第5章

目标约束定位

本章考虑对解法有约束的目标定位问题，这发生在对目标位置有一定预知的情况下，如已知高度。我们不详细讨论如何获得这些信息，因为获取方法多种多样，不在本书的讨论范围之内。我们将考虑硬约束和软约束两种类型的目标约束。

硬约束采取固定的等式或不等式约束条件的形式，可表述为

$$\begin{cases} \boldsymbol{a}(\boldsymbol{x}) = \boldsymbol{0} \\ \boldsymbol{b}(\boldsymbol{x}) \leqslant \boldsymbol{0} \end{cases} \tag{5.1}$$

如果上述约束条件是凸的，那么实现一个考虑这种信息的约束求解器是很简单的[1]①。评估估计性能稍微困难，因为克拉美-罗下界（CRLB）没有考虑约束条件的影响。对于等式约束条件，我们将使用约束的克拉美-罗下界（CCRLB）来评估性能[2-3]，我们将证明不等式约束对性能的影响很小（除非目标接近不等式边界），所以我们将使用克拉美-罗下界来估计性能，即使其没有考虑这些约束的影响。

软约束提供了关于目标位置或不可能位置的统计信息。软约束可以来自外部测量（如独立系统对目标位置的估计），或来自跟踪器（如对目标位置和速度的预先估计）。软约束可表示为一个先验概率分布函数 $f_x(\boldsymbol{x})$，以这种方式给出任意测量值 $\boldsymbol{\zeta}$ 的后验似然函数：

$$\ell(\boldsymbol{\zeta}, \boldsymbol{x}) = \ell(\boldsymbol{\zeta} \mid \boldsymbol{x}) f_x(\boldsymbol{x}) \tag{5.2}$$

$\ell(\boldsymbol{\zeta} \mid \boldsymbol{x})$ 是似然函数，它计算出如果真实目标位置是 $\boldsymbol{x}$，测量 $\boldsymbol{\zeta}$ 发生的可能性。后验似然函数 $\ell(\boldsymbol{\zeta}, \boldsymbol{x})$ 反映了测量似然函数和先验概率分布函数 $f_x(\boldsymbol{x})$ 的组合。

应用目标约束的第一步是将它们从最初产生的域转换到解域。在案例中，我们开发的所有算法以及将在本书中介绍的算法都是在笛卡儿坐标系上运行的，而约束通常是在地面坐标系上。最重要的挑战是，虽然一个约束条件在其原始参考框架中可能是线性的或凸的，但它被转换到一个单独的坐标系中后可能是非线性的。出于这个原因，将专门讨论固定高度（5.1 节）和高度有界（5.2 节）两种类型的硬约束，而不是为所有硬约束提供单一的通用推导。在 5.3 节讨论先验估计和信息统计的使用。

---

① 对于使用凸优化方法，约束和代价函数（在这种情况下，似然函数或其简化形式）也必须是凸的。这在定位问题中通常不成立，但它们通常可以被假设为在真实位置 x 周围区域是局部凸的。

5.1　已知目标高度的定位

第一类约束是已知高度的约束。这种情况出现在已知辐射源在地面或海上船只上(因此,处于零高度),或者是通过识别信号的内容(如自动识别系统(AIS)的广播)①。它也可能出现在卫星辐射源上,其轨道周期或速度是已知的,这两者都直接与轨道高度相关联。

Ho 和 Chan[6] 推导了使用 TDOA 和 FDOA 测量已知目标高度时的定位问题。他们表明,假设地球是球形的(因此,约束条件在 ECEF 坐标系中有一个直接的表示),那么这个问题可以直接得到解决。他们还表明,由于地球的离心率,对已知高度的表示更为复杂,求解需要一个迭代的方法。这建立在 Niezgoda 和 Ho 的早期工作上,该工作解决了已知在地球表面的目标,将表面约束表示为 ECEF 坐标下的二次方程,并通过拉格朗日乘子解决了约束优化问题[7]。

5.1.1　优化问题

定位可以表示为一个有约束的优化问题,其中一些约束(或一组约束)$\boldsymbol{a}(\boldsymbol{x})=\boldsymbol{0}$ 必须得到满足。这些约束条件应用到式(2.24)的最大似然公式。

$$\hat{\boldsymbol{x}}=\arg\left[\max_{\boldsymbol{x}}\ell(\boldsymbol{\zeta}\mid\boldsymbol{x})\right] \tag{5.3}$$

使得

$$\boldsymbol{a}(\boldsymbol{x})=\boldsymbol{0}$$

剩下的就是定义约束条件 $\boldsymbol{a}(\boldsymbol{x})$。在最简单的情况下,即平面地球模型,一个已知的高度约束 $z=h$ 可以表示为

$$\boldsymbol{a}(\boldsymbol{x})=\boldsymbol{x}^{\mathrm{T}}[0,0,1]-h \tag{5.4}$$

鉴于平面地球模型假设的固有误差,通常不考虑这一点。然而,在这里把它作为一个约束条件,对用 ENU 坐标系表示的局部问题有效,对它来说,没有考虑地球的曲率,固定高度的约束条件相当于把目标位置的无坐标(或 z 坐标)设定为 $z=h$,如图 5.1 所示。

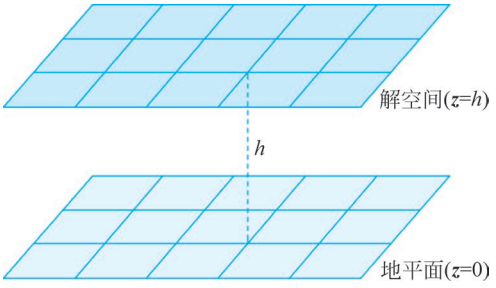


图 5.1　固定高度的平面地球模型示意

球形地球模型示意如图 5.2 所示,根据目标与原点的距离产生二次约束[6]:

$$\boldsymbol{a}(\boldsymbol{x})=\boldsymbol{x}^{\mathrm{T}}\boldsymbol{x}-(R_{\mathrm{e}}+h)^2 \tag{5.5}$$

式中:R_{e} 为地球的平均半径,$R_{\mathrm{e}}\approx 6371\mathrm{km}$。

---

① 自动识别系统是一种基于 VHF 频段的海上交通应答系统,虽然不在最初部署的设计考虑之中,但从空间可以检测到[4-5]。

如果球形地球模型的精度不够，就需要一个椭圆地球模型，如图 5.3 所示，固定高度约束就复杂得多[6]。从式(4.12)可以看出，高度可以在 ECEF 坐标系下计算：

$$h = \frac{p}{\cos\lambda} - R_{\text{eff}}(\phi) \tag{5.6}$$

式中：p 为地球中心到赤道平面上点(x,y,z)的距离，$p=\sqrt{x^2+y^2}$；ϕ 为纬度；λ 为经度；R_{eff} 为有效半径。

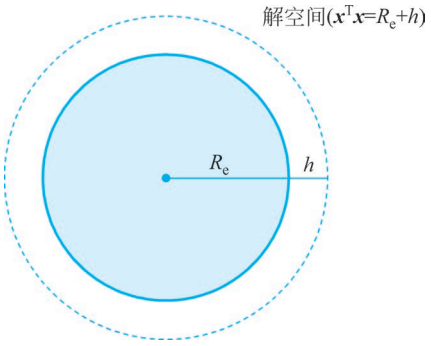


图 5.2 固定高度的球形地球模型示意

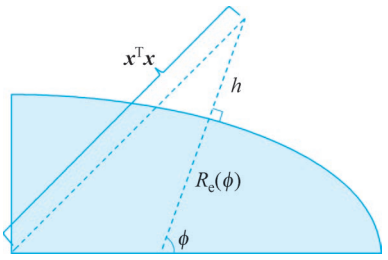


图 5.3 椭圆地球模型固定高度约束示意图（ϕ 是大地纬度，$R_e(\phi)$是该纬度处从地球表面到赤道平面的距离高度约束也被定义为垂直于地球表面，需注意图中地球的离心率被夸大了）

由此，建立约束条件：

$$\boldsymbol{a}(\boldsymbol{x}) = \frac{p}{\cos\lambda} - R_{\text{eff}}(\phi) - h \tag{5.7}$$

图 5.3 说明了大地测量距离计算（通过 h 和$R_e(\phi)$的直线）和地心距离计算（到原点的直线）之间的区别。

这些约束可以通过调用软件包 utils.constraints 中的 fixedAlt 来生成，它返回一对函数句柄，一个是所需的约束（上面定义的），另一个是相对于目标位置 $\boldsymbol{x}$ 的梯度（在 5.1.3 节定义）。

```
h = 10e3; % known target altitude
type = 'ellipse'; % {'flat','sphere',or 'ellipse'} are valid settings
[a,a_grad] = utils.constraints.fixedAlt(h,type);
```

5.1.2 求解器

式(5.3)中的优化问题有很多种求解方法，目前还没有推导出解析解，但存在许多近似解和迭代解。首先讨论如何调整文献[8]中已经提出的解决方案，然后介绍一些专门针对已知高度问题的精选解决方案。

这类优化问题（当代价函数和约束条件都是凸时）的解决方案已在文献[1]①的第 10 章中讨论。这类问题的一般解决方案包括对考虑约束条件的牛顿近似法（在前面用作凸式求

① 正如在文献[8]中所论证的，将在局部凸的假设下计算，利用迭代解决方案求解凸问题，即使代价函数不是严格的凸。这种方法的主要风险是求解器可能收敛于一个错误的解，特别是如果初始的猜想值与真正的解距离太远。

解器的基础)的修改,以及几种求解解析解的数学方法。

1. 通用求解器

在文献[8]中介绍了三种基于似然函数 $\ell(\boldsymbol{x}|\boldsymbol{z})$ 的通用求解器:估计最大似然解的暴力求解器、迭代式的最小二乘求解器以及基于梯度下降的凸优化求解器。这些求解器可以直接作为求解约束问题的三种不同路径。

为了修改暴力搜索的 ML 求解,可直接约束解空间:

$$\ell_c(\boldsymbol{\zeta}|\boldsymbol{x})=\begin{cases}\ell(\boldsymbol{\zeta}|\boldsymbol{x}), & \boldsymbol{a}(\boldsymbol{x})=\boldsymbol{0}\ \text{且}\ \boldsymbol{b}(\boldsymbol{x})=\boldsymbol{0}\\ 0 & \text{其他}\end{cases} \tag{5.8}$$

这个约束是通过将似然性和约束传递给在 utils.constraints 下的 constrainLikelihood,它为 $\ell_c(\boldsymbol{\zeta}|\boldsymbol{x})$ 返回一个函数句柄。这种方法在函数 mlSolnConstrained 中实现,分别在 aoa、tdoa、fdoa 和混合包中。下面是 FDOA 的一个例子,它展示了如何指定约束条件 a 和 b,以及用于等式约束的容差 tol[①]。

```
[x_est,A,grid] = fdoa.mlSolnConstrained(x_fdoa,v_fdoa,rho_dot,C,... x_ctr,search_size,
epsilon,a,b,tol);
```

对于迭代方法来说,情况要稍微复杂一些,但复杂程度并无二致。我们提出了一种迭代约束算法,迭代地求解无约束的优化问题,在每次迭代结束时应用约束条件。为此,在每次迭代时计算目标的期望地心高度和实际地心高度之间的比例项 γ,然后将这个比例项应用于估计位置[②]。

$$\gamma^2=\frac{R_{\mathrm{d}}^2(\phi,h)}{\boldsymbol{x}^{(i)\mathrm{T}}\boldsymbol{P}\boldsymbol{x}^{(i)}} \tag{5.9}$$

$$\tilde{\boldsymbol{x}}^{(i)}=\gamma\boldsymbol{x}^{(i)} \tag{5.10}$$

式中:$\boldsymbol{x}^{(i)}$ 为第 i 次迭代目标位置估计(如 2.2.3 节的梯度下降或 2.2.4 节的迭代最小二乘法);$\tilde{\boldsymbol{x}}^{(i)}$ 为该估计对固定高度约束的投影。

为了便于这种迭代/投影,固定约束函数 fixedAlt 和 fixedCartesian[③] 会产生两个输出,即与约束条件的距离 ε,以及输入位置在约束条件上的投影 $\tilde{\boldsymbol{x}}$。

在每类问题(aoa、tdoa、fdoa 和 hybrid)中,提供了一对约束解算器,它们实现了式(5.9)和式(5.10)可作为函数 gdSolnFixed 和 lsSolnFixd 被调用,例如:

```
[x,x_full] = hybrid.lsSolnFixed(x_aoa,x_tdoa,x_fdoa,v_fdoa,z,C,x_init ,a,tol);
```

新的输入被放在初始目标位置估计值 x_init 之后(这个位置对 GD 和 LS 求解器都是一样的)。a 是等式约束的函数句柄[④],tol 是在等式约束满足或不满足时的可选容差。在 tol

---

① 与其用等于零来测试严格的等式,不如将等式约束约束为具有小于或等于 tol 的大小。这样在测试浮点数时,在数值上更加稳定,因为浮点数几乎从不等于零。

② 在式(5.9)中 γ^2 的定义是针对椭圆问题;在球形情况下,$R_{\mathrm{eff}}(\phi)$ 替换为 R_{e},$\boldsymbol{P}$ 移除。

③ fixedCartesian 函数类似于之前定义的 fixedAlt 约束构建器,此外,它可以用于笛卡儿平面上的通用约束,例如固定的 x、y 或 z 值,或由点 $\boldsymbol{x}_0$ 和矢量 $\boldsymbol{u}$ 定义的直线。两者都可以在 utils.constraints 包中找到。

④ 如果有多个约束,那么它们将作为数组形式的函数句柄传入。

输入之后是常规求解器的标志(如期望的分辨率 epsilon 或标志 force_full_calc)出现的地方。

图 5.4 说明了这个迭代过程,对于一个简单的二维案例,显示了两次迭代。虚线是每个估计值对约束空间的投影,而点画线是下一次迭代(无约束)的更新。

实线旨在代表约束优化问题的标准方法,并不反映任何改进(无论是收敛速度,还是稳定性),这些改进可以通过了解使用的约束类型来实现。这种改进留给读者练习,也留给研究人员进一步研究,在此种情况下,这些算法可以作为一个基准使用。

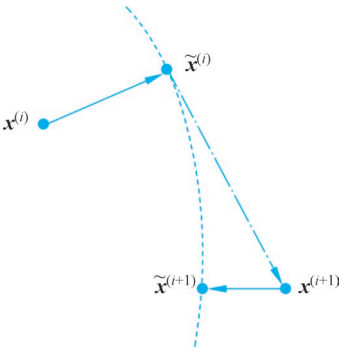


图 5.4 显示了由式(5.9)和式(5.10)所描述的迭代约束优化方法,其中点画线表示无约束的迭代解,实线表示向约束解空间的投影,约束解空间由虚线表示

例 5.1 已知距离的二维方位定位。

首先考虑二维问题,在(−2,0)和(2,0)有两个 AOA 传感器,一个传感器测量目标的到达角为 80°,另一个传感器测量目标的到达角为 87°[①]。绘制场景图,并使用梯度下降求解器对目标进行定位。假设两个传感器的噪声性能相同($\boldsymbol{C}=\sigma^2\boldsymbol{I}$)。

假设沿射程方向的位置 y 已知为 25。生成一个约束函数,并使用固定约束梯度下降求解器来求解 x。绘制两个解,以及传感器和方位线。

解: 第一部分是非常直接的,即定义传感器和它们的测量值,并在图 5.5 中画出。从图 5.5 可以看到,根据视觉上的近似,解应该在点(3.7,32)附近。虽然没有指定协方差,但不需要对 $\boldsymbol{C}$ 有绝对的了解就可以执行求解器,只需要知道所有的测量结果都是同样的噪声(因此,它们在求解中的权重应该是一样的)。因此,简单起见,定义 $\boldsymbol{C}=\boldsymbol{I}$。还必须向 GD 求解器提供一个初始位置估计,所以选择两个传感器之间略微靠前的一个点,作为我们的起始点。

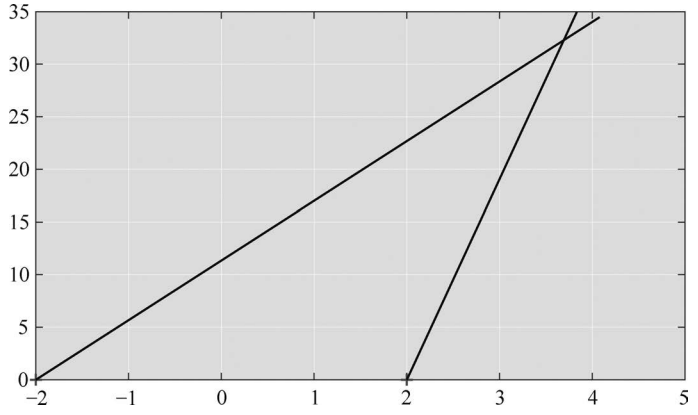


图 5.5 例 5.1 中两个传感器测向线场景

```
x_aoa = [-2,2; 0,0];
psi = [80; 87]*pi/180;
C = eye(n_aoa); % Make it 1 radian std.dev.
x_init = [0;1]; % initial guess
```

① 回想 triang.measurement 函数定义测向弧度为沿+x 轴逆时针方向。

通过调用 triang.gdSoln，可以快速计算出解。

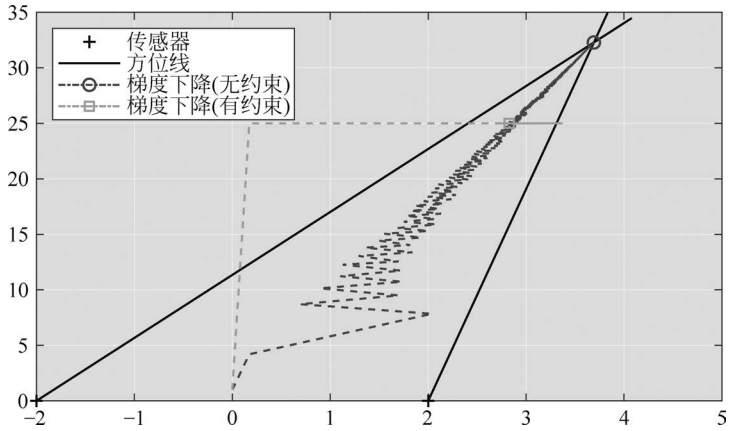
```
[x_ls,x_ls_full] = triang.gdSoln(x_aoa,psi,C,x_init);
```

为了构建有约束的求解，首先定义约束，$y=25$，从图 5.5 可以看到，交点不在这里。因此，期望约束解将不同于无约束解。为了定义约束，使用 utils.constraints 包的函数 fixedCartesian。一旦构建了约束，就以与 gdSoln 相同的方式将其提供给求解器 gdSolnFixed。

```
y_soln = 25;
[a,~] = utils.constraints.fixedCartesian('y',y_soln);
[x_gd_const,x_gd_full_const] = triang.gdSolnFixed(x_aoa,psi,C,... x_init,a);
```

有约束和无约束的结果都绘制在图 5.6 中，并输出在此处：

```
Unconstrained Solution: (3.69,32.28)
Constrained Solution: (2.83,25.00)
```



彩图

图 5.6　两个传感器测向线场景及定位解（分别为无约束梯度下降解以及有约束（$y=25$）梯度下降解）

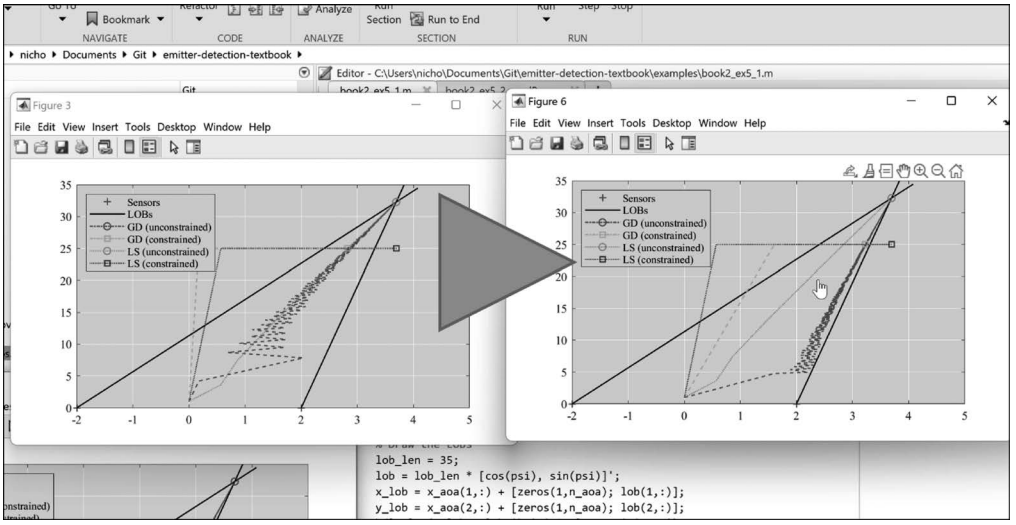
无约束的求解遵循传统的梯度下降方法（之前已经展示过其振荡特性），并收敛到两个 LOB 的交点，正如我们所期望的。然而，受约束的求解在第一次迭代时就迅速捕捉到了 $y=25$，然后沿着这条直线向最可能的解移动，这个解大致出现在两个 LOB 之间。这是最有可能满足固定约束的解。

图 5.7 中链接的视频讨论了例 5.1，并做了一些修改。首先讨论了约束 LS 求解器找到解的能力；然后修改了假设的传感器性能，并讨论了约束 GD 和约束 LS 求解器的表现。

例 5.2　机载 TDOA 传感器对地面目标的定位。

考虑 4 个 TDOA 传感器，在高度 1km 排列在东西方向的线上，每两个传感器之间距离为 10km。在最西边的传感器以北 50km 和以西 10km 处有一个辐射源，高度为 100m，假设所有传感器的计时精度为 100ns。

生成一个具有随机传感器噪声的案例，并在知道和不知道目标高度（100m）的情况下使用迭代梯度下降求解器进行求解。在这两种情况下，用东经 0km、北纬 10km 和与传感器同高度（1km）的猜测来初始化解。绘制传感器、目标和两套解。对轨迹和精度进行讨论。



彩图

图 5.7 例 5.1 的讨论视频(关于约束 LS 求解能力以及传感器性能不一致所产生的影响)

解：首先定义传感器和目标，以及 TDOA 测量误差和测量结果，使用的方法在前面已经有介绍。

```
alt = 1e3;
x_tdoa = [-15e3,-5e3,5e3,15e3; 0,0,0,0; alt1,alt1,alt1,alt1];
[~,n_tdoa] = size(x_tdoa);

% Define target position and initial estimate
tgt_alt = 100; % known target altitude
x_tgt = [-10e3; 40e3; tgt_alt];
x_init = [0;10e3;alt1];
% Sensor Accuracy
time_err = 1e-7;
Croa = (utils.constants.c*time_err)^2 * eye(n_tdoa);
L = chol(Croa,'lower');
% Measurement and Noise
z = tdoa.measurement(x_tdoa,x_tgt,[]);
noise = L*randn(n_tdoa,1); % generate sensor noise
noise_z = utils.resampleNoise(noise,[]); % generate measurement noise
zeta = z + noise_z;
```

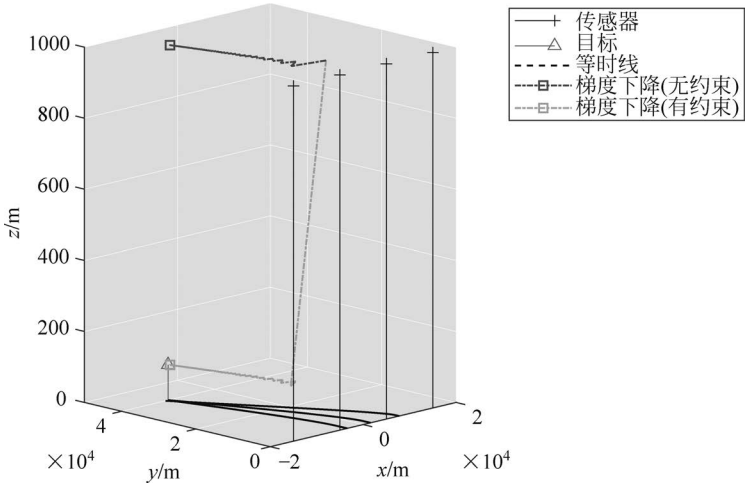
为了求解目标位置，在没有高度约束的情况下使用 tdoa.gdSoln，在有高度约束的情况下使用 tdoa.gdSolnFixed 和 utils.constraints.fixedAlt 提供的约束。

```
[x_gd,x_gd_full] = tdoa.gdSoln(x_tdoa,zeta,Croa,x_init);
[a,~] = utils.constraints.fixedAlt(tgt_alt,'flat');
[x_gd_alt,x_gd_alt_full] = tdoa.gdSolnFixed(x_tdoa,zeta,Croa,x_init,a);
```

这两种情况的结果以及每个求解器的收敛情况绘制在图 5.8 中。两个最终的解是：

```
Unconstrained Solution: -9.87 km E,39.64 km N,1.00 km U
Constrained Solution: -9.87 km E,39.63 km N,0.10 km U
```

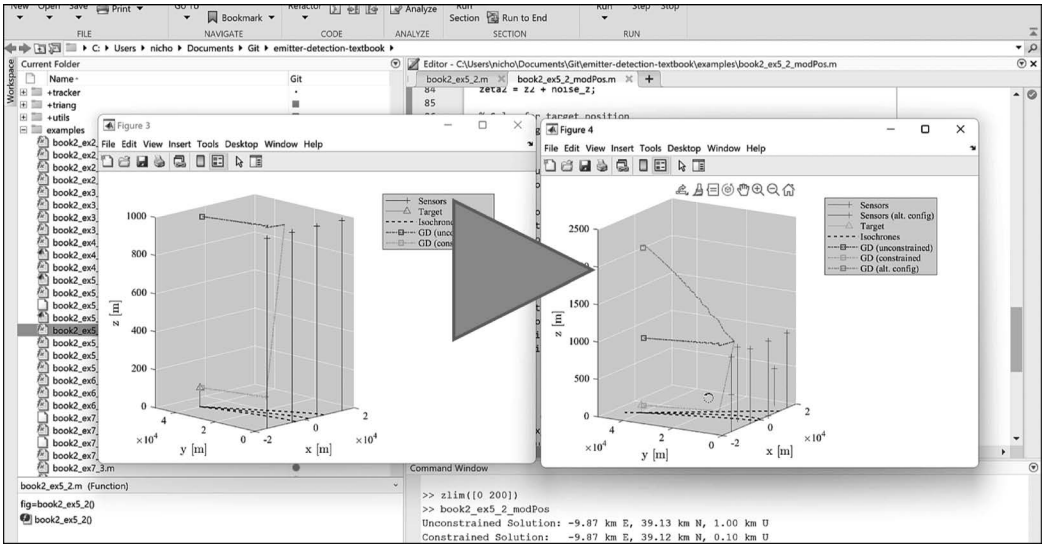
可以看到，无约束求解器在估计东和北坐标方面效果很好，这可能是由于在东西方向的测量噪声低和传感器的间距适当。然而，无约束解算器对高度的估计完全没有偏离最初的1km。这主要是因为传感器没有高度变化，意味着在向上维度没有提供任何信息，因此梯度下降求解器没有理由调整这个估计值。从图5.8中可以看出，无约束求解器从未偏离过这个初始高度。然而，约束求解器立即选择了正确的高度。需要说明的是，在100m的真实高度上，下行(北)位置估计稍微准确。在东、北两个维度上看到这两个估计几乎是相同的，在北维度上偏离大约10m，都在真实目标位置的400m之内。



彩图

图5.8　场景和解(作为参考，等时线(z=0处)用黑色绘制)

在图5.9中链接的视频中探讨了如何利用高度的差异性来提高精确度。



彩图

图5.9　例5.2的讨论视频(利用传感器之间的高度差异来提高精确度)

2. 其他解决方案

下面简单介绍其他几个解决方案，尽管它们没有在本书的相关软件中实现。Bin等[9]使用粒子群优化来解决使用TDOA、FDOA和AOA测量组合的约束定位问题。Cao等[10]

为 TDOA/FDOA 估计使用了一个新的近似代价函数，类似于一个约束的最大似然估计方法，它对已知高度的目标产生了一个迭代求解器（也考虑了传感器位置的不确定性）。

Chan 和 Ho 表明，他们的代数 TDOA 解决方案（在文献[8]第 11 章中描述，并在 tdoa.chanHoSoln 中实现）可以被扩展以考虑已知的高度约束[6]。有兴趣的读者可以查阅文献[8]，了解一些特殊情况的讨论，包括有三个传感器时的代数解和使用拉格朗日乘子法对超定情况的迭代解，或者到 Ho 教授的实验室网站上查阅实现这一算法和许多其他相关定位算法的 MATLAB 代码。

对于等式约束的优化问题，另一类解决方案是通过拉格朗日乘子法将约束直接编码到代价函数上[1]。在我们的例子中，将采取这样的形式：①

$$\ell(\boldsymbol{x}\mid\boldsymbol{\zeta},\boldsymbol{a}(\boldsymbol{x}),\lambda)=\ell(\boldsymbol{\zeta}\mid\boldsymbol{x})-\lambda\boldsymbol{a}^2(\boldsymbol{x}) \tag{5.11}$$

通过包括第二项（$\lambda>0$，并取期望高度和实际高度之差的平方），将任何偏离已知高度的约束作为似然函数的减少。如果非线性约束 $\boldsymbol{a}(\boldsymbol{x})$ 是可微的，这种修改就适用于解决凸问题的数值方法，正如我们在本书及前一版中对其他定位问题的应用。这种方法的主要挑战是拉格朗日参数 λ。大多数求解依靠迭代方法（如牛顿法搜索）来计算 λ 的最优值，然后使用优化的 λ 来解决式(5.11)的估计问题。

5.1.3 性能分析

为了分析性能，我们将着眼于受约束的 CCRLB。这是 CRLB 的一个扩展，它考虑到了约束解空间所带来的不确定性的降低。它最初是由 Gorman 和 Hero[2-3]得出的，但已经被证明是对传统 CRLB 的简单修改[11]。

$$\boldsymbol{C}_x\geqslant\boldsymbol{F}_z^{-1}(\boldsymbol{x})-\boldsymbol{F}_z^{-1}(\boldsymbol{x})\dot{\boldsymbol{A}}(\dot{\boldsymbol{A}}^{\mathrm{T}}\boldsymbol{F}_z^{-1}(\boldsymbol{x})\dot{\boldsymbol{A}})^{-1}\dot{\boldsymbol{A}}^{\mathrm{T}}\boldsymbol{F}_z^{-1}(\boldsymbol{x}) \tag{5.12}$$

式中：矩阵 $\boldsymbol{F}_z^{-1}(\boldsymbol{x})$ 为 Fisher 信息矩阵的逆（对于无约束问题）；$\dot{\boldsymbol{A}}$ 为一组（可能是非线性）约束的梯度（$\boldsymbol{a}(\boldsymbol{x})=\boldsymbol{0}$），对于每个估计变量

$$\dot{\boldsymbol{A}}=\nabla_{\boldsymbol{x}}\boldsymbol{a}^{\mathrm{T}}(\boldsymbol{x})=[\nabla_{\boldsymbol{x}}a_0(\boldsymbol{x}),\cdots,\nabla_{\boldsymbol{x}}a_{N-1}(\boldsymbol{x})] \tag{5.13}$$

式中：$\boldsymbol{a}(\boldsymbol{x})=[a_0(\boldsymbol{x}),a_1(\boldsymbol{x}),\cdots,a_{N-1}(\boldsymbol{x})]$ 是等式约束的集合。CCRLB 的第一项是简单的 CRLB，表明约束优化情况下的误差表现从无约束开始；第二项是 CCRLB（减去后）代表约束条件提供的不确定性的减少，很容易证明这项是半正定的，因此 CCRLB 严格不大于 CRLB。换言之，应用一个高度约束将（假设它是正确的）永远不会增加解的预期误差。

$$\boldsymbol{C}_x\geqslant\underbrace{\boldsymbol{F}_z^{-1}(\boldsymbol{x})}_{\text{CRLB}}-\underbrace{\boldsymbol{F}_z^{-1}(\boldsymbol{x})\dot{\boldsymbol{A}}(\dot{\boldsymbol{A}}^{\mathrm{T}}\boldsymbol{F}_z^{-1}(\boldsymbol{x})\dot{\boldsymbol{A}})^{-1}\dot{\boldsymbol{A}}^{\mathrm{T}}\boldsymbol{F}_z^{-1}(\boldsymbol{x})}_{\text{约束影响}} \tag{5.14}$$

1. 球形地球 CCRLB

对于球面地球问题，在已知高度的情况下，约束的梯度函数简化为 $\dot{\boldsymbol{A}}=\boldsymbol{x}$，CCRLB[6]为

$$\boldsymbol{C}_x\geqslant\boldsymbol{F}_z^{-1}(\boldsymbol{x})-\frac{1}{\boldsymbol{x}^{\mathrm{T}}\boldsymbol{F}_z^{-1}(\boldsymbol{x})\boldsymbol{x}}\boldsymbol{F}_z^{-1}(\boldsymbol{x})\boldsymbol{x}\boldsymbol{x}^{\mathrm{T}}\boldsymbol{F}_z^{-1}(\boldsymbol{x}) \tag{5.15}$$

---

① 在一般的凸优化问题文献中，标准形式是一个最小化代价函数；而在我们的公式中，是最大化似然函数。这就是为什么减去拉格朗日项而不是加上它。

2. 椭圆地球 CCRLB

对于一个椭圆地球，梯度 $\dot{\boldsymbol{A}}$ 参与更多。首先把式(5.7)中的约束条件相对于 x、y 和 z 求偏导，即

$$\frac{\partial \boldsymbol{a}(\boldsymbol{x})}{\partial x}=\frac{x^2-y^2}{x\sqrt{x^2+y^2}\cos\lambda}-\frac{\partial R_{\text{eff}}(\phi)}{\partial x} \tag{5.16}$$

$$\frac{\partial \boldsymbol{a}(\boldsymbol{x})}{\partial y}=\frac{x+y}{\sqrt{x^2+y^2}\cos\lambda}-\frac{\partial R_{\text{eff}}(\phi)}{\partial y} \tag{5.17}$$

$$\frac{\partial \boldsymbol{a}(\boldsymbol{x})}{\partial z}=-\frac{\partial R_{\text{eff}}(\phi)}{\partial z} \tag{5.18}$$

有效半径相对于 x、y 和 z 的偏导数为

$$\frac{\partial R_{\text{eff}}(\phi)}{\partial x}=\frac{ae_1^2\sin\phi\cos\phi}{(1-e_1^2\sin^2\phi)^{\frac{3}{2}}}\frac{\partial\phi}{\partial x} \tag{5.19}$$

$$\frac{\partial R_{\text{eff}}(\phi)}{\partial y}=\frac{ae_1^2\sin\phi\cos\phi}{(1-e_1^2\sin^2\phi)^{\frac{3}{2}}}\frac{\partial\phi}{\partial y} \tag{5.20}$$

$$\frac{\partial R_{\text{eff}}(\phi)}{\partial z}=\frac{ae_1^2\sin\phi\cos\phi}{(1-e_1^2\sin^2\phi)^{\frac{3}{2}}}\frac{\partial\phi}{\partial z} \tag{5.21}$$

令 $\beta=1-e_1^2$；纬度 ϕ 对于 x、y 和 z 的偏导数为

$$\frac{\partial\phi}{\partial x}=\frac{-xz\beta}{\sqrt{x^2+y^2}(z^2+\beta^2(x^2+y^2))} \tag{5.22}$$

$$\frac{\partial\phi}{\partial y}=\frac{-yz\beta}{\sqrt{x^2+y^2}(z^2+\beta^2(x^2+y^2))} \tag{5.23}$$

$$\frac{\partial\phi}{\partial z}=\frac{\beta\sqrt{x^2+y^2}}{z^2+(1-e_1^2)(x^2+y^2)} \tag{5.24}$$

这可以通过使用 utils.constraints[①] 包中提供的 fixedAlt 约束生成器进行数值计算。

```
[a,a_grad] = utils.constraints.fixedAlt(h,type);
```

将第二个函数句柄(a_grad)传递给相关求解器空间(oa、tdoa、fdoa 或混合)中的新约束 CRLB 函数。

```
crlb = hybrid.constrainedCRLB({},a_grad);
```

其中{}是相应的无约束 CRLB 所要求的参数列表。函数句柄 a_grad 接收一个 nDim x M 的辐射源位置集，并返回每个位置的固定高度约束的梯度。然后，受约束的 CRLB 使用这个返回值来计算式(5.12)。

例 5.3　混合定位系统的约束 CRLB。

考虑一个混合定位系统，有一个基于地面的 AOA 传感器(高度为 0m)，以及两个位于

① 对于约束类型的正确设置项是'flat','sphere','ellipse'。

AOA 传感器正东的 TDOA 接收机，分别位于 20km 和 25km 处，高度为 0m。辐射源位于离 AOA 传感器 50km 的地面范围内，方位为自北向东 30°，高度为 10km。AOA 传感器的方位角和仰角都有 5°的误差，而 TDOA 传感器的误差为 1μs。

在高度已知和未知情况下，分别计算真实目标位置的 CRLB，并比较结果。

绘制 RMSE 图，RMSE 由 CRLB 计算得出，计算基于一个由 101×101 个点组成的网格，网格东西向跨度为 20km，南北向跨度为 20km，以真实目标位置为中心，分别在高度已知和未知下进行计算。

解： 首先，与常规做法一致，定义传感器和目标位置。

```
x_aoa = [0,0,0]'; % AOA sensor at reference LLA; origin in ENU
x_tdoa = [20e3,25e3; 0,0; 0,0];
n_aoa = size(x_aoa,2); n_tdoa = size(x_tdoa,2);

bng_tgt = 30; % deg E of N
tgt_rng = 50e3;
tgt_alt = 10e3;
x_tgt = [tgt_rng * sind(bng_tgt);
         tgt_rng * cosd(bng_tgt);
         tgt_alt];
```

定义了角度和到达时间的准确性，用它们来构建 AOA 和 TDOA 的协方差矩阵，并将它们结合起来形成完整的传感器协方差矩阵。尽管只有一个 AOA 传感器，还是为 AOA 生成了一个 2×2 的协方差矩阵，因为它同时生成了方位角和仰角的测量值（要关闭这一点，可以使用二维位置，或者传递一个标志给 hybrid.computeCRLB，指定 do2DAoA＝false）。

```
err_aoa = 3 * pi/180; % 1 deg
err_toa = 1e-6;
C_aoa = err_aoa^2 * eye(2);
C_roa = (utils.constants.c * err_toa)^2 * eye(n_tdoa);
C_full = blkdiag(C_aoa,C_roa);
```

实际使用这些参数来计算 CRLB 和约束 CRLB 是非常简单的。调用 fixedAlt 的第二个输出是一个函数句柄，根据式(5.13)计算约束的梯度。然后，只需将相同的输入与 a_grad 一起传递给 computeCRLBfixed，即可实现 5.1.3 节中定义的 CCRLB。

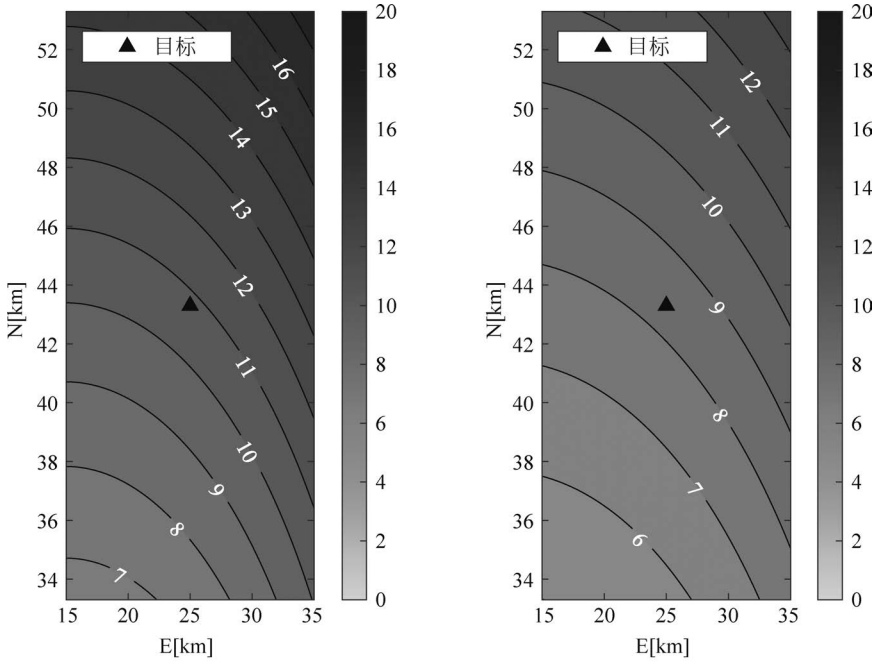
```
C_raw = hybrid.computeCRLB(x_aoa,x_tdoa,[],[],x_tgt,C_full);

[~,a_grad] = utils.constraints.fixedAlt(tgt_alt,'flat');
C_fix = hybrid.computeCRLBfixed(x_aoa,x_tdoa,[],[],x_tgt,C_full,a_grad);
```

结果输出为一对 3×3 协方差矩阵。前者（无约束）的误差高达 $8\times10^7\,m^2$，而后者的主要误差仅为 $5.7\times10^7\,m^2$。更重要的是，第三个维度（高度）在受约束的情况下无噪声，这验证了预期的性能，即约束高度意味着在该维度没有误差（假设约束是准确的）。有趣的是，这也导致其他维度的误差减少，而不仅是将协方差矩阵投影到未受约束的 2×2 子空间上。这一改进是由于现在可以利用传感器收集的所有信息来解决更少的未知数，从而提高估计的准确性。

```
CRLB (unconstrained):
 1.0e+07 *
        1.7851 3.2650 0.7544
        3.2650 8.6964 1.8508
        0.7544 1.8508 1.1415
CRLB (constrained):
 1.0e+07 *
        1.2865 2.0418 0
        2.0418 5.6956 0
        0 0 0
```

最后，在以目标为中心的位置网格上重复测量，而不仅在中心点。一旦定义了测试点的网格，代码就和以前非常相似。唯一的区别是，为了支持绘图，必须将每个 3×3 协方差矩阵简化为一个单一的度量值。我们使用 RMSE，它是通过计算每个协方差矩阵迹的平方根得到的，结果如图 5.10 所示。

```
% Plot for x/y Grid
xy_vec = linspace(-10e3,10e3,201);
[X,Y] = meshgrid(xy_vec);
x_grid = [X(:),Y(:),zeros(numel(X),1)]' + x_tgt;

% Define target position and initial estimate
C_raw_grid = hybrid.computeCRLB(x_aoa,x_tdoa,[],[],x_grid,C_full);
C_fix_grid = hybrid.computeCRLBfixed(x_aoa,x_tdoa,[],[],x_grid,C_full,a_grad);

% Compute RMSE
rmse_raw = reshape(sqrt(C_raw_grid(1,1,:) + C_raw_grid(2,2,:) + C_raw_grid (3,3,:)),size(X));
rmse_fix = reshape(sqrt(C_fix_grid(1,1,:) + C_fix_grid(2,2,:) + C_fix_grid (3,3,:)),size(X));
```



(a) 无约束的均方根误差　　(b) 有约束的均方根误差

图 5.10　定位性能（使用有高度约束和无高度约束的 RMSE）

彩图

5.2 已知高度界的定位

在某些情况下目标的高度可能是有界限的，但并不明确知道，这样的情况是存在的。重复卫星的例子，更有可能的是卫星的轨道高度已知，但有一定的误差。也许是因为观察到了运载火箭，但必须为卫星分离后的机动活动留出一些余地，或者，必须在其最近的观察后留出余地[①]。对于地球上的情况，我们可以考虑，一艘船可能将天线放在高度未知的桅杆上，因此，高度不应该严格为零，而只是接近水面。机载目标也可能同样知道占据一个特定的高度范围，无论是因为政府限制了允许飞行的高度，还是情报数据提供了一些特定飞机飞行高度的知识[②]。在任何这些情况下，考虑已知的高度界限会增加定位的准确性。

5.2.1 优化问题

与已知高度的情况类似，提出了约束优化问题，不等式约束：

$$\hat{\boldsymbol{x}} = \underset{\boldsymbol{x}}{\operatorname{argmax}} \ell(\boldsymbol{x} \mid \boldsymbol{z}), \quad \boldsymbol{b}(\boldsymbol{x}) \leqslant 0 \tag{5.25}$$

定义不等式约束 $\boldsymbol{b}(\boldsymbol{x})$。在最简单的情况下，即平面地球，已知高度约束 $h_{\min} \leqslant z \leqslant h_{\max}$（如图 5.11 所示），并表示如下：

$$\boldsymbol{b}(\boldsymbol{x}) = \begin{bmatrix} \boldsymbol{x}^{\mathrm{T}}[0,0,1] - h_{\max} \\ h_{\min} - \boldsymbol{x}^{\mathrm{T}}[0,0,1] \end{bmatrix} \tag{5.26}$$

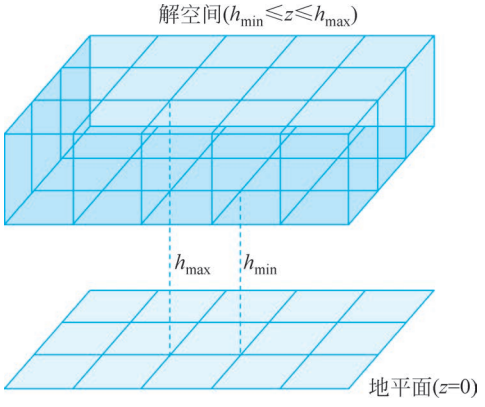


图 5.11 具有高度约束的平面地球模型示意

类似地，可以表达球形地球的高度界限：

$$\boldsymbol{b}(\boldsymbol{x}) = \begin{bmatrix} \boldsymbol{x}^{\mathrm{T}}\boldsymbol{x} - (R_{\mathrm{e}} + h_{\max})^2 \\ (R_{\mathrm{e}} + h_{\min})^2 - \boldsymbol{x}^{\mathrm{T}}\boldsymbol{x} \end{bmatrix} \tag{5.27}$$

对于椭球地球来说，有

---

① 关于卫星轨道挑战性的讨论，包括测量误差对于未来卫星过境和低空轨道的大气阻力预测的影响，参见文献[12-15]。

② 关于空域冲突消融困难性的讨论参见文献[16]。一些关于改善空域冲突的方法的文章，包括使用指定高度层使飞机执行不同任务，可参见文献[17-18]。

$$\boldsymbol{b}(\boldsymbol{x})=\begin{bmatrix}\boldsymbol{x}^{\mathrm{T}}\boldsymbol{P}\boldsymbol{x}-(R_{\mathrm{eff}}(\phi)+h_{\max})^2\\(R_{\mathrm{eff}}(\phi)+h_{\min})^2-\boldsymbol{x}^{\mathrm{T}}\boldsymbol{P}\boldsymbol{x}\end{bmatrix} \tag{5.28}$$

式中：$R_{\mathrm{eff}}(\phi)$和 $\boldsymbol{P}$ 的定义与 5.1 节中的定义类似。这些约束可以通过调用程序包 utils.constraints 中的函数 bounddAlt 来生成。

5.2.2　求解器

正如已知高度的情况代表了一个等式约束的优化问题，对高度边界的了解代表了一个不等式约束的优化问题。关于在凸优化问题中如何处理这个问题的讨论参见文献[1]中的第 11 章。本章的重点是屏障法，尽管也讨论了其他方法。

我们将首先讨论如何修改一般的解决方案来处理这个问题以及其中的限制。

1. 通用求解器

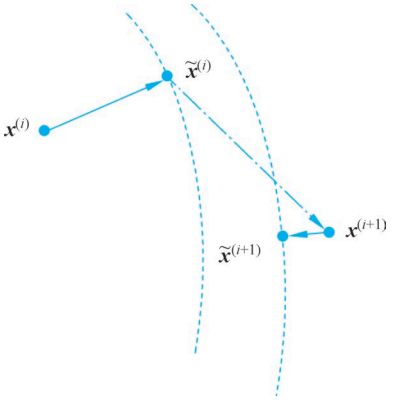
在高度有界约束的情况下，对暴力 ML 解的修改与固定高度的情况相同，求解器被修改为将解空间之外的任何点的可能性设置为 $-\infty$，从而抑制了界外解的任何可能性。同一函数同时实现了固定（等式）和有界（不等式）约束：mlSolnConstrained。

```
[x_est,A,x_grid] = mlSolnConstrained(x_tdoa,rho,C,x_ctr,search_size,epsilon,a,b,tol)
```

其中，a 是等式约束的单元格阵列，b 是不等式约束的单元格阵列，tol 是等式约束的可选容差。

对于迭代方法，也实现了同样的迭代自适应方法，其中无约束求解器被用来更新位置估计，然后在每次迭代中应用边界。在图 5.12 中说明了这种方式高度有界的情况。在这种情况下，有一个额外的步骤，即确定两个不等式约束中的哪一个（如果有）被违反了，需要用来缩放结果。这在 oa、tdoa、fdoa 和混合包中的 lsSolnBounded 和 gdSolnBounded 函数中实现。

在 triang 包中的一个使用实例在此显示。在这个用法中所有的默认参数都没有被指定（控制梯度下降算法的 alpha 和 beta 参数，以及控制收敛时停止和算法执行时输出临时结果的标志）。其他软件包的使用方法类似。这些方法与原始的 gdSoln 函数调用的区别是在初始位置估计后加入了不等式约束 b。



彩图

图 5.12　高度有界问题的迭代更新方法说明（以二维为例）

注：实线箭头表示约束的应用，点画线箭头表示无约束解算器的单个求解步骤。

```
[x,x_full] = gdSolnBounded(x_aoa,psi,C,x_init,b);
```

例 5.4　有界的卫星轨道示例。

考虑 3 个 TDOA 传感器，间距为 50km。中央传感器位于北纬 25°、西经 15°、高度 10m，其他传感器分别在正东和正北 50km 处（同一高度）。有一颗卫星位于北纬 27°、西经 13°、海拔 575km 处。

假设卫星的海拔高度被限制在 500～600km。使用迭代最小二乘法求解卫星的位置，

分别在有海拔约束和无海拔约束的情况下求解。

解： 首先用 ENU 坐标系定义传感器，然后转换为 ECEF 坐标系，因为要处理的是一个在轨道上的卫星，所以应考虑地球的曲率。卫星以 Lat、Lon、Alt 定义，并同样转换为 ECEF 坐标系用于分析，转换为 ENU 坐标系用于绘图。

```
ref_lla = [25, -15,0];
x_aoa_e = [0,50e3,0];
x_aoa_n = [0,0,50e3];
x_aoa_u = [10,10,10];
[x,y,z] = utils.enu2ecef(x_aoa_e,x_aoa_n,x_aoa_u,... ref_lla(1),ref_lla(2),ref_lla(3),
'deg','m');
x_aoa = [x(:),y(:),z(:)]'; % ECEF

n_aoa = size(x_aoa,2);

sat_lla = [27, -13,575e3];
[x,y,z] = utils.lla2ecef(sat_lla(1),sat_lla(2),sat_lla(3),'deg','m');
x_tgt = [x,y,z]'; % ECEF
[e,n,u] = utils.lla2enu(sat_lla(1),sat_lla(2),sat_lla(3),... ref_lla(1),ref_lla(2),ref_
lla(3),'deg','m');
x_tgt_enu = [e,n,u]';
```

通过调用 bounddAlt 和“'ellipse'”标志来定义高度约束，以实现式(5.28)中的高度约束。

```
alt_low = 500e3;
alt_high = 600e3;
b = utils.constraints.boundedAlt(alt_low,alt_high,'ellipse');
```

使用传感器协方差 C_aoa 生成一组测量值，就像我们现在多次做的那样。

```
err_aoa = 1 * pi/180; % 1 deg
C_aoa = err_aoa^2 * eye(2 * n_aoa);
L = chol(C_aoa,'lower');
z = triang.measurement(x_aoa,x_tgt);
n = L * randn(2 * n_aoa,1);
zeta = z + n;
```

解决这两个问题(无约束和有约束)分别是直接调用 gdSoln 和 gdSolnBounded。

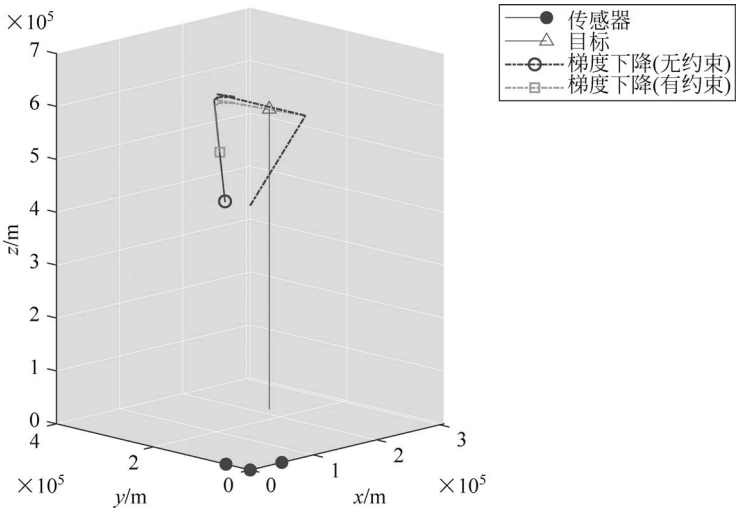
```
 % % Solvers
[x,y,z] = utils.lla2ecef(ref_lla(1),ref_lla(2),500e3);
x_init = [x,y,z]';

[x_gd,x_gd_full] = triang.gdSoln(x_aoa,zeta,C_aoa,x_init);
[x_gd_bnd,x_gd_bnd_full] = triang.gdSolnBounded(x_aoa,zeta,C_aoa,x_init,b);
```

下面输出结果，并输出了每个结果的误差。可以看到，无约束的解只偏离了 0.02°的纬度，偏离了 0.64°的经度和 155km 的高度，三维位置误差略高于 169km。显然，大部分的误差是在高度维度上，这是有道理的，考虑到几何场景(3 个相对密集的纯角度传感器对一个

远距离源进行定位)。在约束的情况下,可以提供外部知识,即高度以 500km 和 600km 为界限。这种高度约束的结果使位置估计只偏离了 97.35km。这有一定的道理,因为无约束的解低于下限约束,所以有约束的解会停留在这个界限。

```
Unconstrained Solution: 27.02 deg N,13.64 deg W,420.39 km
Error: 169.21 km
Constrained Solution: 27.34 deg N,13.43 deg W,500.00 km
Error: 97.35 km
```

在图 5.13 中绘制了传感器、目标以及有约束和无约束的解。在绘图之前,为了更方便地进行可视化,将解从 ECEF 坐标系转换为 ENU 坐标系。



彩图

图 5.13　示例 5.4 中无约束条件和有高度约束情况下的解

2. 屏障法

屏障方法指的是一类问题,它用一个逼近约束的可微函数来代替固定不等式约束(若满足约束,理想情况下没有代价;若违反约束,则代价无穷)。其中一类屏障是缩放的对数[1]:

$$I(u)=-\frac{1}{t}\log u \tag{5.29}$$

式中:t 为锐度参数,$t>0$;u 为不等式约束函数的输出(在我们的例子中为 $b_n(\boldsymbol{x})$),它必须小于或等于零才能满足约束。

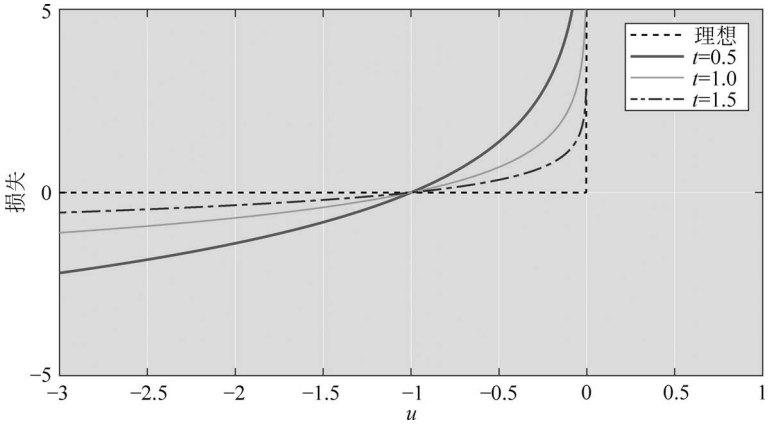
图 5.14 描绘了理想(虚线)和几个对数屏障。随着 $t\to\infty$,屏障接近理想的形式①。在低于约束值时,屏障物的形状会鼓励解偏离不等式约束,但不会阻止它们。更高的 t 值将更准确地反映屏障物,但更有可能出现数值稳定性问题,因为梯度对阻止接近约束条件的解的帮助较小。

关于如何解决优化问题的讨论见文献[1]的第 11 章。

这一点是通过类似于 5.1.1 节中讨论的拉格朗日乘子法:

$$\tilde{\boldsymbol{x}}=\arg\left[\max_{\boldsymbol{x}}\ell(\boldsymbol{\zeta}\mid\boldsymbol{x})-\frac{1}{t}\sum_{n=0}^{N-1}\log(b_n(\boldsymbol{x}))\right] \tag{5.30}$$

① t 的选择是收敛速度(较低的 t 更好)和边界精度(较高的 t 更好)之间的权衡。



彩图

图 5.14 不等式约束($u \leqslant 0$)的对数和理想屏障图示

文献[1]第 11 章中介绍了许多数值方法,可以用来求解式(5.30)。简洁起见,我们没有实现其中任何一个,但鼓励感兴趣的读者从那里开始研究。

5.2.3 性能分析

不等式约束问题的 CRLB 不能以闭合形式求解[19],性能只能通过蒙特卡洛模拟来分析。值得注意的是,不等式约束已经被证明对 CRLB 没有影响[3,20]。因此,2.4.5 节中的标准 CRLB 可以用于已知的高度约束条件(只要约束是准确的)。

5.3 具有统计先验的定位

硬地理约束的主要问题,如已知的高度或地理界限,如果假设不正确,性能将受到影响,定位解会比预期的差。在不能保证目标约束的情况下,不确定性应该通过统计先验来获取,用先验概率分布函数 $f_x(\boldsymbol{x})$表示。任何一组测量值$\boldsymbol{\zeta}$ 的后验似然函数可表示为

$$\ell(\boldsymbol{\zeta},\boldsymbol{x}) = \ell(\boldsymbol{\zeta}\,|\boldsymbol{x}) f_x(\boldsymbol{x}) \tag{5.31}$$

式中:$\ell(\boldsymbol{\zeta}\,|\boldsymbol{x})$是似然函数,它计算出如果真实的目标位置是 $\boldsymbol{x}$,测量$\boldsymbol{\zeta}$ 发生的可能性有多大。后验似然函数 $\ell(\boldsymbol{\zeta},\boldsymbol{x})$反映了测量似然函数和先验概率分布函数 $f_x(\boldsymbol{x})$的结合。这种情况如图 5.15 所示。

如果使用的是对数似然,那么先验信息也必须同样转换为对数形式并加到对数似然中,而不是相乘,即

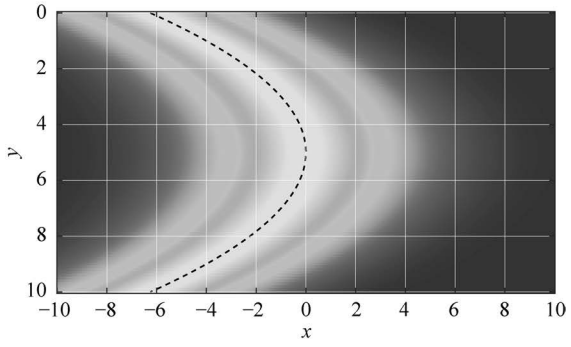
$$\ell(\boldsymbol{\zeta},\boldsymbol{x}) = \ell(\boldsymbol{\zeta}\,|\boldsymbol{x}) + \log(f_x(\boldsymbol{x})) \tag{5.32}$$

5.3.1 先验示例

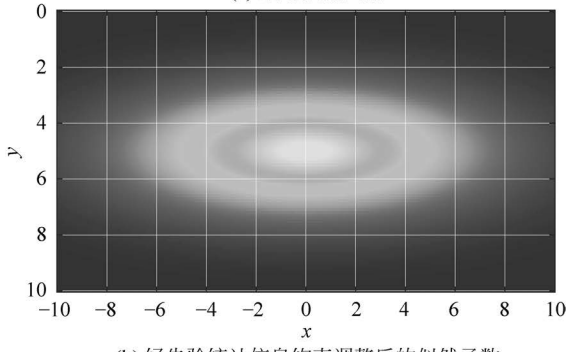
如果有一个先验或外部生成的估计值,只要其分布是完全确定的,就可以用来提高估计性能。这通常表示为高斯分布,其平均值为所提供的估计值,而协方差矩阵则由该估计值的置信度或准确度定义。对于某些估计值 $\hat{\boldsymbol{x}}$ 误差协方差 $\boldsymbol{C}_{\hat{x}}$,

$$\boldsymbol{x} \sim \mathcal{N}(\hat{\boldsymbol{x}}, \boldsymbol{C}_{\hat{x}}) \tag{5.33}$$

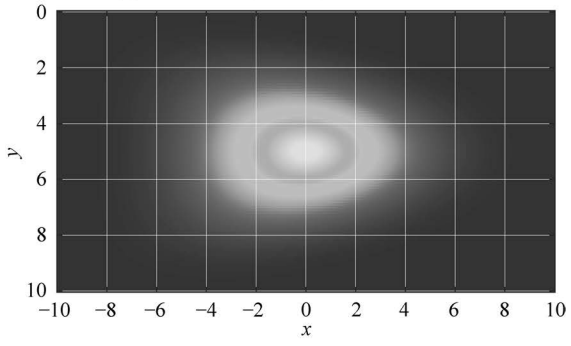
其他分布也同样有效,简单起见,将使用高斯分布(特别是,因为它们只由均值矢量和协方差矩阵两个参数定义)。



(a) 原始似然函数



(b) 经先验统计信息约束调整后的似然函数



(c) 经后验信息乘以先验信息约束调整后的似然函数

彩图

图 5.15　使用先验信息约束调整似然函数的示例

5.3.2　利用先验定位

以前提出的定位的解决方案，如暴力的最大似然估计或迭代最小二乘法估计都适用于有先验的定位，以及所提出的原始问题。缺点是：之前得出的迭代最小二乘法和凸式求解器都假定问题是局部凸的，因此可以用似然函数的梯度来搜索最优解，而先验因素可能使问题不完全凸。对于用平滑分布定义的先验，如高斯先验，不存在这样的问题[①]。

实现是通过将先验定义为一个接收目标位置并返回概率的函数，然后将其传递给相关目录中修改后的 ML 求解器 mlSolnPrior。梯度下降和迭代最小二乘法的迭代求解器本质上并不考虑给定测量集的似然性，因此修改这些算法以包括统计先验较为复杂[②]。

① 关于保持凸性的方法的讨论，可参见文献[1]的 3.2 节。

② 例如，有关先验分布如何影响凸优化问题的讨论可参见文献[1]的 7.1.2 节。

对于有先验的 ML，根据式(5.32)[①]，先验的对数被添加到对数似然函数中。例如，tdoa 包中的版本使用以下两行来生成，然后修改似然函数的函数柄。

```
ell = tdoa.likelihood(... );
ell_posterior = @(x) ell(x) + log(prior(x));
```

例 5.5 外部估计对地面 TDOA 的先验辅助。

4 个接收机的地面 TDOA 系统，以 Y 形配置排列，一个传感器在中心，其他的在正北 10km 外、西边 30°、东边 30°(换言之，在北边 0°、120°，东边 240°)。接收机的时间误差为 1μs。一个辐射源位于 100km 外的正东方向，高度为 20kft(1ft＝0.3048m)。在这个问题上，假设地球是球形的。

一个外部传感器产生了对目标在 ENU 坐标下[95,0,10]km 处的位置估计，且协方差矩阵为

$$\boldsymbol{C}=\begin{bmatrix}5 & 1 & 0\\ 1 & 50 & 0\\ 0 & 0 & 10\end{bmatrix}\times 10^{6}\,\mathrm{m}^{2} \tag{5.34}$$

构建高斯先验，用它来解决有随机传感器噪声的定位问题，使用最大似然求解器，网格向东和向北延伸±50km，以目标的真实向垂直坐标为中心，样本点的间距为 250m。与没有先验的定位结果进行比较。

解： 用一些三角学的方法直接定义 TDOA 传感器的位置，很容易地增加围绕中心参考的环形接收机的数量。由于默认的行为是第 n 个接收机是共同的参考，把这个放在最后。定义目标位置的过程稍微复杂。为了考虑地球在远距离的曲率，首先以地面单位定义位置，也就是沿地球表面向东和向北的距离(m)，然后是垂直高度。可以使用这些作为 ENU 下的坐标，但这样一来，所陈述的“向上”坐标和指定位置在地球表面以上的高度之间会有很大的差异。为了说明球形地球的情况，使用实用程序 correctENU。这个工具将指定的坐标(沿地球表面的东/北，以及当地的高度)转换为笛卡儿坐标系的 ENU。在我们的案例中，结果表明，虽然目标在地球表面以上 12.1km(40kft)，实际上是 11.41km。

```
baseline = 10e3;
n_tdoa = 4;
tdoa_ang = pi/6 + 2*pi/3 *(0:n_tdoa-2);
x_tdoa = baseline * [cos(tdoa_ang),0; ...
                     sin(tdoa_ang),0;
                     0,0,0,0]; % ENU
 % Define target coordinates
tgt_rng = 100e3;
tgt_alt = 40e3*unitsratio('m','ft');
x_tgt_g = [tgt_rng; 0; tgt_alt]; % spherical Earth coors (East,North,Alt)
[e,n,u] = utils.correctENU(x_tgt_g(1),x_tgt_g(2),x_tgt_g(3));
x_tgt = [e; n; u]; % ENU
```

① 虽然式(5.32)在数学上是完备的，但选择增加一个可调参数 $\lambda\in[0,1]$，用于提供似然和先验($\lambda=1$)之间的线性权重。默认值 $\lambda=0.5$ 在数学上等同于式(5.32)，任何偏离都表示测量值($\lambda<0.5$)或先验($\lambda>0.5$)的权重更大。

为了生成先验，首先输入先验位置估计值和协方差矩阵，然后用它们生成一个函数句柄，该句柄将它们作为平均值和协方差调用 mvnpdf。注意，mvnpdf 按行处理数据，而输入为列，所以必须将 x 和 x_prior 进行转置操作。就像处理沿射程方向的目标位置一样，将把先验值转换成 ENU，以校正地球的曲率。

```
x_prior_g = [95; 10; 10] * 1e3;
[e,n,u] = utils.correctENU(x_prior_g(1),x_prior_g(2),x_prior_g(3));
x_prior = [e; n; u];
C_prior = [20,5,0; 5,10,0; 0,0,10] * 1e5;

prior = @(x) mvnpdf(x',x_prior',C_prior);
```

生成测量值的过程按常规进行。

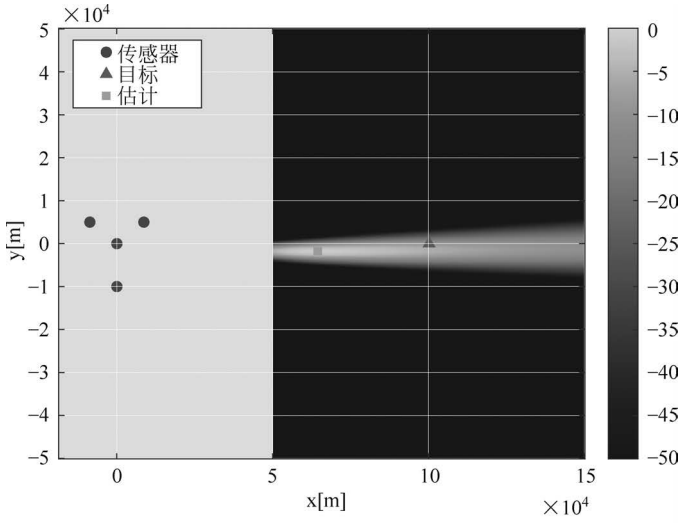
```
z = tdoa.measurement(x_tdoa,x_tgt);

time_err = 3e-7;
C_roa = (utils.constants.c * time_err)^2 * eye(n_tdoa);
C_rdoa = utils.resampleCovMtx(C_roa,[]);
L = chol(C_rdoa,'lower');

n = U * randn(size(z));

zeta = z + n;
```

接下来是一个简单直接的过程，即定义一个搜索网格，然后在有先验和无先验的情况下执行最大似然的定位。

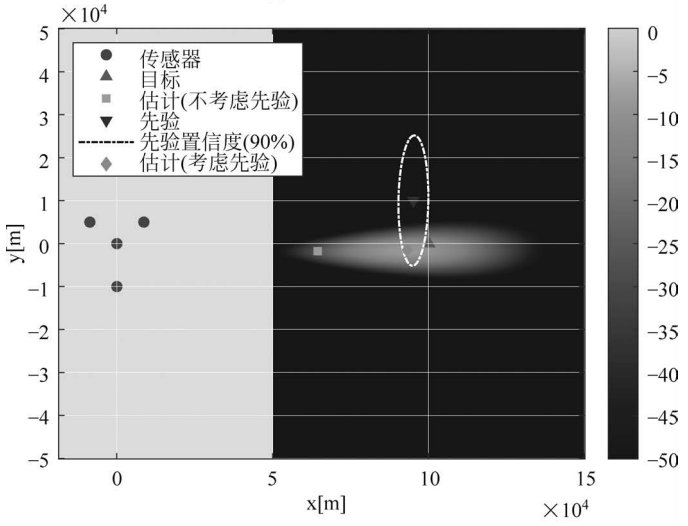
```
x_ctr = x_tgt;
grid_size = [50e3,50e3,0];
epsilon = 250;
[x_ml,A,x_grid] = tdoa.mlSoln(x_tdoa,zeta,C_roa,x_ctr,grid_size,epsilon);
[x_ml_p,A_p,~] = tdoa.mlSolnPrior(x_tdoa,zeta,C_roa,prior,x_ctr,grid_size,epsilon);
```

这里输出了定位的估计值，并绘制在图 5.16 中。我们注意到，图 5.16(a)中没有先验的原始情况，显示了一个预期的结果：良好的角度定位，以及来自 TDOA 传感器的相对较差的下距离定位。解与真正的目标位置相差大约 35km，考虑假设的 500ns 的时间误差，结果是合理的。然而，在有先验的情况下，从图 5.16(b)中看到了白色的虚线，表示为 90%的置信度所画的椭圆，它提供了一些范围信息。即使给出的先验值与真实位置相差约 12km，结果的估计值仍减小到 11km，因为先验被赋予了与测量值相同的权重，能够利用新的测量值来改善北面和上面的坐标，而先验改善了估计东面坐标的能力(略)。注意，真正的目标位置是在先验提供的 90%置信椭圆之外，在这种情况下统计先验并不准确，但仍提供了一些好处。

```
Solution w/o prior: 64.44 km, -1.75 km,11.41 km
    Error: 35.79 km
Solution w/prior: 89.44 km, -0.75 km,11.41 km
    Error: 10.78 km
```



(a) 无先验分布



(b) 有先验分布

彩图

图 5.16 ML 定位结果说明

注：图(b)还显示了先验期望(紫色倒三角)以及先验置信度 90%的误差椭圆(白色虚线)。

5.3.3 性能分析

为了分析性能，将采用 CRLB，并展示如何修改它以包括 $\boldsymbol{x}$ 上的先验分布。从 Fisher 信息矩阵的随机定义[21]开始，应用于对数似然函数 $\ell(\boldsymbol{\zeta}\,|\boldsymbol{x})$，即

$$F_{i,j}(\boldsymbol{\zeta}\,|\boldsymbol{x}) = E\left[\frac{\partial^2}{\partial x_i \partial x_j}\ell(\boldsymbol{\zeta}\,|\boldsymbol{x})\right] \tag{5.35}$$

将同样的公式应用于后验似然函数，并简化其分量，即

$$\begin{aligned} F_{i,j}(\boldsymbol{x},\boldsymbol{\zeta}) &= E\left[\frac{\partial^2}{\partial x_i \partial x_j}(\ell(\boldsymbol{\zeta}\,|\boldsymbol{z}) + \log(f_{\boldsymbol{x}}(\boldsymbol{x})))\right] \\ &= E\left[\frac{\partial^2}{\partial x_i \partial x_j}\ell(\boldsymbol{\zeta}\,|\boldsymbol{z})\right] + E\left[\frac{\partial^2}{\partial x_i \partial x_j}\log(f_{\boldsymbol{x}}(\boldsymbol{x}))\right] \end{aligned} \tag{5.36}$$

$$=F_{i,j}(\boldsymbol{\zeta}\,|\boldsymbol{x})+F_{i,j}(\boldsymbol{x})$$

式中：$F_{i,j}(\boldsymbol{\zeta}\,|\boldsymbol{x})$是基于以目标位置 $\boldsymbol{x}$ 为条件的接收数据的可能性的标准 Fisher 信息矩阵，见式(5.35)；$F_{i,j}(\boldsymbol{x})$是先验的 Fisher 信息矩阵。因此，对于给定的任何先验信息，都必须对其求解。在均值为 $\hat{\boldsymbol{x}}$ 的高斯先验的情况下(假定为无偏的 $\boldsymbol{x}$ 的估计)，以及误差协方差 $\boldsymbol{C}_{\hat{\boldsymbol{x}}}$，先验的 Fisher 信息矩阵为

$$\boldsymbol{F}(\boldsymbol{x})=\boldsymbol{C}_{\hat{\boldsymbol{x}}}^{-1} \tag{5.37}$$

因此，在位置上有先验分布的 CRLB 为

$$\boldsymbol{C}_{\boldsymbol{x}}\geqslant[\boldsymbol{F}(\boldsymbol{\zeta}\,|\boldsymbol{x})+\boldsymbol{C}_{\hat{\boldsymbol{x}}}^{-1}]^{-1}=[\boldsymbol{J}_{\boldsymbol{z}}^{\mathrm{T}}(\boldsymbol{x})\boldsymbol{C}_{\boldsymbol{\zeta}}^{-1}\boldsymbol{J}_{\boldsymbol{z}}(\boldsymbol{x})+\boldsymbol{C}_{\hat{\boldsymbol{x}}}^{-1}]^{-1} \tag{5.38}$$

为了实现这一点，建立了实用函数 makePrior，它将返回两个函数句柄：一个是先验分布 $f_{\boldsymbol{x}}(\boldsymbol{x})$；另一个是 Fisher 信息矩阵($\boldsymbol{F}(\boldsymbol{x})$)[①]。后者可以被传递到 CRLB 的一个修改形式中，称为 computeCRLBpriors，它将接收所有与 CRLB 相同的输入、匹配函数 computeCRLB 以及处理先验 FIM 的函数，并将从式(5.38)计算后验 FIM。

```
[prior,fim_prior] = utils.makePrior('gaussian',prior_mean,prior_covariance);
crlb = tdoa.computeCRLBpriors({ typical crlb params },fim_prior);
```

习题

1. 导出式(5.16)～式(5.18)中的偏导数$\frac{\partial a(\boldsymbol{x})}{\partial x}$、$\frac{\partial a(\boldsymbol{x})}{\partial y}$和$\frac{\partial a(\boldsymbol{x})}{\partial z}$。

2. 重复例 5.2，使用迭代最小二乘求解器。对有无目标高度约束的差异进行评论，并与梯度下降的情况进行比较。

3. 考虑例 5.3，为 1000 次蒙特卡洛试验产生噪声测量。实施一个迭代最小二乘求解器，将位置估计值的 RMSE 与例 5.3 中的 CRLB 进行比较(包括知道和不知道目标高度)。

4. 重复例 5.3，用 FDOA 接收机代替 TDOA，频率误差为 100Hz，载波频率为 3GHz，传感器速度为 100m/s，航向为北偏东 45°。

5. 考虑例 5.4，生成 1000 次蒙特卡洛试验的噪声测量值和解估计值，并计算 RMSE。计算无偏的 CRLB，并与有界和无界求解器的性能进行比较。

6. 为东经 100km 处的目标定义一个 ENU 坐标的固定约束。

7. 在 ENU 坐标中为北纬 100km 和 200km 之间的目标定义一个有界约束。

8. 在 ECEF 中为一颗位于 1500km 的卫星定义一个固定高度的约束条件。

参考文献

① 参见 utils.makePrior 的使用说明。当前只实现了高斯和均匀先验。