k-1$ 时刻目标加速度自关函数。

例如，取式（7-101）中的 b 为 10，即

$$\begin{cases} r_k(1)=r_{k-1}(1)+\dfrac{1}{k}[10\hat{a}(k)\hat{a}(k-1)-r_{k-1}(1)] \\ r_k(0)=r_{k-1}(0)+\dfrac{1}{k}[10\hat{a}(k)\hat{a}(k)-r_{k-1}(0)] \end{cases}$$

根据系统方程 $\boldsymbol{x}(k+1)=\boldsymbol{A}(k+1,k)\boldsymbol{x}(k)+\boldsymbol{U}(k)\bar{a}(k)+w(k)$ 得到目标运动加速度，方程满足如下一阶马尔可夫随机序列：

$$\hat{a}(k+1)=\beta\hat{a}(k)+w^a(k) \tag{7-102}$$

式中，$\hat{a}(k+1)$ 为 $k+1$ 时刻的加速度，$\hat{a}(k)$ 为 k 时刻的加速度，β 为离散后加速度随

机序列的机动频率，$w^a(k)$ 为零均值白噪声离散序列，方差为 $\delta_{aw}^2=\delta_a^2(1-\beta^2)$，其中 δ_a^2 为零均值白噪声 $w(t)$ 的方差，β 与 α 的关系为 $\beta=\mathrm{e}^{-\alpha T}$。

一阶马尔可夫时间加速度序列满足以下参数关系：

$$\beta=\frac{r_k(1)}{r_k(0)},\quad \delta_{aw}^2=r_k(0)-\alpha r_k(1) \tag{7-103}$$

式中，$r_k(1)$ 为 k 时刻的加速度向前一步相关函数，$r_k(0)$ 为 k 时刻的加速度自相关函数，α 与 β 分别为加速度的机动频率及其离散化后加速度序列的机动频率，自适应参数 α 和 δ_a^2 可通过计算得到

$$\alpha=-\frac{\ln r_k(1)-\ln r_k(0)}{T} \tag{7-104}$$

$$\delta_a^2=\frac{r_k(0)-\alpha r_k(1)}{1-\left(\dfrac{r_k(1)}{r_k(0)}\right)^2} \tag{7-105}$$

式中，$r_k(1)$ 为 k 时刻的加速度向前一步相关函数，$r_{k-1}(0)$ 为 k 时刻的加速度自相关函数，ln 为取以 e 为底的对数计算，α 和 δ_a^2 为系统自适应参数，T 为采样间隔。

步骤 7：利用步骤 5 中求得的目标加速度均值和步骤 6 中求得的修正后的系统自适应参数更新步骤 2 中的目标运动模型。

步骤 8：重复步骤 2～步骤 7，直至所有测量数据全部执行完毕，则结束。

利用上面给出的各个步骤，给出 MATLAB 程序来实现利用自适应动力学模型的估计方法。先来看一下需要调用的函数：myStarmodel 是在已知采用周期 T、机动频率 a 以及过程噪声方差 qq 的基础上，求系统参数 A、Q、U 的函数，myStarmodel 是一个三阶的过程模型函数。

```
function [A,Q,U]=myStarmodel(T,a,qq)

A=[1 T (a*T-1+exp(-a*T))/a/a;0 1 (1-exp(-a*T))/a;0 0 exp(-a*T)];

q11=(1-exp(-2*a*T)+2*a*T+2*(a^3)*(T^3)/3-2*a^2*T^2-4*a*T*exp(-a*T))/
2/(a^5);
q12=(exp(-2*a*T)+1-2*exp(-a*T)+2*a*T*exp(-a*T)-2*a*T+a^2*T^2)/2/(a^4);
q13=(1-exp(-2*a*T)-2*a*T*exp(-a*T))/2/(a^3);
q22=(4*exp(-a*T)-3-exp(-2*a*T)+2*a*T)/2/(a^3);
q23=(exp(-2*a*T)+1-2*exp(-a*T))/2/(a^2);
q33=(1-exp(-2*a*T))/2/a;

Q=qq*[q11 q12 q13;q12 q22 q23;q13 q23 q33];
```

```
U=[(-T+T^2/2+(1-exp(-a*T))/a)/a;T-(1-exp(-a*T))/a;1-exp(-a*T)];
```

因为需要考虑输入项，所以 Kalman 滤波器也和原来略有差别，本书使用如下函数：

```
function [xe,xee,pk]=mykalmanadfun(A,U,C,Q,R,xe,y,p,Ea)

   xee=A*xe+U*Ea;
   p=A*p*A'+Q;
   l=p*C'*inv(C*p*C'+R);
   xe=xee+l*(y-C*xe);
   pk=inv(inv(p)+C'*inv(R)*C);
```

本书用 StartrackingModel 来实现自适应模型，完成一边估计参数一边给出系统的状态估计结果，函数的输入变量分别为含有测量噪声的传感器数据 y、机动频率初值 a、过程噪声方差初值 qq、采样周期 T、系统测量噪声方差 R、测量矩阵 C。在函数中首先设置待估计状态的初值为全 0 的 3 行 1 列向量，估计方差的初值为单位矩阵，一般可以把这个单位矩阵的对角元素设置大些，在程序中用 1000 乘以这个单位矩阵。程序的输出是将每一个采样点的计算值按照时间顺序以序列的形式输出，包括自相关函数序列 RR0、向前一步自相关函数序列 RR1、状态估计结果序列 xx1、向前一步状态估计结果序列 xxe1、基于每一步估计结果计算出的系统参数，如机动频率序列 aa1、过程噪声方差序列 qq1、加速度估计均值序列 Ea1，因此在程序循环之前先设置这些量为空矩阵。

接下来进入循环，依次计算基于每一个采样测量值的状态估计以及相应的参数估计结果。首先根据第一个测量值估计状态，然后根据估计结果计算均值和两个自相关函数，继而计算系统参数的估计值，包括机动频率 a、过程噪声方差 qq，然后进行下一次循环。当然在下一次计算之前，需要把计算得到的值排序到列的尾部，例如，RR0=[RR0 R0]就是将计算得到的 R0 数值放在原来矩阵的最后一列，这样每计算出一个新值就将其放在最后一列。

在计算的过程发现了这样的数值问题，由于估计的结果带有一定误差，使 b=R1/R0 有可能计算为负值，在接下来计算机动频率 a 的求取对数过程中，就没有办法计算下去了。因此在程序中有一个判断，就是当 R1/R0 的比值为正值时再更新 b 的值，否则就不更新，认为这是由计算误差导致系统出现的数值问题。另一个数值问题是递推运算过程中如果 δ_a^2 过大会导致程序发散。在程序中使用 2*a*qb/(1–b*b)<100000 来约束 δ_a^2 的值（2*a*qb/(1–b*b)就是 δ_a^2 ），来保证递推计算的收敛。

```
function [RR0,RR1,xx1,xxe1,aa1,qq1,Ea1]=StartrackingModel(y,a,qq,T,R,C)

xe=zeros(3,1);p=1000*eye(3);
xx1=[];aa1=[];qq1=[];Ea1=[];RR0=[];RR1=[];
xxe1=[];
```

```
for i=1:1  %从第 0 第 1 步
Ea=xe(3);
[A,Q,U]=myStarmodel(T,a,qq);
[xe,xee,p]=mykalmanadfun(A,U,C,Q,R,xe,y(i),p,Ea);
xx1=[xx1 xe];
xxe1=[xxe1 xee];
qsum=0;
%得到第 1 步的均值和两个自相关函数。
Ea=xee(3);
R0=xee(3)*xee(3);
R1=xee(3)*0;
end

for i=2:length(y)
    if i>4
if R1/R0>0
    b=R1/R0;

     qb=(R0-b*R1);
a=-log(b)/T;
if 2*a*qb/(1-b*b)<100000;
qq=2*a*qb/(1-b*b);
    end
end
    end
    [A,Q,U]=myStarmodel(T,a,qq);
[xe,xee,p]=mykalmanadfun(A,U,C,Q,R,xe,y(i),p,Ea);
  Ea=((i-1)/i)*Ea+xee(3)/i;
 R0=R0+(10*xee(3)*xee(3)-R0)/i;
    R1=R1+(10*xee(3)*xxe1(3,i-1)-R1)/i;

RR0=[RR0 R0];
RR1=[RR1 R1];
xx1=[xx1 xe];
xxe1=[xxe1 xee];
aa1=[aa1 a];
qq1=[qq1 qq];
Ea1=[Ea1 Ea];
End
```

用下面这段程序调用上述函数，实现对 target.mat 模拟目标运动轨迹数据的跟踪。

```
clc
clear
```

```
summ=0;N=10;covvv=[];
for n=1:N

%对模拟轨迹 target 进行跟踪。

 load mytarget
%设置采样周期、测量方程的参数。
T=0.1;
R=25;
C=[1 0 0];

y=xys+sqrt(R)*randn(size(xys));
%估计横轴，设置模型参数初值。
 a=1/20;
 xamax=5;
 qq=(xamax)^2*(4-pi)/pi;

[RR0,RR1,xx1,xxe1,aa1,qq1,Ea1]=StartrackingModel(y(1,:),a,qq,T,R,C);
%由于在估计横轴的时候，模型参数的初值发生了变化，因此，在估计纵轴的时候，再次设置
%其初值。
a=1/20;
 xamax=5;
 qq=(xamax)^2*(4-pi)/pi;
[RR02,RR12,xx2,xxe2,aa2,qq2,Ea2]=StartrackingModel(y(2,:),a,qq,T,R,C);

%估计结束后，计算方差并画在结果图。
covv=diag(cov(xys'-[C*xx1;C*xx2]'))
covvv=[covvv covv];
summ=summ+covv;
end
summ/N

plot(xys(1,:),xys(2,:),'b-.');hold on
plot(C*xx1,C*xx2,'--');hold off
legend('参考轨迹','估计轨迹')
 figure
 subplot(2,1,1),plot(ts,xys(1,:),'-',ts,C*xx1,'--')
 legend('参考轨迹','估计轨迹')
 xlabel('采样时刻'),ylabel('横轴')
 subplot(2,1,2),plot(ts,xys(2,:),'-',ts,C*xx2,'--')
legend('参考轨迹','估计轨迹')
xlabel('采样时刻'),ylabel('纵轴')
```

```
figure
subplot(2,1,1),plot(ts,xys(1,:)-C*xx1)
xlabel('采样时刻'),ylabel('横轴')
subplot(2,1,2),plot(ts,xys(2,:)-C*xx2)
xlabel('采样时刻'),ylabel('纵轴')
figure
subplot(3,1,1),plot(ts(2:length(ts)),aa1)
xlabel('(a)')
subplot(3,1,2),plot(ts(2:length(ts)),qq1)
xlabel('(b)')
subplot(3,1,3),plot(ts(2:length(ts)),Ea1)
xlabel('(c)')
figure
subplot(3,1,1),plot(ts(2:length(ts)),aa2)
xlabel('(a)')
subplot(3,1,2),plot(ts(2:length(ts)),qq2)
xlabel('(b)')
subplot(3,1,3),plot(ts(2:length(ts)),Ea2)
xlabel('(c)')
```

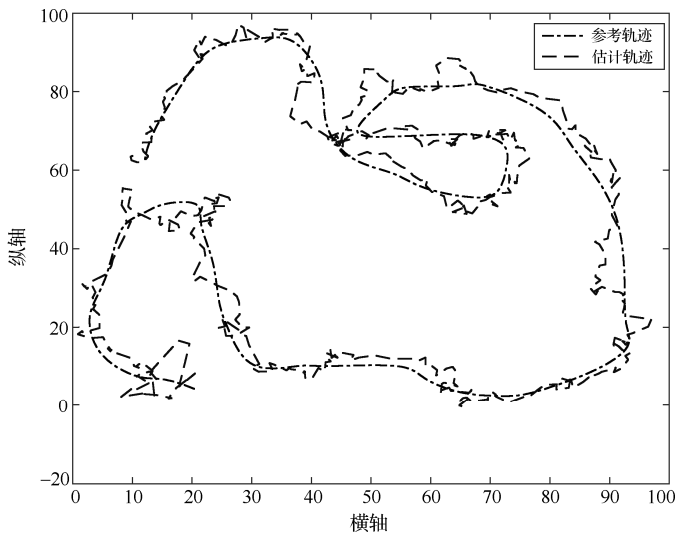
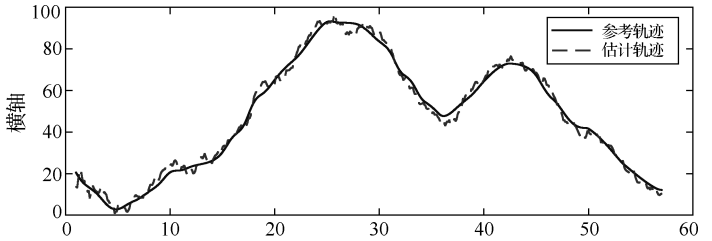


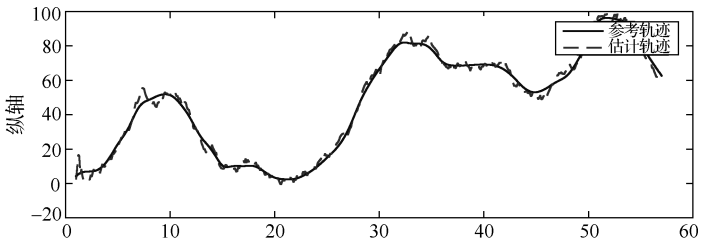
图 7-20　估计结果和实际轨迹进行比较

接下来把自适应模型和前面使用过的 CV 模型、CA 模型、Singer 模型和当前统计模型进行比较。对于 CV 模型和 CA 模型，将过程噪声方差设置为 $Q=1$，对于 Singer 模型，将过程噪声方差和机动频率分别设置为 $Q=1$ 和 $\alpha=1/20$。经过 100 次 Monte Carlo 仿真，计算上述方法估计的位移的均方差（Root-Mean Square Errors，RMSE）然后进行比较。因为当前统计模型对设置参数比较敏感，所以设置了多组 α 和 $\alpha_{\max}$ 参

数。表 7-2 给出了比较结果，发现在使用不同参数时，当前统计模型的估计结果会有很大不同，当前统计模型的参数选择很合适时，方差会很小，如果选择不合适的参数，方差就会比较大。与之相比，自适应模型方差会更小一些。

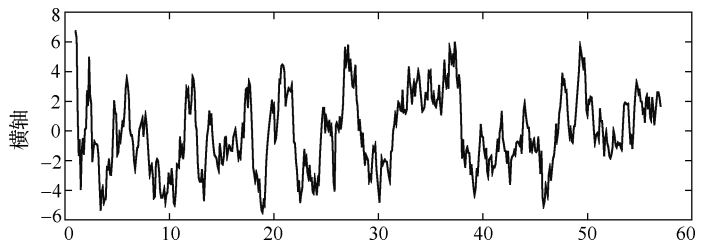


(a) 横轴结果

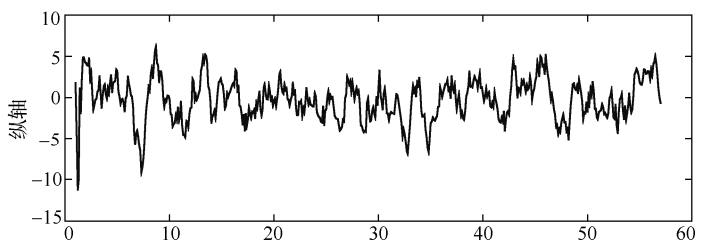


(b) 纵轴结果

图 7-21　估计结果和实际轨迹的横轴、纵轴比较



(a) 横轴结果



(b) 纵轴结果

图 7-22　估计结果和实际轨迹的横轴、纵轴之差

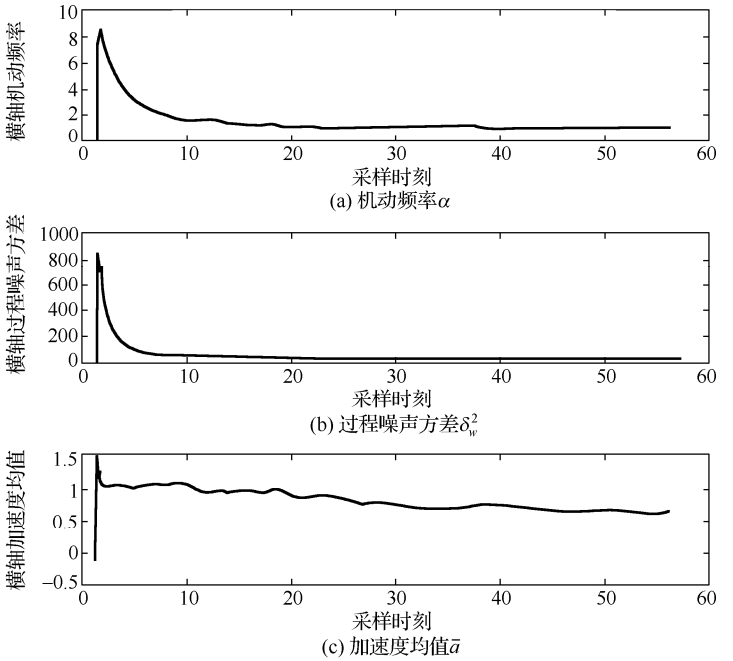


图 7-23　横轴估计参数

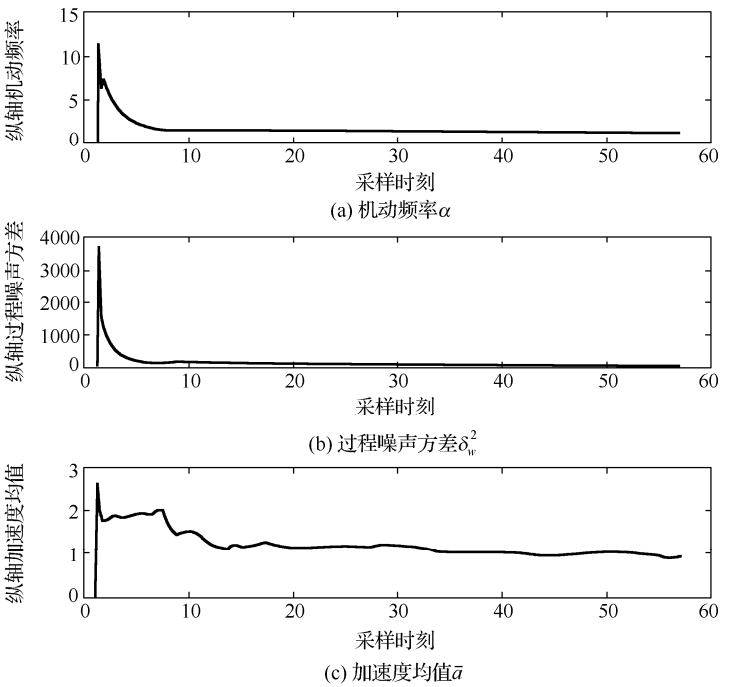


图 7-24　纵轴估计参数

但是在仿真的过程当中也发现，使用自适应模型有时会产生状态估计偏差和参数估计偏差相互影响的偏差累计现象，会使估计方差比较大。这时候特点为，加速度的估计均值趋于 0，而噪声方差估计值 qq 会趋于某一恒值，将其称为估计过程的“偏差失调”现象，这时候的估计结果是没有意义的。

表 7-2　各种估计模型比较的结果

模型类型	自适应模型	CV 模型	CA 模型	Singer 模型	当前统计模型		
					$\alpha=1/30$ $\alpha_{\max}=3$	$\alpha=1/20$ $\alpha_{\max}=30$	$\alpha=1/20$ $\alpha_{\max}=3$
横轴	7.3284	7.4468	5.5662	9.0224	14.105	6.2168	15.817
纵轴	7.0634	18.8009	16.9470	21.8765	5.1523	5.893	5.8298

7.9　本 章 小 结

本章介绍了 CV 模型、CA 模型、Singer 模型、Jerk 模型及 IMM 模型，这些模型的共同特点是需要对目标的机动特性进行先验假设，这其实是离线模型辨识的问题。对于有限类目标来讲也是非常复杂的，随着目标跟踪应用背景的不断扩大，面对各种大量一无所知的目标，由于先验知识的缺乏，这些基于假设的模型一般很难得到较好的跟踪效果。在这些机动模型研究的基础上，可以试着讨论以下几个问题。

既然已经有许多机动模型研究成果，研究另外的建模方法有必要吗？作为武器打击为研究目的的军用背景从 20 世纪 70 年代以来极大地推进了机动目标的研究，利用雷达跟踪战机的机动，迅速而准确地打击要求机动目标研究必须立足于准确的目标模型和能够快速跟踪的方法上，由此，人们研究了战机等目标的机动特征并得到了相关动力学描述模型，研究成果最大的成就在于特定类型目标的相关模型参数的先验值方面。近年来，机动目标跟踪背景发生了很大的变化，部分民用背景跟踪目标的出现大大增加了机动目标的类型，目标的机动特性与原军事背景广泛研究的战机运动有很大的区别，如视频跟踪中汽车、人及物联网中的被跟踪物品的运动特性等。对于大规模出现的民用目标，研究其准确的先验知识几乎不可能。因此，研究不再基于目标先验知识假设的机动目标建模方法是十分有必要的。

另外，这些模型的加速度一般为随机过程分布，而不是具有突变的确定数据，由于确定过程与随机过程具有完全不同的数学特性，因此研究具有随机过程特性加速度的建模方法对提高机动目标跟踪性能将具有极大的意义。

既然不再刻意于研究目标的先验知识，那么还有什么可以利用来得到目标的运动模型呢？在只能得到测量数据的条件下，有效利用测量数据是首先想到的方法。因此，

给出基于数据驱动的建模与跟踪方法近年来已成为一个热点研究方向。这种方法基于“数据中含有模型而模型融于数据中”的基本思想，理论上只要存在机动目标的运动测量数据就可以得到目标模型并进行跟踪，根据第 6 章的内容可以知道，得到的数据中含有系统噪声，需要利用准确的系统模型，才能在存在噪声的情况下得到状态的最优估计，但目前系统模型参数也需要进行估计，因此，整个系统同时存在着两个估计过程，如何保证整个系统在两个估计过程下收敛是这种数据驱动建模方法的关键，给出了保证闭环估计系统的必要条件。

从理论研究角度，这种基于数据的自适应模型方法其实才刚刚开始，还有很多值得探讨的地方，例如，如何给出稳定的估计结果，如何避免“偏差失调”现象等。

第 8 章　基于 RFID 的室内跟踪系统仿真研究

在室内，RFID 是一种很典型的定位技术。定位，顾名思义，就是知道目标的位置在哪里。而跟踪系统则不同，是根据各个时刻的位置测量信息知道目标在每一时刻的位置。在跟踪定位系统中，跟踪方法往往是定位方法的进一步处理，能够勾勒出目标的运动轨迹，甚至下一时刻可能的位置，研究者进而还可以根据目标的运动轨迹来分析目标的行为是否有“不良”企图等。

RFID、Wi-Fi 等定位技术得到的是目标的位置，但给出的不是目标的准确位置，而是目标所在的区域，具有一定的误差。若想求取目标的运动轨迹、并预测下一个出现的位置，研究动力学模型和估计方法是十分必要的。

本章在第 6～7 章的基础上，研究 RFID 测量系统的跟踪方法。先简述 RFID 跟踪系统的特点，进而提出一种用于室内 RFID 跟踪的递归估计方法，包括以下内容。

（1）根据目标的加速度特征，建立在线实时模型。

（2）将 RFID 测量系统的不规则采样时间间隔转化为模型的时变参数。

（3）对多个 RFID 阅读器的测量数据进行融合估计。

同时，在 EKF 和 UKF 的基础上分别给出两种估计方法，并对跟踪性能进行比较。通过仿真结果可知，该方法可以有效改善室内 RFID 系统的跟踪性能，即使在低检测区也可以得到比较好的跟踪结果。

8.1　RFID 跟踪系统的特点

在许多大型室内跟踪系统中，如物品跟踪、工业自动化、商品管理和卫生保健系统，位置作为非常重要的信息被频繁使用。近年来，RFID 技术由于可以获取距离信息而广泛应用于室内跟踪系统。

RFID 通过电磁将数据传输到 RF 兼容集成电路而实现存储和检索。RFID 基本上由两部分组成：阅读器和标签。一旦标签靠近阅读器，就可以通过接收到的信号强度信息（RSSI）提取标签和阅读器间的距离，并将标签携带信息以及与阅读器的距离信息发送至数据处理中心。

很多研究者提到使用“网格”结构（见图 8-1），采用几何方法中的三边测量法放置阅读器，该结构可以简单、有效地实现基于三角形测量数据的定位。然而实际系统中通常无法将阅读器放置成理想的规则结构。例如，在超市中，阅读器会不规则地放

置于顾客感受不到的地方（见图 8-2）。因此，在实际室内跟踪系统中，阅读器是不均匀分布的，当目标进入阅读器密集的区域，许多阅读器同时获得信息，而当目标进入阅读器稀疏的区域时，就会出现没有阅读器报出目标位置的现象。另外，由于测量环境因素还会导致 RSSI 测量不准确。

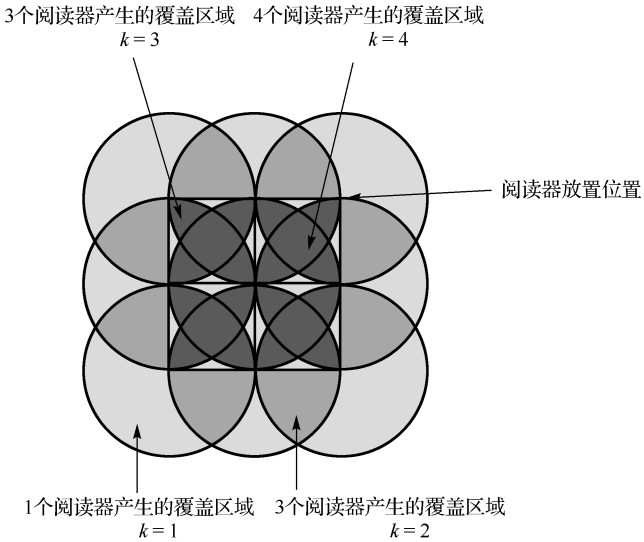


图 8-1　RFID 阅读器网络的方格布局

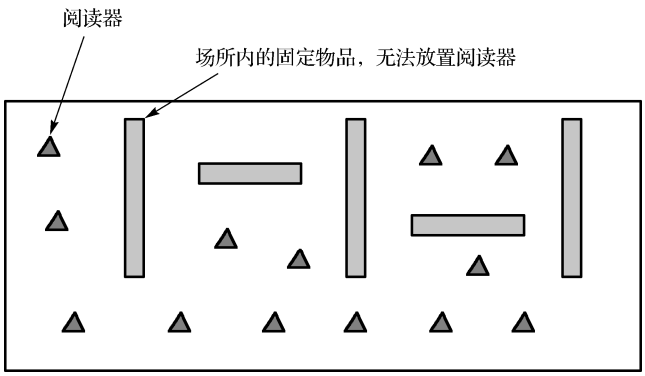


图 8-2　有 RFID 阅读器的实际室内跟踪区域

8.2　不规则采样系统的模型转化

RFID 通过电磁传输到 RF 兼容集成电路的方式存储、检索数据。一旦标签靠近阅读器，就可以得到阅读器和标签之间的距离，并将距离信息发送到数据处理中心。

因为数据驱动测量机制，RFID 的测量没有固定的时间采样点，是一种典型的不规则采样。

前面已经讲过，“当前统计模型”具有一定的代表性，是自适应模型的基础，比 Singer 模型更适用于描述机动目标。下面就以“当前统计模型”为例，讨论具有不规则采样时刻的系统动力学模型，这也是本章提出的 RFID 跟踪系统所要采用的基础模型。设连续时间模型的状态空间可以表示如下：

$$\dot{\boldsymbol{x}}(t)=\boldsymbol{A}\boldsymbol{x}(t)+\boldsymbol{U}\bar{a}(t)+\boldsymbol{B}w(t) \tag{8-1}$$

$$\boldsymbol{A}=\begin{bmatrix}0&1&0\\0&0&1\\0&0&-\alpha\end{bmatrix},\quad \boldsymbol{U}=\begin{bmatrix}0\\0\\\alpha\end{bmatrix},\quad \boldsymbol{B}=\begin{bmatrix}0\\0\\1\end{bmatrix} \tag{8-2}$$

式中，$w(t)$ 由过程噪声决定，其给定的协方差矩阵为 $w(t)\sim N(0,2\alpha\delta_a^2)$。假设测量数据在采样时间 t_i 获得，其测量方程式如下：

$$\boldsymbol{z}(t_i)=\boldsymbol{H}(t_i)\boldsymbol{x}(t_i)+v(t_i),\quad i=0,1,2,\cdots \tag{8-3}$$

式中，$\boldsymbol{H}(t_i)$ 是测量矩阵，$\boldsymbol{v}(t_i)$ 是方差 R 已知的测量噪声，如 $v(t_i)\sim N(0,R)$。

根据微分方程（8-1）可以得到

$$\boldsymbol{x}(t)=\mathrm{e}^{\boldsymbol{A}(t-t_0)}\boldsymbol{x}(t_0)+\int_{t_0}^{t}\mathrm{e}^{\boldsymbol{A}(t-\lambda)}\boldsymbol{U}\bar{a}(\lambda)\mathrm{d}\lambda+\int_{t_0}^{t}\mathrm{e}^{\boldsymbol{A}(t-\lambda)}\boldsymbol{B}w(\lambda)\mathrm{d}\lambda \tag{8-4}$$

可以看到任何已知积分区间$[t_0,\ t]$，如果状态初始值 $\boldsymbol{x}(t_0)$ 已知，参数 $\boldsymbol{A}$、$\boldsymbol{U}$、$\boldsymbol{B}$、$\bar{a}(t)$、$w(t)$ 已知，那么可以得到任意时刻的 $\boldsymbol{x}(t)$ 值。

考虑从 t_i 到 t_{i-1} 的时间间隔并假设：

$$\bar{a}(\lambda)=\bar{a}(t_{i-1}),\ w(\lambda)=w(t_{i-1}),\quad \lambda\in[t_{i-1},t_i]$$

则可得

$$\int_{t_{i-1}}^{t_i}\mathrm{e}^{\boldsymbol{A}(t_i-\lambda)}\boldsymbol{U}\bar{a}(\lambda)\mathrm{d}\lambda=\int_{t_{i-1}}^{t_i}\mathrm{e}^{\boldsymbol{A}\,(t_i-\lambda)}\boldsymbol{U}\mathrm{d}\lambda\ \bar{a}(t_{i-1})$$

$$\int_{t_{i-1}}^{t_i}\mathrm{e}^{\boldsymbol{A}(t_i-\lambda)}\boldsymbol{B}w(\lambda)\mathrm{d}\lambda=\int_{t_{i-1}}^{t_i}\mathrm{e}^{\boldsymbol{A}(t_i-\lambda)}\boldsymbol{B}\mathrm{d}\lambda w(t_{i-1})$$

设置 $\mathrm{th}_i=t_i-t_{i-1}$，得到系统矩阵为 $\mathrm{th}_i=t_i-t_{i-1}$，$\boldsymbol{U}_d(t_{i-1})=\int_{t_{i-1}}^{t_i}\mathrm{e}^{\boldsymbol{A}\,(t_i-\lambda)}\boldsymbol{U}\mathrm{d}\lambda$，噪声为 $\boldsymbol{w}_d(t_{i-1})=\int_{t_{i-1}}^{t_i}\mathrm{e}^{\boldsymbol{A}(t_i-\lambda)}\boldsymbol{B}\mathrm{d}\lambda\boldsymbol{w}(t_{i-1})$，其协方差为 $\boldsymbol{Q}_d(t_{i-1})=E[\boldsymbol{w}_d(t_{i-1})\boldsymbol{w}_d^{\mathrm{T}}(t_{i-1})]$。

由于式（8-2）中的过程矩阵 $\boldsymbol{A}$ 不是全秩矩阵，不能通过拉格朗日−埃尔米特插值方法计算矩阵指数 $\mathrm{e}^{\boldsymbol{A}\mathrm{th}_i}$。在这里采用拉普拉斯转换得到

$$(s\boldsymbol{I}-\boldsymbol{A})^{-1}=\begin{bmatrix}s & -1 & 0\\ 0 & s & -1\\ 0 & 0 & s+\alpha\end{bmatrix}^{-1}=\begin{bmatrix}\dfrac{1}{s} & \dfrac{1}{s^2} & \dfrac{1}{s^2(s+\alpha)}\\ 0 & \dfrac{1}{s} & \dfrac{1}{s(s+\alpha)}\\ 0 & 0 & \dfrac{1}{s+\alpha}\end{bmatrix} \tag{8-5}$$

矩阵指数 $\mathrm{e}^{\boldsymbol{A}\mathrm{th}_i}$ 可以通过拉普拉斯逆变换获得

$$\boldsymbol{A}_d(t_{i-1})=\begin{bmatrix}1 & \mathrm{th}_i & \dfrac{\alpha\mathrm{th}_i-1+\mathrm{e}^{-\alpha\mathrm{th}_i}}{\alpha^2}\\ 0 & 1 & \dfrac{1-\mathrm{e}^{-\alpha\mathrm{th}_i}}{\alpha}\\ 0 & 0 & \mathrm{e}^{-\alpha\mathrm{th}_i}\end{bmatrix} \tag{8-6}$$

同样地，可以得到系统参数为

$$\boldsymbol{U}_d(t_{i-1})=\begin{bmatrix}\dfrac{1}{\alpha}\left(-\mathrm{th}_i+\dfrac{\alpha\cdot\mathrm{th}_i^2}{2}+\dfrac{1-\mathrm{e}^{-\alpha\cdot\mathrm{th}_i}}{\alpha}\right)\\ \mathrm{th}_i-\dfrac{1-\mathrm{e}^{-\alpha\cdot\mathrm{th}_i}}{\alpha}\\ 1-\mathrm{e}^{-\alpha\cdot\mathrm{th}_i}\end{bmatrix} \tag{8-7}$$

方差 $\boldsymbol{w}_d(t_{i-1})$ 为

$$\boldsymbol{Q}_d(t_{i-1})=E[\boldsymbol{w}_d(t_{i-1})\boldsymbol{w}_d^{\mathrm{T}}(t_{i-1})]=2\alpha\delta_\alpha^2\begin{bmatrix}q_{11} & q_{12} & q_{13}\\ q_{12} & q_{22} & q_{23}\\ q_{13} & q_{23} & q_{33}\end{bmatrix} \tag{8-8}$$

其参数为

$$\begin{cases}q_{11}=\dfrac{1}{2\alpha^5}\left[1-\mathrm{e}^{-2\alpha\cdot\mathrm{th}_i}+2\alpha\cdot\mathrm{th}_i+\dfrac{2\alpha^3\mathrm{th}_i^3}{3}-2\alpha^2\mathrm{th}_i^2-4\alpha\cdot\mathrm{th}_i\mathrm{e}^{-\alpha\cdot\mathrm{th}_i}\right]\\ q_{12}=\dfrac{1}{2\alpha^4}[\mathrm{e}^{-2\alpha\cdot\mathrm{th}_i}+1-2\mathrm{e}^{-\alpha\cdot\mathrm{th}_i}+2\alpha\cdot\mathrm{th}_i\mathrm{e}^{-\alpha\cdot\mathrm{th}_i}-2\alpha\cdot\mathrm{th}_i+\alpha^2\mathrm{th}_i^2]\\ q_{13}=\dfrac{1}{2\alpha^3}[1-\mathrm{e}^{-2\alpha\cdot\mathrm{th}_i}-2\alpha\cdot\mathrm{th}_i\mathrm{e}^{-\alpha\cdot\mathrm{th}_i}]\\ q_{22}=\dfrac{1}{2\alpha^3}[4\mathrm{e}^{-\alpha\cdot\mathrm{th}_i}-3-\mathrm{e}^{-2\alpha\cdot\mathrm{th}_i}+2\alpha\cdot\mathrm{th}_i]\\ q_{23}=\dfrac{1}{2\alpha^2}[\mathrm{e}^{-2\alpha\cdot\mathrm{th}_i}+1-2\alpha\cdot\mathrm{th}_i]\\ q_{33}=\dfrac{1}{2\alpha}[1-\mathrm{e}^{-2\alpha\cdot\mathrm{th}_i}]\end{cases} \tag{8-9}$$

然后，可以得到跟踪系统的离散状态空间模型：

$$\begin{cases} \boldsymbol{x}(t_i) = \boldsymbol{A}_d(t_{i-1})\boldsymbol{x}(t_{i-1}) + \boldsymbol{U}_d(t_{i-1})\overline{a}(t_{i-1}) + w_d(t_{i-1}) \\ \boldsymbol{z}(t_i) = \boldsymbol{H}(t_i)\boldsymbol{x}(t_i) + v(t_i) \end{cases} \tag{8-10}$$

$\boldsymbol{x} = [x, \dot{x}, \ddot{x}]^{\mathrm{T}}$ 是系统待估计状态，并且其初始均值方差为 $\boldsymbol{x}_0$ 和 $\boldsymbol{P}_0$，$w_d(t_i)$ 和 $v(t_i)$ 为零均值白噪声且独立于初始状态 $\boldsymbol{x}_0$，$\boldsymbol{z}(t_i)$ 为测量向量，$\boldsymbol{H}(t_i)$ 为测量矩阵，$v(t_i)$ 是方差为 R 的测量噪声。现在，不规则采样转化为动态参数系统，可以看到等采样间隔仅仅是不规则采样问题的一种特例。

8.3　RFID 系统模型

如前所述，系统模型对 Kalman 滤波器的性能影响很大，本节先来讨论 RFID 跟踪系统的模型，包括测量模型和机动目标模型。RFID 系统的测量模型是一个典型的非线性模型，本书会简单地给出一些参数的设置方法。RFID 跟踪系统的机动目标模型，选用第 7 章给出的自适应运动模型。

8.3.1　RFID 测量模型

在重构跟踪系统描述模型之前，需要先讨论 RFID 测量方面的知识，t_i 时刻第 n 个阅读器与标签的距离 $z_n(t_i)$ 可以通过 RSSI 方法获得，其值为 $P_n(d, \phi, t_i)$。

$$z_n(t_i) = d_0 10^{\frac{P_r(d_0) - P_n(d,\phi,t_i)}{10q}} \tag{8-11}$$

式中，d_0 表示与闭环相关的距离参数，$P_r(d_0)$ 表示与 d_0 有关并以 dBm 为单位的参数，q 为路径代价指数。

设 $x, \dot{x}, \ddot{x}$ 代表目标的位置、速度、加速度的横轴状态信息，$y, \dot{y}, \ddot{y}$ 是目标的纵轴状态信息，时刻系统的状态变量可以描述为

$$\boldsymbol{x}(t_i) = [x(t_i) \quad \dot{x}(t_i) \quad \ddot{x}(t_i) \quad y(t_i) \quad \dot{y}(t_i) \quad \ddot{y}(t_i)]$$

又设 $d_n(t_i)$ 表示第 n 个阅读器与标签在采样时间为 t_i 时的实际距离，则该实际距离为

$$d_n(t_i) = \sqrt{[x(t_i) - x_n(0)]^2 + [y(t_i) - y_n(0)]^2} \tag{8-12}$$

式中，$x_n(0)$、$y_n(0)$ 分别代表 RFID 阅读器的横、纵坐标，$x(t_i)$、$y(t_i)$ 为二维空间中目标的实际位置。

通常，距离 $z_n(t_i)$ 以及实际距离 $d_n(t_i)$ 是不相同的，测量误差可以描述为

$$z_n(t_i) = d_n(t_i) + v_n(t_i) \tag{8-13}$$

$v_n(t_i)$ 是第 n 个阅读器在采样时间 t_i 时的测量噪声，其方差满足

$$v_n(t_i)/d_n(t_i) \sim N\left(0,\left(\frac{0.2303\sigma_p}{\gamma}\right)^2\right)$$

式中，σ_p 为标准差，γ 是路径损耗指数，$d_n(t_i)$ 为目标与第 n 个阅读器间的距离，‘/’表示除法运算。可以看出，测距误差与实际距离的比值服从零均值、标准差为 $0.2303\sigma_p/\gamma$ 的正态分布。接收功率的标准偏差可以被固定在 4dBm，这也是无线通信报告中的标准差均值。实际测量路径损耗指数 γ 通常设置为 1.6～6.5。

8.3.2　机动目标运动模型

接下来，讨论不规则采样下 RFID 跟踪系统目标运动的动力学模型。本书选择利用第 7 章机动目标动力学模型中的自适应模型，但做如下改动：假设在 t_i 时刻加速度均值为 0，原因在于室内目标模型的加速度均值在经过起初的波动后经常收敛于零。则得到，不规则采样间隔下的系统模型为

$$\boldsymbol{x}(t_i)=\boldsymbol{A}(t_{i-1})\boldsymbol{x}(t_{i-1})+\boldsymbol{w}(t_{i-1}) \tag{8-14}$$

式中，$\boldsymbol{A}(t_{i-1})=\begin{bmatrix}A_x(t_{i-1}) & 0\\ 0 & A_y(t_{i-1})\end{bmatrix}$ 为状态转移矩阵，$\boldsymbol{w}(t_{i-1})=[w_x(t_{i-1}) \quad w_y(t_{i-1})]^{\mathrm{T}}$ 代表横、纵坐标的过程噪声，并且假设横轴、纵轴的噪声是独立的，则方差矩阵可以定义为 $\boldsymbol{Q}(t_{i-1})=\begin{bmatrix}Q_x(t_{i-1}) & 0\\ 0 & Q_y(t_{i-1})\end{bmatrix}$。

根据第 7 章可知横、纵轴的系统参数为

$$\boldsymbol{A}_\eta(t_{i-1})=\begin{bmatrix}1 & \mathrm{th}_i & \dfrac{\alpha_\eta \mathrm{th}_i-1+\mathrm{e}^{-\alpha_\eta \mathrm{th}_i}}{\alpha^2}\\ 0 & 1 & \dfrac{1-\mathrm{e}^{-\alpha_\eta \mathrm{th}_i}}{\alpha_\eta}\\ 0 & 0 & \mathrm{e}^{-\alpha_\eta \mathrm{th}_i}\end{bmatrix} \tag{8-15}$$

$$\boldsymbol{Q}_\eta(t_{i-1})=2\alpha_\eta\delta_{\alpha\eta}^2\begin{bmatrix}q_{11} & q_{12} & q_{13}\\ q_{12} & q_{22} & q_{23}\\ q_{13} & q_{23} & q_{33}\end{bmatrix} \tag{8-16}$$

$$\begin{cases} q_{11}=\dfrac{1}{2\alpha_\eta^5}\left[1-\mathrm{e}^{-2\alpha_\eta \mathrm{th}_i}+2\alpha_\eta \mathrm{th}_i+\dfrac{2\alpha_\eta^3\mathrm{th}_i^3}{3}-2\alpha_\eta^2\mathrm{th}_i^2-4\alpha_\eta \mathrm{th}_i\mathrm{e}^{-\alpha_\eta \mathrm{th}_i}\right] \\ q_{12}=\dfrac{1}{2\alpha_\eta^4}[\mathrm{e}^{-2\alpha_\eta \mathrm{th}_i}+1-2\mathrm{e}^{-\alpha_\eta \mathrm{th}_i}+2\alpha_\eta T_0\mathrm{e}^{-\alpha_\eta \mathrm{th}_i}-2\alpha_\eta \mathrm{th}_i+\alpha_\eta^2\mathrm{th}_i^2] \\ q_{13}=\dfrac{1}{2\alpha_\eta^3}[1-\mathrm{e}^{-2\alpha_\eta \mathrm{th}_i}-2\alpha_\eta \mathrm{th}_i\mathrm{e}^{-\alpha_\eta \mathrm{th}_i}] \\ q_{22}=\dfrac{1}{2\alpha_\eta^3}[4\mathrm{e}^{-\alpha_\eta \mathrm{th}_i}-3-\mathrm{e}^{-2\alpha_\eta \mathrm{th}_i}+2\alpha_\eta \mathrm{th}_i] \\ q_{23}=\dfrac{1}{2\alpha_\eta^2}[\mathrm{e}^{-2\alpha_\eta \mathrm{th}_i}+1-2\alpha_\eta \mathrm{th}_i] \\ q_{33}=\dfrac{1}{2\alpha_\eta}[1-\mathrm{e}^{-2\alpha_\eta \mathrm{th}_i}] \end{cases} \tag{8-17}$$

式中，$\mathrm{th}_i=t_i-t_{i-1}$ 表示随时间变化的采样间隔，$\eta=x,y$ 分别表示横、纵坐标轴，α_η 表示机动频率，$\delta_{\alpha\eta}^2$ 表示高斯白噪声方差。

假设估计加速度 $\hat{a}(t_i)$ 可以在任意采样时间 t_i 获得，对于稳定的一阶马尔可夫过程，β 和 δ_{aw}^2 的自相关统计学关系可以描述为如下函数：

$$\begin{bmatrix} r(0) & r(1) \\ r(1) & r(0) \end{bmatrix}\begin{bmatrix} 1 \\ -\beta \end{bmatrix}=\begin{bmatrix} \delta_{aw}^2 \\ 0 \end{bmatrix}$$

式中，$r(0)$ 与 $r(1)$ 是加速度的自相关函数：

$$r_i(1)=r_{i-1}(1)+\frac{1}{i}[\hat{a}(t_i)\hat{a}(t_{i-1})-r_{i-1}(1)] \tag{8-18}$$

$$r_i(0)=r_{i-1}(0)+\frac{1}{i}[\hat{a}(t_i)\hat{a}(t_{i-1})-r_{i-1}(0)] \tag{8-19}$$

参数 β 和 δ_{aw}^2 为

$$\beta=\frac{r_i(1)}{r_i(0)} \tag{8-20}$$

$$\delta_{aw}^2=r_i(0)-\alpha r_i(1) \tag{8-21}$$

因此，可以根据公式 $\delta_{aw}^2=\delta_a^2(1-\beta^2)$ 和 $\beta=\mathrm{e}^{-\alpha \mathrm{th}_i}$ 的关系得到机动频率 α 和高斯白噪声方差 δ_a^2。

8.4　基于可变数量 RFID 阅读器的 EKF 跟踪方法

采用泰勒级数法可以使 EKF 实现线性化非线性系统，进而可以利用 Kalman 滤波器获得状态估计。这种方法因为相对简单并且在处理非线性系统方面表现出的效果而引起学者的浓厚兴趣。

基于跟踪算法可以按如下步骤重构 EKF。

（1）状态预测：

$$\hat{\boldsymbol{x}}(t_i \mid t_{i-1}) = \boldsymbol{A}(t_{i-1})\hat{\boldsymbol{x}}(t_{i-1} \mid t_{i-1}) \tag{8-22}$$

$$\boldsymbol{P}(t_i \mid t_{i-1}) = \boldsymbol{A}(t_{i-1})\boldsymbol{P}(t_{i-1} \mid t_{i-1})\boldsymbol{A}^{\mathrm{T}}(t_{i-1}) + \boldsymbol{Q}(t_{i-1}) \tag{8-23}$$

式中，$\boldsymbol{P}(t_i \mid t_{i-1})$ 是状态方差预测值。

（2）状态融合估计：

$$\hat{\boldsymbol{x}}(t_i \mid t_i) = \hat{\boldsymbol{x}}(t_i \mid t_{i-1}) + \sum_{n=1}^{N(t_i)} \boldsymbol{K}_n(t_i)\{\boldsymbol{z}_n(t_i) - \boldsymbol{h}_n[\boldsymbol{x}_n(t_i)]\} \tag{8-24}$$

式中，$\boldsymbol{z}_n(t_i)$ 是通过 RSSI 方法在第 n 个阅读器获得的位置，$\boldsymbol{N}(t_i)$ 为 t_i 时刻可变数量阅读器的数目。

$$\boldsymbol{K}_n(t_i) = \boldsymbol{P}(t_i \mid t_{i-1})\boldsymbol{h}_{nx}^{\mathrm{T}}(t_i)\boldsymbol{S}_n^{-1}(t_i) \tag{8-25}$$

$$\boldsymbol{S}_n(t_i) = \boldsymbol{h}_{nx}(t_i)\boldsymbol{P}(t_i \mid t_{i-1})\boldsymbol{h}_{nx}^{\mathrm{T}}(t_i) + \boldsymbol{R}(t_i) \tag{8-26}$$

$$\boldsymbol{P}^{-1}(t_i \mid t_i) = \boldsymbol{P}^{-1}(t_i \mid t_{i-1}) + \sum_{n=1}^{N(t_i)} \boldsymbol{h}_{nx}^{-1}(t_i)\boldsymbol{R}_n(t_i)\boldsymbol{h}_{nx}(t_i) \tag{8-27}$$

$$\boldsymbol{h}_n[t_i, \hat{\boldsymbol{x}}(t_i \mid t_{i-1})] = \sqrt{[\hat{\boldsymbol{x}}_n(t_i \mid t_{i-1}) - \boldsymbol{x}_n(0)]^2 + [\hat{\boldsymbol{y}}_n(t_i \mid t_{i-1}) - \boldsymbol{y}_n(0)]^2} \tag{8-28}$$

测量函数 $\boldsymbol{h}_{nx}(t_i)$ 的雅可比矩阵为

$$\begin{aligned}\boldsymbol{h}_{nx}(t_i) &= [\nabla_x \boldsymbol{h}_n'(t_i)]_{\boldsymbol{x}=\hat{\boldsymbol{x}}(t_i \mid t_{i-1})} \\ &= \begin{bmatrix} \dfrac{\hat{\boldsymbol{y}}_n(t_i \mid t_{i-1})}{\boldsymbol{h}_n[t_i, \hat{x}(t_i \mid t_{i-1})]} & 0 & 0 & \dfrac{\hat{\boldsymbol{x}}_n(t_i \mid t_{i-1})}{\boldsymbol{h}_n[t_i, \hat{x}(t_i \mid t_{i-1})]} & 0 & 0 \end{bmatrix}\end{aligned} \tag{8-29}$$

（3）参数更新。

加速度估计值 $\hat{a}_\eta(t_i)$ 为 $\hat{a}_x(t_i) = \hat{\dot{x}}(t_i \mid t_i), \hat{a}_y(t_i) = \hat{\dot{y}}(t_i \mid t_i)$，$\eta = x$ 或 y。

当 $i \leqslant K_0$ 时，$\delta_{ai,\eta}^2$ 和 $\alpha_{i,\eta}$ 为

$$\delta_{ai,\eta}^2 = \begin{cases} (4-\pi)[a_M - \hat{a}_\eta(t_i)]^2 / \pi, & \hat{a}_\eta(t_i) > 0 \\ (4-\pi)[\hat{a}_\eta(t_i) - a_M]^2 / \pi, & \hat{a}_\eta(t_i) < 0 \\ \text{比较小的正数}, & \hat{a}_\eta(t_i) = 0 \end{cases}$$

当 K_0 为小于 10 的正数时机动频率 $\alpha_{i,\eta}$ 为正，a_{M} 为正数，$-a_{\mathrm{M}}$ 是 a_{M} 的相反数。

当 $i > K_0$ 时，$\delta_{ai,\eta}^2$ 和 $\alpha_{i,\eta}$ 为

$$r_{i,\eta}(1) = r_{i-1,\eta}(1) + \frac{1}{i}[\hat{a}_\eta(t_i)\hat{a}_\eta(t_{i-1}) - r_{i-1,\eta}(1)] \tag{8-30}$$

$$r_{i,\eta}(0) = r_{i-1,\eta}(0) + \frac{1}{i}[\hat{a}_\eta(t_i)\hat{a}_\eta(t_i) - r_{i-1,\eta}(0)] \tag{8-31}$$

$$\beta_{i,\eta} = \frac{r_{i,\eta}(1)}{r_{i,\eta}(0)} \tag{8-32}$$

$$\delta_{aw\ i,\eta}^2 = r_{i,\eta}(0) - \beta_{i,\eta} r_{i,\eta}(1) \tag{8-33}$$

$$\delta_{a\ i,\eta}^2 = \frac{\delta_{aw\ i,\eta}^2}{1-\beta_{i,\eta}^2} \tag{8-34}$$

$$\alpha_{i,\eta} = \frac{\ln \beta_{i,\eta}}{-\mathrm{th}_i} \tag{8-35}$$

根据 $\delta_{ai,\eta}^2$ 和 $\alpha_{i,\eta}$，系统矩阵式（8-15）、式（8-16）可以在每个采样时间进行更新。同时，集中融合估计式（8-24）是基于具有可变数目的阅读器，所以所开发的递归估计算法与常规固定数量的融合方法是不同的（具体细节，请读者参阅“信息融合”方面的参考书）。

上述过程比较复杂，给出了两个函数，完成前两步的预测与估计的函数为如下 myEKFadfun。该函数的输入变量为系统模型 A、Q、R，状态估计初值 xe，测量数据 ym，估计方差初值 p 等。mm 为一个向量，表明哪一个 RFID 阅读器给出了测量数据，readerxy 包含了给出数据的阅读器的横、纵坐标位置。

```
function [xe,xee,pk]=myEKFadfun(A,Q,R,xe,ym,mm,p,readerxy)
%本函数用于使用 EKF 方法估计 RFID 系统状态，完成式（8-22）～式（8-29）的运算
    xee=A*xe;
    p=A*p*A'+Q;
   dm=sqrt((xee(1)-readerxy(1,mm)).^2+(xee(4)-readerxy(2,mm)).^2);
    Rm=dm.^2*R;
    for m=1:length(mm)
    Hm(m,:)=[(xee(1)-readerxy(1,mm(m)))/dm(m),0,0,(xee(4)-readerxy
(2,mm(m)))/dm(m),0,0];
    end

    sumHm=0;
    for m=1:length(mm)
```

```
        sumHm=sumHm+Hm(m,:)'*inv(Rm(m))*Hm(m,:);
    end

    pk=inv(inv(p)+sumHm);

    Km=[];
    for m=1:length(mm)
        Km=[Km pk*Hm(m,:)'*inv(Rm(m))];
    end

    sumxe=0;
      for m=1:length(mm)
        sumxe=sumxe+Km(:,m)*(ym(m)-dm(m));
    end
    xe=xee+sumxe;
```

另外一个函数 funDataDrivenModelYWwithEKF 是调用上述函数的，实现的 EKF 估计的整个过程，包括计算不规则采样数据的周期，找出第几个传感器有测量数据，利用自适应模型方法进行横、纵轴估计。该函数除了调用 myEKFadfun 之外，还调用了第 7 章的 myStarmodel 函数，其中 TT 是采样时间间隔序列。

```
function
[xx1,xxe1,P33,NN]=funDataDrivenModelYWwithEKF(TT,R,ax,qqx,ay,qqy,xe,
p,y,N,readerxy)
xx1=[];aa1=[];qq1=[];Ea1=[];RR0=[];RR1=[];P33=[];NN=[];
xxe1=[];
Ea=[xe(3);xe(6)];

MM=3;
[J,I]=size(y);

for i=1:I %循环计算每一个测量数据
%计算不规则采样数据的周期。
    if length(TT)==1
        T=TT;
    else
        T=TT(i);
    end
%找出第几个传感器有测量数据。
ym=[];mm=[];
for j=1:J  %计算第几个传感器有测量数据
   if isnan(y(j,i))
```

```
    else
       ym=[ym;y(j,i)];
       mm=[mm;j];
       end
end

%
if i<=2  %从 0 到 1 步
     [Ax,Qx,Ux]=myStarmodel(T,ax,qqx);
     [Ay,Qy,Uy]=myStarmodel(T,ay,qqy);
     A=blkdiag(Ax,Ay);
     U=blkdiag(Ux,Uy);
     Q=blkdiag(Qx,Qy);

     [xe,xee,p]=myEKFadfun(A,Q,R,xe,ym,mm,p,readerxy);;

%得到第 1 步的均值和两个自相关函数。
     Eax=xe(3);
     R0x=xe(3)*xe(3);
     R1x=xe(3);

     Eay=xe(6);
     R0y=xe(6)*xe(6);
     R1y=xe(6);

     Ea=[Eax;Eay];
     else  %其他步
% x 轴的计算。
      if i>10    %第 10 步以后进行统计计算自相关函数，否则缺少必需的统计数据量。
        bx=R1x/R0x;
        if bx>0
            qbx=(R0x-bx*R1x);
            ax=-log(bx)/T;
            qqx=2*ax*qbx/(1-bx*bx);
             if qqx>10000^2  qqx=10000^2;end  %qq 设置为有界值，如果 qq 发散，
                                              %则系统发散
        end  %如果 b 小于 0，则会出现计算出复数的情况，出现计算问题，不更新模型参
             %数 a 和 qq
        end
% y 轴的计算。
       if i>10  %第 10 步以后进行统计计算自相关函数，否则缺少必需的统计数据量。
```

```
      by=R1y/R0y;
          if by>0
              qby=(R0y-by*R1y);
              ay=-log(by)/T;
              qqy=2*ay*qby/(1-by*by);
          if qqy>10000^2  qqy=10000^2;end   %qq 设置为有界值，若 qq 发散，则
                                            %系统发散
          end   %若 b 小于 0，则会出现计算出复数的情况，出现计算问题，不更新模型
                %参数 a 和 qq
      end

     [Ax,Qx,Ux]=myStarmodel(T,ax,qqx);
     [Ay,Qy,Uy]=myStarmodel(T,ay,qqy);
     A=blkdiag(Ax,Ay);
     U=blkdiag(Ux,Uy);
     Q=blkdiag(Qx,Qy);

     [xe,xee,p]=myEKFadfun(A,Q,R,xe,ym,mm,p,readerxy);

     R0x=R0x+((xe(3))*(xe(3))-R0x)/i;
     R1x=R1x+((xe(3))*(xx1(3,i-1))-R1x)/i;

     R0y=R0y+((xe(6))*(xe(6))-R0y)/i;
     R1y=R1y+((xe(6))*(xx1(6,i-1))-R1y)/i;
end

xx1=[xx1 xe];
xxe1=[xxe1 xee];
P33=[P33 p(3,3)];
NN=[NN N];
end
```

8.5　基于可变数量 RFID 阅读器的 UKF 跟踪方法

基于第 5 章讨论的 UKF 跟踪方法，本书直接给出 RFID 跟踪系统的方法如下。

（1）$\boldsymbol{W}^{(0)}$ 为均值权重，为了保证原始数据集的正态性、平均值和协方差，系统引入了 sigma：

$$\begin{aligned}\boldsymbol{x}^{(0)} &= [x_x^{(0)} \quad x_w^{(0)} \quad x_v^{(0)}]^{\mathrm{T}} \\ &= [\hat{x}(t_{i-1} \mid t_{i-1}) \quad 0_{N_w} \quad 0_{N_v}]^{\mathrm{T}}\end{aligned} \tag{8-36}$$

$$\boldsymbol{W}^{(0)} = \boldsymbol{W}^{(0)} \tag{8-37}$$

$$\begin{aligned}\boldsymbol{x}^{(i)} &= [x_x^{(i)} \quad x_w^{(i)} \quad x_v^{(i)}]^{\mathrm{T}} \\ &= \boldsymbol{x}^{(0)} + \left(\sqrt{\frac{N_x}{1-\boldsymbol{W}^{(0)}}\boldsymbol{P}(t_{i-1}\,|\,t_{i-1})}\right)_i\end{aligned} \tag{8-38}$$

$$\boldsymbol{W}^{(i)} = \frac{1-\boldsymbol{W}^{(0)}}{2N_x} \tag{8-39}$$

$$\begin{aligned}\boldsymbol{x}^{(i+N_x)} &= [x_x^{(i+N_x)} \quad x_w^{(i+N_x)} \quad x_v^{(i+N_x)}]^{\mathrm{T}} \\ &= x^{(0)} - \left(\sqrt{\frac{N_x}{1-\boldsymbol{W}^{(0)}}\boldsymbol{P}(t_{i-1}\,|\,t_{i-1})}\right)_i\end{aligned} \tag{8-40}$$

$$\boldsymbol{W}^{(i+N_x)} = \frac{1-\boldsymbol{W}^{(0)}}{2N_x} \tag{8-41}$$

0_{N_w} 是与过程噪声 $\boldsymbol{w}(t_{i-1})$ 同维度的列向量，0_{N_v} 是与测量噪声 $\boldsymbol{v}(t_{i-1})$ 同维度的列向量，N_x 是系统状态的维度，$\left(\sqrt{\frac{N_x}{1-\boldsymbol{W}^{(0)}}\boldsymbol{P}(t_{i-1}\,|\,t_{i-1})}\right)$ 是矩阵的第 i 列主要平方根。通过过程模型可以实例化每个点，进而得到转化集。

$$\hat{\boldsymbol{x}}^{(i)} = \boldsymbol{A}(t_{i-1})\boldsymbol{x}_x^{(i)} + \boldsymbol{x}_w^{(i)}, \quad i = 0,1,\cdots,2N_x \tag{8-42}$$

（2）预测均值：

$$\hat{\boldsymbol{\mu}} = \sum_{i=0}^{2N_x} \hat{\boldsymbol{x}}^{(i)}\boldsymbol{W}^{(i)} \tag{8-43}$$

（3）预测方差：

$$\boldsymbol{P}(t_i\,|\,t_{i-1}) = \sum_{i=0}^{2N_x}\boldsymbol{W}^{(i)}\left\{\hat{\boldsymbol{x}}^{(i)} - \hat{\boldsymbol{\mu}}\right\}\left\{\hat{\boldsymbol{x}}^{(i)} - \hat{\boldsymbol{\mu}}\right\}^{\mathrm{T}} + \boldsymbol{Q}(t_{i-1}) \tag{8-44}$$

（4）通过计量模型分别给出预测点：

$$\hat{\boldsymbol{z}}_n^{(i)} = \boldsymbol{h}_n[t_i, \hat{\boldsymbol{x}}^{(i)}] + \boldsymbol{x}_v^{(i)} \tag{8-45}$$

（5）观测向量预测：

$$\hat{\boldsymbol{z}}_n = \sum_{i=0}^{2N_x} \hat{\boldsymbol{z}}_n^{(i)}\boldsymbol{W}^{(i)} \tag{8-46}$$

（6）协方差改进：

$$\boldsymbol{S}_n(t_i) = \sum_{i=0}^{2N_x}\boldsymbol{W}^{(i)}\left\{\hat{\boldsymbol{z}}_n^{(i)} - \hat{\boldsymbol{z}}_n\right\}\left\{\hat{\boldsymbol{z}}_n^{(i)} - \hat{\boldsymbol{z}}_n\right\}^{\mathrm{T}} + \boldsymbol{R}_n(t_i) \tag{8-47}$$

（7）交叉协方差矩阵：

$$\boldsymbol{P}_{xz,n}(t_i)=\sum_{i=0}^{2N_x}\boldsymbol{W}^{(i)}\left\{\hat{\boldsymbol{x}}^{(i)}-\hat{\boldsymbol{\mu}}\right\}\left\{\hat{\boldsymbol{z}}_n^{(i)}-\hat{\boldsymbol{z}}_n\right\}^{\mathrm{T}} \tag{8-48}$$

（8）更新：

$$\hat{\boldsymbol{x}}(t_i\mid t_i)=\hat{\boldsymbol{x}}(t_i\mid t_{i-1})+\sum_{n=1}^{N(t_i)}\boldsymbol{K}_n(t_i)\left\{\boldsymbol{z}_n(t_i)-\hat{\boldsymbol{z}}_n\right\} \tag{8-49}$$

$$\boldsymbol{K}_n(t_i)=\boldsymbol{P}_{xz,n}(t_i)\boldsymbol{S}_n^{-1}(t_i) \tag{8-50}$$

估计方差为

$$\boldsymbol{P}(t_i\mid t_i)=\boldsymbol{P}(t_i\mid t_{i-1})-\sum_{n=1}^{N(t_i)}\boldsymbol{K}_n(t_i)\boldsymbol{S}_n(t_i)\boldsymbol{K}_n^{\mathrm{T}}(t_i) \tag{8-51}$$

（9）参数更新。

与第四部分中的步骤 3 相同。

如上述算法所示，（1）～（7）都是基本的 UKF 方法。但是（8）使用了多传感器集中式融合的方法，如式（8-49）中的状态估计更新、式（8-50）中的滤波器增益、式（8-51）中的估计协方差都与基本的 UKF 方法不同。

UKF 估计方法的函数程序调用和输出的量与 myEKFadfun 非常相似，但由于 UKF 计算过程的需要，输入、输出变量中都包含了 sss，表示方差矩阵求开方的结果，其中输入量为前一步的方差矩阵求开方的结果，输出量为当前步的计算结果。

```
function [xe,xee,pk,sss]=myUKFadfun(A,Q,R,xe,ym,mm,p,readerxy,sss)
%本函数完成 UKF 式（8-36）～式（8-51）计算。

mx=6;
Xx=0.01;
Oo=2;
Vv=0;
Kk=-3;
xe=[xe;zeros(size(xe));zeros(size(ym(1)))];
p=blkdiag(p,Q,R);
ssss=sqrtm((mx+Kk)*p); %先对方差矩阵求开方
if isreal(ssss) %若矩阵的方根为实数，则使用该数
     sss=ssss; %否则，不更改由方差矩阵得到的数据
end

    s(:,1)=xe;
```

```
for ss=2:mx+1;
    s(:,ss)=xe+sss(:,ss-1);
    s(:,ss+mx)=xe-sss(:,ss-1);
end

Wx=ones(1,13)/2/(mx+Kk);
Wp=ones(1,13)/2/(mx+Kk);

Wx(1)=Kk/(mx+Kk);
Wp(1)=1-Xx*Xx+Oo+(Kk/(mx+Kk));

xee=0;
for i=1:length(s(1,:))
    xx(:,i)=A*s(1:6,i)+s(7:12,i);
    xee=xee+Wx(i)*xx(:,i);
end

for i=1:length(s(1,:))
    xee1(:,i)=xx(:,i)-xee;
end

p=0;
for i=1:length(s(1,:))
   p=p+Wp(i)*xee1(:,i)*xee1(:,i)';
end
p=p+Q;

lmm=length(mm);

for m=1:lmm
    for i=1:length(s(1,:))
dmn(m,i)=sqrt((xx(1,i)-readerxy(1,mm(m))).^2+(xx(4,i)-readerxy
(2,mm(m))).^2)+s(13,i);
    end
end

dm=0;
for i=1:length(s(1,:))
   dm=dm+Wx(i)*dmn(:,i);
end

for i=1:length(s(1,:))
```

```
   dm1(:,i)=dmn(:,i)-dm;
end

for m=1:lmm
   pxz=zeros(size(Wp(1)*xee1(:,1)*dm1(m,1)));
   for i=1:length(s(1,:))
        pxz=pxz+Wp(i)*xee1(:,i)*dm1(m,i)';
   end
   Pxz(:,m)=pxz;
end

for m1=1:lmm
    for m2=m1
       pzz=0;
       for i=1:length(s(1,:))
         pzz=pzz+Wp(i)*dm1(m1,i)*dm1(m2,i)';
       end
       Rm(m1,m2)=dm(m1)*dm(m2)*R;
       Pzz(m1,m2)=pzz+Rm(m1,m2);
    end
end
K=Pxz*inv(Pzz);
pk=p-K*Pzz*K';
sumxee=0;
for  m=1:lmm
    sumxee=sumxee+K(:,m)*(ym(m)-dm(m));
end
xe=xee+sumxee;
```

和前面的类似，下面的 funDataDrivenModelYWnonparell 函数是调用 myUKFadfun 函数的，实现的 UKF 估计的整个过程，包括计算不规则采样数据的周期，找出第几个传感器有测量数据，利用自适应模型方法进行横、纵轴估计。

```
function
[xx1,xxe1,P33,NN,qqxx,RR0x,RR0y]=funDataDrivenModelYWnonparell(TT,R,
ax,qqx,ay,qqy,xe,p,y,N,readerxy)
xx1=[];qqxx=[];qq1=[];Ea1=[];RR0=[];RR1=[];P33=[];NN=[];
RR0x=[];RR0y=[];
xxe1=[];
Ea=[xe(3);xe(6)];
sss=ones(13);
MM=3;
[J,I]=size(y);
```

```
for i=1:I %设置循环次数
%适用于不规则采样数据的周期。
    if length(TT)==1
        T=TT;
    else
        T=TT(i);
    end

ym=[];mm=[];
for j=1:J  %找出第几个传感器有数据
   if isnan(y(j,i))

   else
       ym=[ym;y(j,i)];
       mm=[mm;j];
       end
end

     if i==1  %从 0 到 1 步
[Ax,Qx,Ux]=myStarmodel(T,ax,qqx);
[Ay,Qy,Uy]=myStarmodel(T,ay,qqy);
A=blkdiag(Ax,Ay);
U=blkdiag(Ux,Uy);
Q=blkdiag(Qx,Qy);

 [xe,xee,p,sss]=myUKFadfun(A,Q,R,xe,ym,mm,p,readerxy,sss);%使用 UKF 方
%法进行非线性估计

%得到第 1 步的均值和两个自相关函数。
Eax=xee(3);
R0x=xee(3)*xee(3);
R1x=xee(3);

Eay=xee(6);
R0y=xee(6)*xee(6);
R1y=xee(6);

Ea=[Eax;Eay];
   else  %其他步
% x 轴。
```

```
    if i>4   %第 4 步以后进行统计计算自相关函数，否则缺少必需的统计数据量
    bx=R1x/R0x;
    if bx>0
        qbx=(R0x-bx*R1x);
ax=-log(bx)/T;

qqx=2*ax*qbx/(1-bx*bx);

 if qqx>10000^2  qqx=10000^2;end  % qq 设置为有界值，若 qq 发散，则系统发散
 end  %若 b 小于 0，则会出现计算出复数的情况，出现计算问题，不更新模型参数 a 和 qq
    end
% y 轴。
    if i>4   %第 4 步以后进行统计计算自相关函数，否则缺少必需的统计数据量
    by=R1y/R0y;
    if by>0
        qby=(R0y-by*R1y);
ay=-log(by)/T;
qqy=2*ay*qby/(1-by*by);
 if qqy>10000^2  qqy=10000^2;end  % qq 设置为有界值，若 qq 发散，则系统发散
 end  %若 b 小于 0，则会出现计算出复数的情况，出现计算问题，不更新模型参数 a 和 qq
    end
    [Ax,Qx,Ux]=myStarmodel(T,ax,qqx);
[Ay,Qy,Uy]=myStarmodel(T,ay,qqy);
A=blkdiag(Ax,Ay);
U=blkdiag(Ux,Uy);
Q=blkdiag(Qx,Qy);


    lmm=length(mm);
[xe,xee,p,sss]=myUKFadfun(A,Q,R,xe,ym,mm,p,readerxy,sss);

 %计算自适应模型。
    R0x=R0x+((xe(3))*(xe(3))-R0x)/i; %若这样取，则利用估计值进行自适应参数计算
    R1x=R1x+((xe(3))*(xx1(3,i-1))-R1x)/i;  %不是并行的

    R0y=R0y+((xe(6))*(xe(6))-R0y)/i;
    R1y=R1y+((xe(6))*(xx1(6,i-1))-R1y)/i;

 end


qqxx=[qqxx qqx];
```

```
RR0x=[RR0x R0x];
RR0y=[RR0y R0y];

xx1=[xx1 xe];
xxe1=[xxe1 xee];
P33=[P33 p(3,3)];
NN=[NN N];
end
```

8.6　仿 真 研 究

仿真实验中，使用 6.4.2 节介绍的 RFID 室内跟踪系统仿真数据平台软件产生仿真数据。初始状态估计 $\boldsymbol{x}_0$ 和协方差 $\boldsymbol{P}_0$ 假设为 $\boldsymbol{x}_0=[x(0)\quad 0\quad 0\quad y(0)\quad 0\quad 0]^{\mathrm{T}}$ 和 $\boldsymbol{P}_0=\mathrm{diag}(10,10,10,10,10,10)^{\mathrm{T}}$。机动频率 α_η 始值设定为 1/20，$a_{\mathrm{M}}=30$，$a_{-\mathrm{M}}=-30$。平均点的初始权重 $W_x^{(0)}=\dfrac{\kappa}{N_x+\kappa}$，尺度参数为 $\kappa=-3$，$N_x=6$。

本书将给出如下两个仿真实验：实验 1 为比较基于 EKF、UKF 的方法的不同跟踪结果；实验 2 为通过与几何方法进行对比说明使用动态模型的必要性。下面给出每一个实验的具体内容及分析结果。

1. 实验 1

在仿真平台中设置了 19 个 RFID 阅读器（坐标点列于表 8-1），每一个阅读器高检测率的测定空间用颜色区分（从内到外的等值线图表示逐渐减弱的测量能力，白色表示无法得到测量数据的区域），目标的参考轨迹在图 8-3 中以“黑”星显示。

表 8-1　RFID 阅读器的坐标点

标号	1	2	3	4	5	6	7	8	9	10
横坐标	22.46	13.94	25.92	43.66	48.27	9.56	11.41	52.41	80.76	68.77
纵坐标	28.80	60.96	83.47	77.63	50.44	37.87	9.79	7.74	16.22	54.53
标号	11	12	13	14	15	16	17	18	19	
横坐标	65.092	74.77	87.90	92.28	91.82	61.17	34.678	5.64	95.05	
纵坐标	78.51	94.59	83.77	59.50	39.03	27.63	60.67	95.17	7.456	

注意到图 8-4 中有两个低检测率区域（用椭圆形的细线圈起来的部分）图 8-2 给出了图 8-4 中从 11.1s 到 12.1s（低探测区域标有“①”的部分）的轨迹信息，表中“—”表示 RFID 阅读器没有测量数据输出。可以看到，表 8-2 中的采样时间是不规则的，从 11.5s 至 11.7s 以及 11.8s 到 12.0s 没有测量数据。

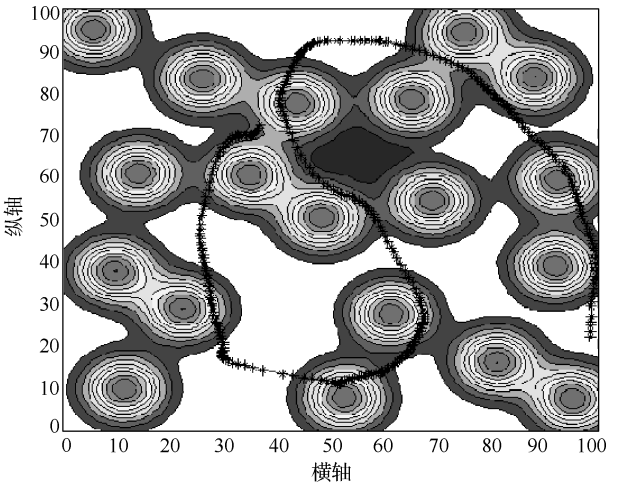


图 8-3　RFID 阅读器的测量空间和参考轨迹

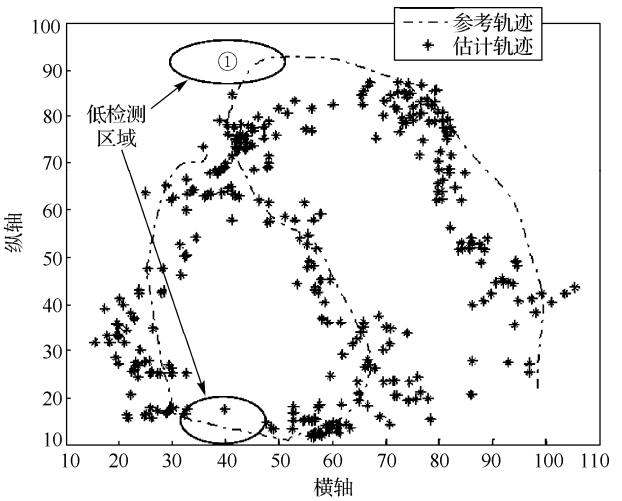


图 8-4　参考轨迹和 UKF 估计轨迹

此外，由于目标远离所有的阅读器，因此在较低的检测率区域内大部分时间里仅测得一个距离数据。已知 RFID 得到的信息是以阅读器为中心的测量距离，如果不进行轨迹估计，是无法仅通过一个测量数据就获得目标的位置及轨迹的。

表 8-2　RFID 阅读器系统的测量

采样时间/s / 阅读器	11.1	11.2	11.3	11.4	11.5	11.7	11.8	12.0	12.1
1	—	—	—	—	—	—	—	—	—
2	—	—	—	—	—	—	—	—	—

续表

采样时间/s / 阅读器	11.1	11.2	11.3	11.4	11.5	11.7	11.8	12.0	12.1
3	—	—	—	—	—	—	—	—	27.629
4	—	20.502	—	—	—	—	18.050	20.269	—
5	—	—	—	—	—	—	—	—	—
6	—	—	—	—	—	—	—	—	—
7	—	—	—	—	—	—	—	—	—
8	—	—	—	—	—	—	—	—	—
9	—	—	—	—	—	—	—	—	—
10	—	—	—	—	—	—	—	—	—
11	17.524	—	—	21.693	—	—	—	—	23.469
12	—	20.297	25.370	—	23.228	—	—	—	27.779
13	—	—	—	—	—	—	—	—	—
14	—	—	—	—	—	—	—	—	—
15	—	—	—	—	—	—	—	—	—
16	—	—	—	—	—	—	—	—	—
17	—	—	—	—	—	37.689	—	—	—
18	—	—	—	—	—	—	—	—	—
19	—	—	—	—	—	—	—	—	—

用 8.4 节给出的 EKF 和 8.5 节给出的 UKF 方法估计图 8-3 中的目标的运动轨迹。如图 8-4 所示，给出了基于 UKF 方法的估计轨迹，此时，横轴的估计协方差为 43.7320、纵轴的估计协方差为 30.1378。可以看到，即使在阅读器数量已经发生很大变化的情况下，通过可变数量阅读器得到的估计也是很平滑，没有突然变化的。此外，在较低检测率区时，目标的大致移动方向仍然可以得到，并且一旦得到足够的测量，估计误差也会迅速降低。

但是发现，在算法的运算过程中，会出现数值计算的问题。$\boldsymbol{P}(t_{i-1}\,|\,t_{i-1})$ 由于计算机的舍入误差而成为负定矩阵，进而矩阵 $\left(\sqrt{\frac{N_x}{1-\boldsymbol{W}^{(0)}}\boldsymbol{P}(t_{i-1}\,|\,t_{i-1})}\right)_i$ 的平方根会出现复数而导致发散的估计结果。这种情况是不正确的，因为估计协方差不能为负定矩阵，这种情况下算法将舍弃当前步骤的计算结果，而采用前一周期得到的 $\delta_{ai,\eta}^2$ 和 $\alpha_{i,\eta}$。

图 8-5 给出了基于 EKF 方法的估计结果，横轴的估计协方差为 49.2714，纵轴的估计协方差为 38.524。与 UKF 不同的是，EKF 在低检测率区域内估计效果不会明显变差，但整体的估计性能不如 UKF，这和很多非线性估计的研究结果一致。

认为由于在 EKF 估计方法中非线性测量模型已经被线性化，所以距离测量模式不会影响估计结果。但因为线性机制造成的测量模型误差导致估计性能下降，因此在 RFID 室内跟踪系统中 EKF 的估计性能比 UKF 的估计性能差。该实验的程序如 C8-1 所示，目前该程序得到的是 EKF 的估计结果，其中中间一段：

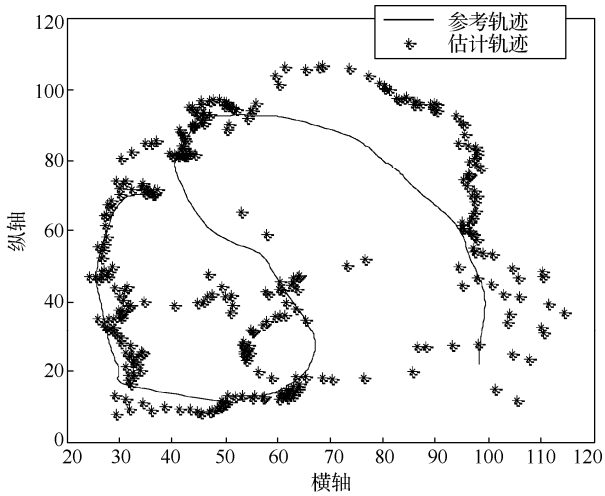


图 8-5　参考轨迹和 EKF 估计轨迹

```
%使用 UKF 计算的方法，可以节约计算时间，估计横轴、纵轴使用最小二乘估计 Yule-Walker
方法。
%[xx1,xxe1,P33,NN,qqxx,RR0x,RR0y]=funDataDrivenModelYWnonparell(TT,
R,ax,qqx,ay,qqy,xe,p,dm,N,readerxy);
```

被注释。若要得到 UKF 的估计结果，将该段取消注释即可。

```
%C8_1.m
clc
clear
summ=0;
N=10;    %计算 10 次，再求平均，需要在 GUI 上设置个可以设置数字的文板框，再把这个数
         %据提出来应用到程序中
covvv=[];    %估计方差
for n=1:N
%不规则采样跟踪开始。
%读取数据。
load RFIDm5
TT=[];TTT=[];  %为记录下面计算得出的量，需要先设置两个空变量
%计算得到不规则采样数据的周期
for i=1:length(dm(1,:))
if i==1
        T=ts(1);tt=0;
         TT=[TT T];TTT=[TTT tt];
   elseif i==2;
        T=ts(i)-ts(i-1);tt=0;
        TT=[TT T];TTT=[TTT tt];
```

```
  else
      T=ts(i)-ts(i-1);
       TT=[TT T];
       tt=TT(i)-TT(i-1);TTT=[TTT tt];
end
end

%选择模型参数。
 ax=1/20;
 xamax=3;
 qqx=(xamax)^2*(4-pi)/pi;

  ay=1/20;
 xamax=30;
 qqy=(xamax)^2*(4-pi)/pi;

xe=[0 0 0 0 0 0]';
 p=100000*eye(6);

M=10;

ap=4;v=4;
R=(0.2303*ap/v)^2;

%估计横轴、纵轴使用最小二乘估计 EKF 方法。
[xx1,xxe1,P33,NN]=funDataDrivenModelYWwithEKF(TT,R,ax,qqx,ay,qqy,xe,
p,dm,N,readerxy);

%使用 UKF 计算的方法，可以节约计算时间，估计横轴、纵轴使用最小二乘估计 Yule-Walker
%方法。
%[xx1,xxe1,P33,NN,qqxx,RR0x,RR0y]=funDataDrivenModelYWnonparell(TT,R
,ax,qqx,ay,qqy,xe,p,dm,N,readerxy);

cov1=xys-[xx1(1,:);xx1(4,:)];%计算方差
mm=mean(cov1,2);
cov1=[cov1(1,:)-mm(1);cov1(2,:)-mm(2)];
cov1=diag(cov1*cov1');
cov1=cov1/length(xys(1,:));
covvv=[covvv cov1];

end
```

```
covvv;
XY=sum(covvv,2)/N
 %画出结果。
XY2=sqrt(XY(1)*XY(1)+XY(2)*XY(2))
plot(xys(1,:),xys(2,:),'k-.','LineWidth',4);hold on
plot(xx1(1,:),xx1(4,:),'r*');hold off
legend('参考轨迹','估计轨迹')
  figure

 subplot(2,1,1),plot(ts,xys(1,:),'k-.','LineWidth',4);hold
on,plot(ts,xx1(1,:),'r*')
 legend('参考轨迹','估计轨迹')
 xlabel('采样时刻'),ylabel('横轴估计结果')

 subplot(2,1,2),plot(ts,xys(2,:),'k-.','LineWidth',4);hold on,
plot(ts,xx1(4,:),'r*')
 legend('参考轨迹','估计轨迹')
 xlabel('采样时刻'),ylabel('纵轴估计结果')

 figure

subplot(2,1,1),plot(ts,xys(1,:)-xx1(1,:),'k','LineWidth',3)
xlabel('采样时刻'),ylabel('横轴估计误差')
subplot(2,1,2),plot(ts,xys(2,:)-xx1(4,:),'k','LineWidth',3)
xlabel('采样时刻'),ylabel('横轴估计误差')
```

2. 实验 2

实验中，在少量 RFID 阅读器的情况下，将前面所提到的方法与几何方法进行对比。如图 8-6 所示，该空间仅有 10 个 RFID 阅读器。由于阅读器不能覆盖所有的空间，因此存在较多低检测率区域，只有一个或者没有阅读器可以获得目标的位置信息。本书的方法跟踪结果展现在图 8-6 中，其中，横轴、纵轴的估计方差分别为 118.7703 和 104.6939。需要注意的是这些值比实验 1 中的方差值大，这是获得的信息变少导致的。

针对图 8-6 的轨迹，利用图 8-1 中给出的几何方法进行估计。如图 8-7 所示，给出了估计结果，该方法获得的横轴、纵轴估计方差分别为 307.2591 和 339.0478，远远大于使用机动目标运动模型的估计结果。由于该方法必须同时获得 2 个或 2 个以上的传感器数据，才能给出较为准确的定位数据，所以当只有一个阅读器获得测量范围时，估计过程就会停止并等待下一次采样。图 8-8 标记出了一个等待周期，由于在等待周期中没有估计更新，导致整个跟踪性能变差。

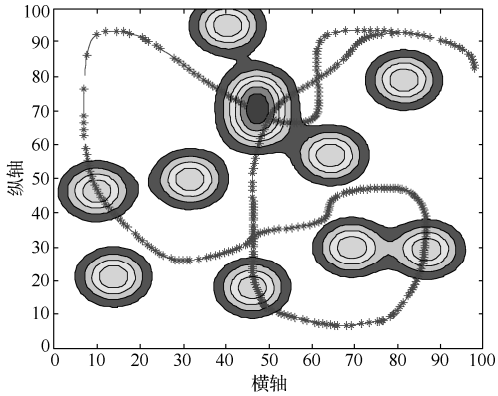


图 8-6　少量 RFID 阅读器区域及目标运动的实际参考轨迹

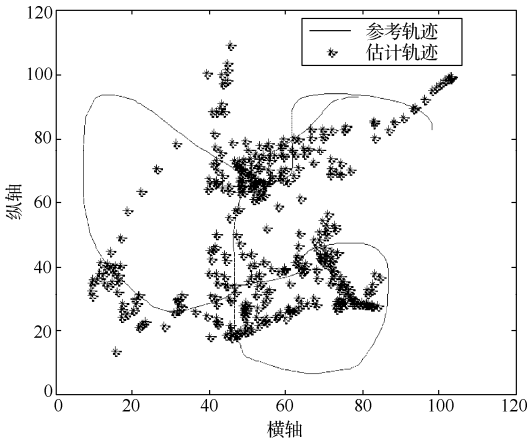


图 8-7　参考轨迹和 UKF 估计轨迹

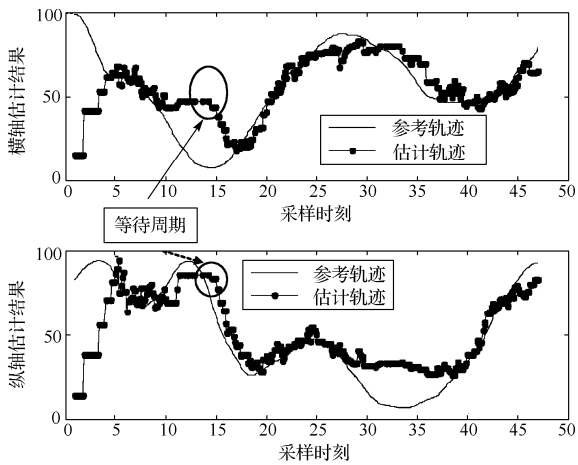


图 8-8　几何定位方法得到的横轴、纵轴估计

是否有效地应用机动目标的运动模型对跟踪性能十分重要，这也是本章方法与图 8-1 所示的几何方法的主要区别。该实验的程序与前面非常类似，主要差别在于多出判断是否是获得 2 个或 2 个以上的传感器数据的部分，由于篇幅的限制，此处不再给出程序。

8.7　本 章 小 结

当标签在 RFID 测量范围内时，标签和阅读器之间的距离可被 RSSI 提取。同时，由于室内跟踪系统测量过程的特点，通过 RFID 技术获得的距离是多元的、不确定和非线性的，并且采样方式是不规则的。

本章分析了 RFID 测量的特点，给出了 RFID 测量模型，并提出了以 EKF 和 UKF 为基础的融合估计算法。为了适应室内机动目标的运动特性，使用了目标的自适应动态模型。仿真结果表明，本书开发的以 UKF 为基础的估计方法可以在较低计算成本下获得较好的估计性能，并且，即使目标进入低检出率区域，系统仍然可以正常工作。

参 考 文 献

Dan S. 2013. 最优状态估计——Kalman, H_∞ 及非线性滤波. 张勇刚, 李宁, 等, 译. 北京: 国防工业出版社

Haug A J. 2014. 贝叶斯估计与跟踪使用指南. 王欣, 于晓, 译. 北京: 国防工业出版社

党建武. 2009. 水下多目标跟踪理论. 西安: 西北工业大学出版社

付梦印, 邓志红, 闫莉萍. 2010. Kalman 滤波理论及其在导航系统中的应用. 2 版. 北京: 科学出版社

苗贝贝, 金学波. RFID 室内跟踪系统仿真数据软件: 中国, 2014SR020578

秦永元, 张洪钺, 汪叔华. 1998. Kalman 滤波与组合导航原理. 西安: 西北工业大学出版社

赵广元. 2011. MATLAB 与控制系统仿真实践. 北京: 北京航空航天大学大学出版社

周宏仁, 敬忠良, 王培德. 1991. 机动目标跟踪. 北京: 国防工业出版社

Arasaratnam I, Haykin S. 2011. Cubature Kalman smoothers . Automatica. DOI: 10.1016/j.automatica

Jin X B, Du J J, Bao J. 2013. Maneuvering target tracking by adaptive statistics model. Journal of China Universities of Posts and Telecommunications, (20): 108-114

Jin X B, Lian X F, Su T L, et al. 2014. Closed-loop estimation for randomly sampled measurements in target tracking system. Mathematical Problems in Engineering, http: //dx.doi.org/10.1155/2014/315908

Julier S J, Uhlmann J K. 1995. A new approach for filtering nonlinear system. Proceedings of the 1995 American Control Conference: 1628-1632

Julier S J, Uhlmann J K. 1996. A general method for approximating nonlinear transformation of probability distributions. http: //www.eng.ox.ac.uk

Julier S J, Uhlmann J K. 1999. New extension of the Kalman filter to nonlinear systems. Signal Processing Sensor Fusion & Target Recognition VI, 3068: 182-193

Julier S J, Uhlmann J K. 2000. A new method for the nonlinear transformation of means and covariances in filters and estimators. IEEE Transactions on Automatic Control, 45(3): 477-482

Julier S J, Uhlmann J K. 2002. Reduced sigma point filters for the propagation of means and covariances through nonlinear transformations. Proceedings of American Control Conference, Jefferson City: 887-892

Kalman R E, Bucy R S. 1961. New results in linear filtering and prediction theory, transactions of the ASME. Journal of Basic Engineering, (83): 95-107

Kalman R E. 1960. A new approach to linear filtering and prediction problems. Transactions of the ASME - Journal of Basic Engineering, (82): 35-45

Mehrotra K, Mahapatra P R. 1997. A Jerk model for tracking highly maneuvering targets. IEEE Transaction on Aerospace and Electronics, (33): 1094-1105

Merwe R V D. 2004. Sigma-Point Kalman Filters for Probabilistic Inference in Dynamic State-Space

Models. http: //www.cslu.ogi.edu/publications

Norgarrd M, Poulsen N K, Ravn O. 2000. New developments in state estimation for nonlinear systems. Automatica, 36(11): 1627-1638

Reza A W, Tan K G, Dimyati K. 2010. Tracking via square grid of rfid reader positioning and diffusion algorithm. Wireless Personal Communications, (61): 227-250

Seah C E, Hwang I. 2011. Algorithm for performance analysis of the IMM algorithm. IEEE Transactions on Aerospace and Electronic Systems, (47): 1114-1124

Sorenson H W. 1985. Kalman Filtering: Theory and Application. New York: IEEE Press, http: //www.cs.unc.edu/~welch/media/pdf/Kalman1960.pdf

Zhou J, Shi J. 2011. A comprehensive multi-factor analysis on RFID localization capability. Advanced Engineering Informatics, (25): 32-40