卫星信号仿真平台
SignalSim 设计说明

1.4 版

莫钧

2023/11/13

前言

我从近 20 年前开始从事卫星导航接收机设计工作，从最早修改调试单独的功能模块，到后来接触整体设计，再到后来设计整个接收机的架构。在这个过程中，对于接收机的设计过程和需求积累了很多的经验，也在不断地寻找更好的设计、验证和调试的方法及工具。

卫星导航接收机是一个比较复杂的系统，从射频前端、基带处理、信号跟踪、观测量生成一直到位置解算。在设计和实现的过程中，就会有对其中单独的功能单元进行验证和测试的需求。上述的各个功能单元，基本上在数据流上是串行的，也就是说上一级的输出作为下一级的输入。因此，在功能单元的设计和实现中，往往需要人为地设计一个功能单元的输入，以替代前一级的输出。这样，就可以不必依赖前一级的功能单元。另外，卫星导航接收机往往需要工作在各种场景之下。当无法在实际的物理条件下在相应的场景进行测试的时候，需要更好更有效率地对场景进行模拟。

我在这些年工作地过程中，也不断地根据上述需求设计一些仿真平台以及简单的信号生成程序。SignalSim 是我总结了这些需求和当初满足这些需求所做的工作的基础上，综合形成的一个多功能多层次卫星信号仿真平台。

这个平台，首先可以作为一个教学平台，因为平台的基本功能是以源代码形式提供，这样使用者可以明了数据产生的方式，并根据自己的需求改变产生仿真数据的方法和逻辑。其次，仿真平台可以纯粹运行在 PC 上，因此，不必借助实际的接收机或者硬件平台，就可以进行仿真和开发。第三，仿真平台可以实现从中频采样数据一直到原始观测量的多个层级的数据仿真，方便研究和仿真接收机各个功能模块。

同时，这个仿真平台也可以当作企业在设计接收机时的开发验证平台使用，通过适当的定制化设计，仿真平台可以产生与接收机的实际实现相匹配的数据流，方便工程人员进行开发和验证，并且可以在一定程度上减少对卫星信号模拟器的依赖。

SignalSim 本身是一个多功能的系统，在发布的时候会分阶段实现。由于接收机的设计多种多样，信号的特性也很复杂，因此这个平台也只能提供一个基本的框架。象类似于基带相关结果生成和仿真的部分，也会按照 OPENGNSS 设计中的结构去实现。如果接收机的基带相关实现方法不同，也需要改动相应的函数进行适应。

设计这个信号仿真平台并且进行开源的目的，就是希望能够更多地帮助到卫星导航相关领域的学习者和从业者可以更好地理解和学习相关的知识及提高工作效率，也希望能够借助这个平台，吸引更多的人参与到这个行业中来，为卫星导航产业尽绵薄之力。

SignalSim 的源代码可以在 gitee 上进行访问，下载/克隆地址为：

https://gitee.com/opengnss/SignalSim.git

莫钧

2020 年 7 月 8 日

于美国加州 Fremont 市

版权信息

版本更新

版本	日期	更新说明
1.0	2020.7.8	初始版本
1.1	2021.6.29	增加 XML 配置文件解析说明和原始观测量生成程序说明
1.2	2023.1.21	增加了对多频信号支持的说明
1.3	2023.8.26	增加了对常用函数的说明
1.4	2023.11.13	增加了对 RINEX 4 格式的支持以及对多种格式导航电文的支持

1. SignalSim 整体设计构思

在卫星导航接收机的设计和测试中，需要构造各种不同的场景，其中包含一些极端的场景。由于在实际条件下测试所有的场景是不可能的，所以需要有各种不同的仿真的手段来进行不同场景的仿真。

最常见的仿真手段就是卫星信号模拟器。但是卫星信号模拟器有若干固有的缺陷。第一是价格比较昂贵，给每一位开发和测试人员配备一台模拟器不太现实。第二是模拟器只能测试完整的接收机，很难测试独立的模块。第三是模拟器只能给出轨迹和原始观测量作为标准参考量，无法给出其他的参考结果用于调试。第四是模拟器只能实时运行，如果遇到需要减慢速度（如以较低频率运行的接收机基带原型）或加快速度（如测试场景的时间很长）就无法满足需求。用于卫星信号录制和回放的设备也有类似的缺陷。

因此，一个基于软件或者软硬件结合的成本低廉且可以输出中间参考量的信号模拟平台就可以克服上述缺点。SignalSim 就是这样的一个软件平台。SignalSim 设计为可以用于开发和测试位置解算程序，基带跟踪程序，基带硬件加速单元。经过适当的硬件加速和扩展，甚至可以当作一个简单的卫星信号模拟器使用。

SignalSim 的设计思路是基于卫星信号重构的方式，首先计算得到在某个特定接收时刻收到的各个卫星信号的发射时刻，然后依此得到基带信号。对一定时间间隔内的基带信号进行插值可以得到每一个采样时刻的数字中频采样信号，再经过上变频就可以得到射频信号。上述的步骤在每一步都可以输出作为中间结果，并用于接收机相应功能模块的开发与调试。

SignalSim 的核心部分就是产生特定时刻的中频信号。用于产生中频信号的公式如下：

$$s_{IF}(t) = \sum_{n=1}^{N}\{\sqrt{2P_n}D_n(t - T_p - \delta t_{iono})C_n(t - T_p - \delta t_{iono})\cos[\omega_{IF}t - \omega_{RF}(T_p - \delta t_{iono} + \phi_0)]\} + noise$$

其中P_n是卫星信号的功率，C_n和D_n分别是扩频码和广播的调制数据。ω_{IF}和ω_{RF}分别是中频和射频频率。T_p是卫星信号传播时间，这个传播时间需要经过星钟、地球自转、相对论和对流层延时调整，但是不包括电离层延时。δt_{iono}是电离层延时。ϕ_0是初始相位，该相位可以简单地设置为 0。

需要注意的是，对于特定的接收机，ϕ_0、ω_{IF}和ω_{RF}是常数，而T_p和δt_{iono}都是时变量。同时，每颗卫星的信号功率P_n也可能因为环境和仰角的不同而变化。上述变量中是T_p快速变化的，其他的变量都是缓变的。

我们定义码相位$T_c = t - T_p - \delta t_{iono}$，载波相位$\Phi = 1/2\pi\,[\omega_{IF}t - \omega_{RF}(T_p - \delta t_{iono})]$。其中码相位的单位是秒，载波相位的单位是周。之所以这样定义，是因为计算时可以完全忽略载波相位的整数部分，而只保留小数部分。这样，中频信号可以被重新写成：

$$s_{IF}(t) = \sum_{n=1}^{N}\{\sqrt{2P_n}D_n(T_c)C_n(T_c)\cos[2\pi\Phi]\} + noise$$

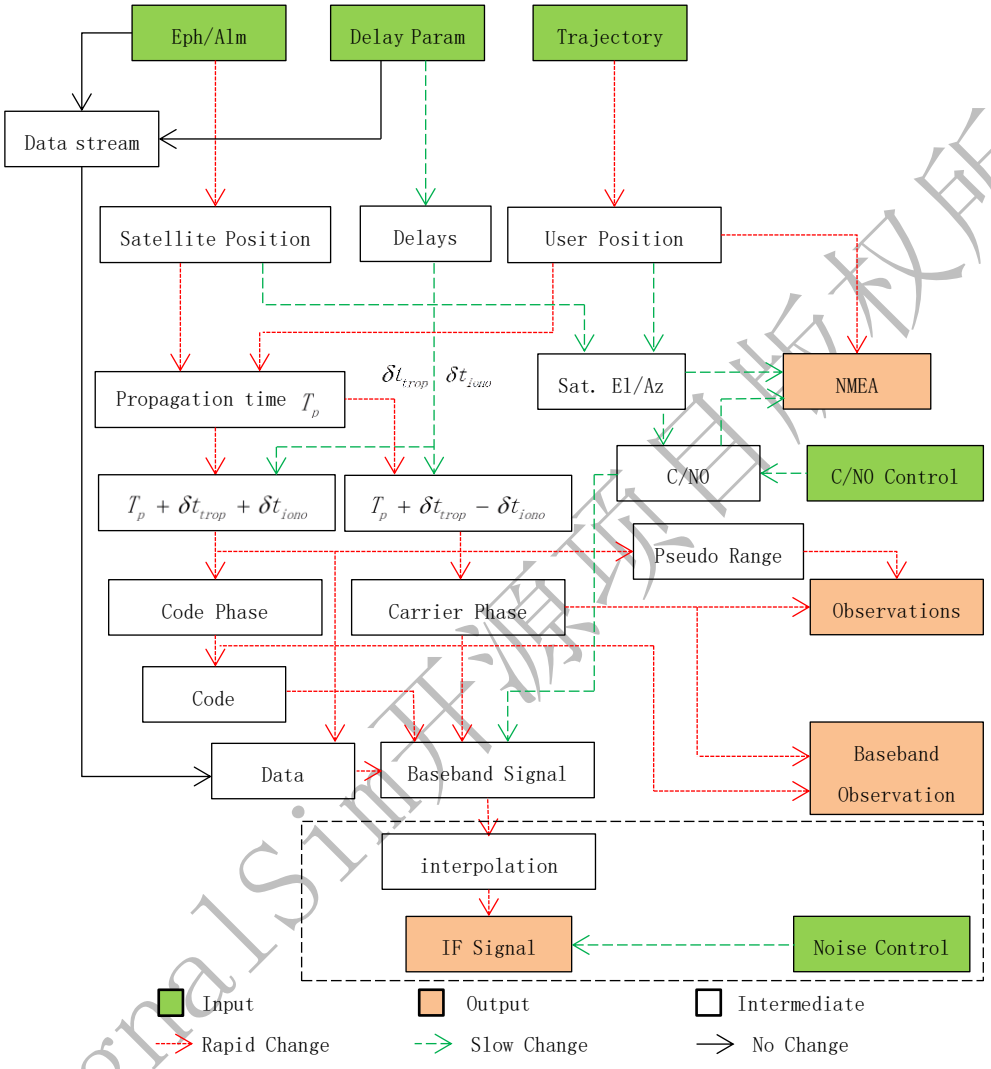
对于复信号，也可以有类似的中频复信号公式：

$$s_{IF}(t) = \sum_{n=1}^{N} \{\sqrt{2P_n} B_n e^{i[\omega_{IF} t - \omega_{RF}(T_p - \delta t_{iono} + \phi_0)]}\} + noise$$

对于每一颗卫星，都有：

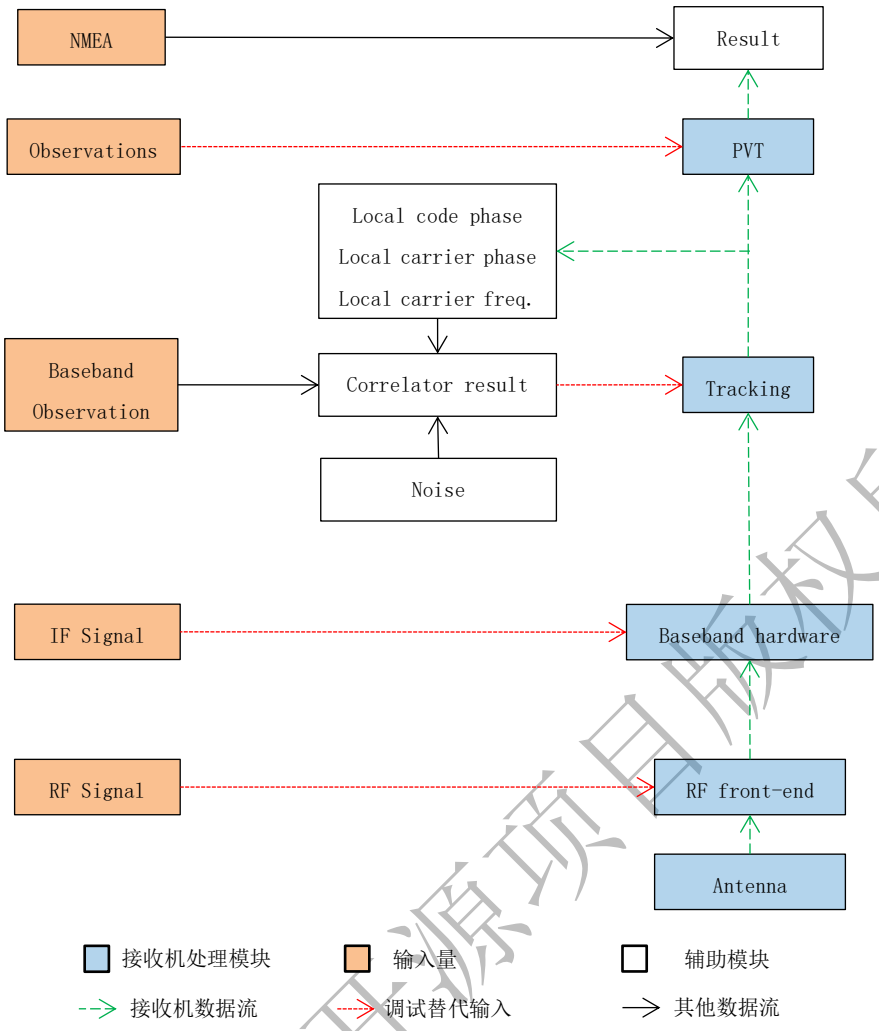
$$B = D_I(t - T_p - \delta t_{iono}) C_I(t - T_p - \delta t_{iono}) + iD_Q(t - T_p - \delta t_{iono}) C_Q(t - T_p - \delta t_{iono})$$

SignalSim 的整体设计如下图所示：



在上图中，绿色的部分是输入量，包括接收机轨迹、星历/历书、对流层/电离层延时模型参数，卫星功率控制和噪声模型控制。完整的 SignalSim 可以包括四个不同阶段的输出。第一个是可以作为定位参考结果的接收机位置的输出，可以是 NMEA 的形式，也可以是其他形式。第二个是原始观测量的输出，可以用于调试位置解算程序。第三个是基带观测量，包含卫星信号的码相位和载波相位，可以用于调试基带跟踪程序。第四个是数字中频采样，可以用于调试基带信号处理逻辑。如果虚线框内的部分采用硬件加速实时输出的话，还可以经过上变频得到射频信号。

下图显示了各个输出与接收机各个处理模块的对应关系。可以看出，SignalSim 实现了一个反向的接收机处理流程。



为了更好地组织输入量，并且方便以后进行扩展，输入量采用 XML 的格式。具体的组织形式和定义方法详见《SignalSim 输入信息 XML 规范文档》。

除了上述基本功能以外，SignalSim 还可以根据需要进行扩展，比如增加如惯性传感器等数据。惯性传感器的数据源可以根据轨迹文件和载体姿态产生，并加入传感器误差模型。这样，可以进行松耦合/紧耦合，甚至深耦合的算法研究。

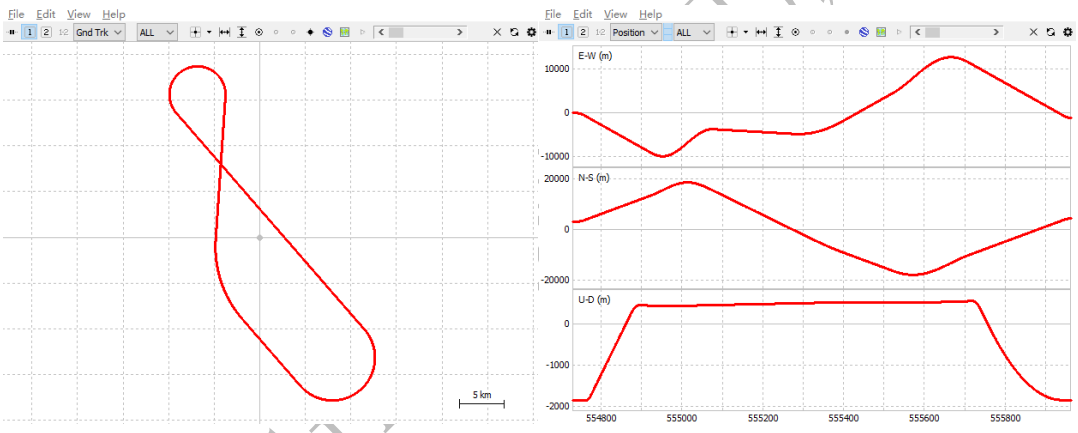
2. 接收机轨迹

接收机轨迹是 SignalSim 的输入之一，定义了接收机的动态运行轨迹。为了使接收机轨迹设计设计更加灵活，并且计算轨迹上的点更加精确，接收机轨迹的设计不是采用特征点的方式，而是采用轨迹拼接的方式。

轨迹首先包含接收机的初始位置和速度，然后以起始点为基础，整个轨迹采用一段一段的不同类型的轨迹首尾相接拼接而成。轨迹的类型包含以下几种：

- ✓ 匀速直线运动
- ✓ 匀加速直线运动
- ✓ 垂直加速运动
- ✓ 变加速直线运动
- ✓ 水平转弯

绝大多数载体的运行轨迹，都可以采用上述类型的轨迹通过拼接来进行模拟。比如，下图显示了一段飞行的轨迹：



飞行轨迹首先向北偏西方向运动，然后加速起飞。到达一定高度平飞后顺时针转弯 225 度，再继续飞行后逆时针转弯 45 度，再飞行一段时间后逆时针转弯 180 度，最后下降高度返回原来的位置。

这样的轨迹拼接方式，仅仅需要改变初始的位置和速度，就可以将相同的轨迹以不同的起始位置和角度放在任意需要的位置上。当然，由于地球是一个椭球体，所以对于范围比较大的运动轨迹，原本的水平移动在经过一段距离以后，就不再是水平的。虽然每一段轨迹在实际计算的时候会根据轨迹的起始位置重新计算 ECEF 坐标系下的位置速度与本地坐标系的位置速度的转换关系，但是由于各个位置不同的转换矩阵，最终的轨迹可能会与球体坐标系产生一定的偏差。这个偏差不是因为计算累积误差引起的，而是因为直角坐标和球体坐标的不同引起的，因此据此产生的参考轨迹与原始观测仍然是一致的，不会影响仿真平台输出的精确性。另外，如果载体运动范围不太大，这个误差可以忽略，而如果载体运动超过一定范围，建议做适当的调整。比如如果水平直线飞行达到几十公里以上，此时按照直线来说，高度会逐渐上升，载体会变成具有向上的速度。

这时候可以增加一段垂直加速运动进行调整，将水平速度变为 0 或者负值。这样相当于用多段直线拟合一个圆。

3. 原始观测量生成

原始观测量主要包括伪距、载波相位、多普勒和 C/N0。这里对上述观测量的产生进行逐一地阐述。在实际的接收机中，卫星信号的实际的码相位和载波相位是未知的，计算原始观测量是基于跟踪过程中产生的本地信号可以复现卫星信号，因此用本地信号的码相位和载波相位进行计算。在生成作为参考的无误差的原始观测量时，我们可以直接用已知的卫星信号计算。

伪距的计算公式是$\rho = c(T_r - T_t)$。因此，在不考虑钟差的情况下，本地时间（观测时刻）为 t 时接收到的卫星信号的发射时刻就是T_c。根据$T_c = t - T_p - \delta t_{iono}$，我们可以得到$\rho = cT_p + c\delta t_{iono}$。

载波相位的计算可以分为整周部分和小数部分，并且整周部分在基带跟踪时是连续计数的。在进行跟踪时卫星信号在观测时刻 t 的载波相位就是：

$$\Phi = \frac{1}{2\pi}[\omega_{IF}t - \omega_{RF}(T_p - \delta t_{iono})] = f_{IF}t - f_{RF}(T_p - \delta t_{iono})$$

通常，中频或者为 0，或者是整数赫兹（或 kHz）。而本地观测时刻也是秒（或者毫秒）的整数倍，这样可以保证两者的乘积为整数。由于本地生成的载波相位观测量可以与信号的载波相位相差整周，所以可以得到载波相位观测量为：

$$\hat{\Phi} = -\Phi + N = -f_{IF}t + f_{RF}(T_p - \delta t_{iono}) + N = -f_{IF}(t - t_0) + f_{RF}(T_p - \delta t_{iono})$$

其中$N = f_{IF}t_0$，表示在某初始时刻t_0时人为设置的整周数。

于是，当$t = t_0$时，就有$\hat{\Phi} = f_{RF}(T_p - \delta t_{iono}) = f_{RF}T_p - \frac{1}{\lambda}c\delta t_{iono}$

当$t = t_1$时，按照上述计算$\hat{\Phi}$的公式带入t_1计算时，可以发现公式的第一项$-f_{IF}\Delta t = -f_{IF}(t_1 - t_0)$是时间间隔内由中频产生的整周部分，因此需要扣除掉，所以依然可以得到$\hat{\Phi} = f_{RF}T_p - \frac{1}{\lambda}c\delta t_{iono}$。实际计算的时候，也可以人为地增加一个整周部分 N，从而将计算公式变为：$\hat{\Phi} = f_{RF}T_p - \frac{1}{\lambda}c\delta t_{iono} + N$。在没有发生周跳的时候，上述公式的 N 在各个历元间是不变的。如果发生了周跳，则 N 可以变化。同时也可以取 N 为半整数，意味着出现了半周模糊度。

多普勒的计算比较简单。首先，计算卫星与接收机之间的相对视向速度$v = (\overrightarrow{v_s} - \overrightarrow{v_r}) \cdot \hat{\boldsymbol{r}}$，其中$\overrightarrow{v_s}$是卫星速度矢量，$\overrightarrow{v_r}$是接收机速度矢量，$\hat{\boldsymbol{r}}$是接收机到卫星的视向单位矢量。于是就可以得到$f_D = -v/\lambda = \frac{1}{\lambda}(\overrightarrow{v_s} - \overrightarrow{v_r}) \cdot \hat{\boldsymbol{r}}$。如果卫星钟飘比较小的时候，可以忽略。但是如果考虑到卫星钟飘的话，就需要加上修正项。此时，多普勒的计算公式是

$$f_D = -v/\lambda = \frac{1}{\lambda}(\overrightarrow{v_s} - \overrightarrow{v_r}) \cdot \hat{\boldsymbol{r}} + f_{RF}[af_1 + af_2(t - t_{OC})]$$

或者

$$f_D = -v/\lambda = \frac{1}{\lambda}\{(\overrightarrow{\boldsymbol{v}_s} - \overrightarrow{\boldsymbol{v}_r}) \cdot \hat{\boldsymbol{r}} + c[af_1 + af_2(t - t_{OC})]\}$$

当af_2比较小时，该项也可以忽略（通常af_2的量级是小于 10^{-14}，$t - t_{OC}$的量级小于 10^4，因此相应的多普勒改正量小于 0.01Hz）。

C/N0 可以根据卫星仰角，或者用户预设的卫星信号功率计算策略进行调整。

4. 相关结果产生

再下一层次的信号是相关结果的产生。1ms 的相关结果（或者通过相干累加产生的更长时间的相关结果）可以用于鉴别器的输入，并产生用于控制跟踪环路的鉴频、鉴相和码延迟鉴别的结果。

实际上，按照 SignalSim 的主流程，只能够产生在观测时刻的卫星信号的码相位和载波相位，而相关器的输出则还会受到本地码相位、本地载波相位以及信号强度、噪底的影响。相关结果的输出产生的公式如下：

$$Cor_n = A \cdot R(|\Delta t + (n-p)\Delta t_I|) \cdot Sinc\left(\Delta\bar{f} \cdot \mathrm{T}\right) \cdot e^{j\Delta\bar{\phi}} + \sigma N$$

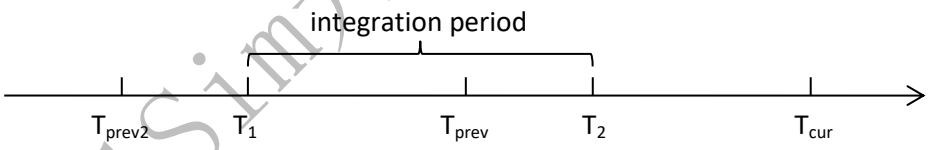
其中，n 代表相关器索引，A 是信号幅度，由信号功率决定。Δt是码相位差，Δt_I是相关器间距，p 表示峰值相关器索引。$\Delta\bar{f}$是积分时间段内的平均频差，T 是积分时间，通常是 1ms。$\Delta\bar{\phi}$是积分时间段内的平均相位差，也就是平均的卫星信号相位减本地载波相位。σ是噪声幅度，N 是标准高斯分布的复数白噪声。而 R()是相关函数，取值如下：

$$R(\Delta t) = \begin{cases} 1 - \dfrac{\Delta t}{T_c}, \Delta t \le T_c \\ 0, \Delta t > T_c \end{cases}$$

当信号带宽受限的时候，相关函数的形状会有变化，可以根据带宽进行调整。

积分时间段内的平均频率差$\Delta\bar{f}$和平均相位差$\Delta\bar{\phi}$通过以下方法进行计算：

由于本地码频率和信号码频率的差别不会很大，因此我们可以近似在积分周期内本地码相位和信号码相位的差固定不变，也就是可以取观测时刻的码相位差。而载波的频率差和相位差要按照下述方法进行计算：



在上图中，T_{cur} 表示当前观测时刻，T_{prev} 表示前一毫秒的观测时刻，T_{prev2} 表示再前一毫秒的观测时刻。而 T1 到 T2 表示一个完整的码周期，其积分值的结果在 T_{cur} 时刻输出。知道 T_{cur} 时刻的码相位，就可以计算得到码周期起始时刻 T_1 和结束时刻 T_2 相对于 T_{cur} 及 T_{prev} 的关系。

尽管载波相位和载波频率是联系在一起的，为了简便计算，我们对平均的载波频率差和平均的载波相位差分别计算。

设$\alpha = \frac{T_2 - T_{prev}}{T_{cur} - T_{prev}}$表示积分区间在最近 1 毫秒内所占的比例，或者等效的当前时刻在 1ms 内的码相位为$1 - \alpha$。

如果在上述三个观测时刻分别计算得到的信号多普勒分别是 $f_{D,cur}$、$f_{D,prev}$ 和 $f_{D,prev2}$，并且认为信号频率是线性变化的，则可以得到在 T_1 到 T_{prev} 和 T_{prev} 到 T_2 两个时间段上的平均信号多普勒分别是：

$$f_{D1} = f_{D,prev2} + \frac{1+\alpha}{2}\left(f_{D,prev} - f_{D,prev2}\right) = \frac{1-\alpha}{2} f_{D,prev2} + \frac{1+\alpha}{2} f_{D,prev}$$

$$f_{D2} = f_{D,prev} + \frac{\alpha}{2}\left(f_{D,cur} - f_{D,prev}\right) = \frac{2-\alpha}{2} f_{D,prev} + \frac{\alpha}{2} f_{D,cur}$$

如果在 T_{prev2} 和 T_{prev} 两个时刻，跟踪环路设置的本地载波频率分别是 $f_{LO,prev2}$ 和 $f_{LO,prev}$，则在 T1 到 T2 的时间段内，平均的频率差为：

$$\Delta\bar{f} = (1-\alpha)\left(f_{D1} - f_{LO,prev2}\right) + \alpha(f_{D2} - f_{LO,prev})$$

同样的，如果在三个时刻，信号的相位分别是分别为$\Phi_{s,prev2}$、$\Phi_{s,prev}$和$\Phi_{s,cur}$，而本地的以周为单位的载波相位分别为$\Phi_{LO,prev2}$、$\Phi_{LO,prev}$和$\Phi_{LO,cur}$，则可以分布计算三个时刻以弧度为单位的相位差$\Delta\phi = 2\pi(\Phi_s - \Phi_{LO})$，则平均的相位差为：

$$\Delta\bar{\phi} = (1-\alpha)\left(\frac{1-\alpha}{2}\Delta\phi_{prev2} + \frac{1+\alpha}{2}\Delta\phi_{prev}\right) + \alpha\left(\frac{2-\alpha}{2}\Delta\phi_{prev} + \frac{\alpha}{2}\Delta\phi_{cur}\right)$$

对于不同的相关器来说，$\Delta\bar{f}$和$\Delta\bar{\phi}$是相同的。但是，相关函数 R 的参数不同，计算得到不同的相关结果。

需要注意的是，各个相关器之间的噪声是具有相关性的，这是因为相邻的相关器的本地码具有一定的相关性。因此，在产生各个相关器的噪声的时候，也一定要注意这一点。由于噪声是复数，各个相关器之间的噪声的实部或者虚部的相关矩阵是这样的：

$$\mathbf{R} = \begin{bmatrix} 1 & 1-\Delta & 1-2\Delta & \cdots & 0 \\ 1-\Delta & 1 & 1-\Delta & \cdots & 0 \\ 1-2\Delta & 1-\Delta & 1 & \cdots & 0 \\ \vdots & \vdots & \vdots & \ddots & \vdots \\ 0 & 0 & 0 & \cdots & 1 \end{bmatrix}$$

也就是主对角元素为 1，表示自相关为 1；次对角元素为$1-\Delta$，表示相邻相关器之间噪声的相关系数为$1-\Delta$；再次对角元素为$1-2\Delta$，表示间隔 2 的相关器之间噪声的相关系数是$1-2\Delta$；以此类推。其中Δ是相关器间隔与码片间隔的比值。如果相关器间隔为 1/2 码片，则$\Delta = \frac{1}{2}$；相关器间隔为 1/8 码片，则$\Delta = \frac{1}{8}$。

由于上述相关矩阵$\mathbf{R}$为对称正定阵，因此可以将上述矩阵做 Cholesky 分解，得到三角阵$\mathbf{C}$。于是有$\mathbf{R} = \mathbf{C} \cdot \mathbf{C}^{\boldsymbol{T}}$。当需要产生相关的 N 个复数噪声时，首先产生 N 对独立同分布的高斯白噪声形成 N 个复数噪声，然后将三角阵$\mathbf{C}$与产生的 N 个复数噪声进行相乘，就可以得到符合相关关系为$\mathbf{R}$的 N 个相关噪声了。

5. 中频采样信号

通过计算特定时刻的码相位和载波相位，根据第一章中的公式，可以计算得到每一个采样时刻的中频信号。但是，为了提高计算速度，不会计算所有采样时刻的码相位和载波相位，因为这样即使仅仅计算 16MHz 的采样频率的信号都会需要庞大的运算量。

实际实现中，会每隔一定的时间间隔计算一次码相位和载波相位，时间间隔以内的部分，按照均匀插值来进行计算。如果动态比较大的话，时间间隔可以取 1ms，如果动态比较小的话，时间间隔可以取 10ms。

6. XML 配置文件解析说明

为了更好地将配置信息组织起来，配置文件采用 XML 的格式。采用 XML 格式组织配置信息有以下的好处：首先，XML 文件可以形成层次结构，这样便于将各种不同的配置和输入信息分类，形式上比较清晰；其次，XML 文件具备可扩展性，如果需要增加配置数据的内容、扩展更多的类型等等，只需要简单增加所支持的类型；第三，XML 文件可以做到比较好的向后兼容性，未来进行扩展后原有的配置文件可以不需要修改就能被未来的程序所支持。

对于 XML 配置文件的组织以及解析方式说明如下：

6.1 XML 文件结构及文件内容读取

XML 文件将信息以树状结构以文本的方式组织起来，因此很适合于用来分类组织复杂的配置信息。XML 文件除了文件头和注释行以外，一般有唯一的一个根节点（根元素），其下以嵌套的形式包含若干的子元素，而子元素还可以包含若干更下一个层次的子元素。每一个元素有用尖括号包含的一个元素标签名（tag）和一个元素内容（text），同时，每一个元素还可以有若干的属性说明。

文件 XmlElement.cpp 包含的几个类实现了对 XML 文件内容的读取。

上述内容在程序中组织的时候，用类 CXmlElementTree 来组织一个 XML 文件的内容，包括了文件头，注释行和指向跟元素的指针。类成员函数包括了从文件读入数据和将数据写入文件的操作。

从根节点开始，所以的元素按照树的形式进行组织。每一个节点元素用类 CXmlElement 存放。所有节点的叶子节点以链表的形式连在一起，每个节点指向链表中下一个节点的指针是 SiblingElements。父节点通过指针 ChildElements 指向链表头，也就是第一个子节点。如果一个节点没有叶子节点，则 ChildElements 为空指针，子节点链表的最后一个节点 SiblingElements 为空指针。

元素有一个字符串存放 tag 和一个字符串存放 text，另外还有一个简单的字典类（CSimpleDict）的实例用来维护所有的属性。一个元素的属性可以有多个字典条码，每一个字典条目包含了一对键（key）和值（value）的组合。当元素中的字典条目增加时，其内存可以自动扩展。

通常来说，配置值在 XML 文件中以叶子节点的形式存放，同一类配置值组织成一个父节点下的若干子节点。配置名称作为元素的 tag，配置值作为元素的 text。元素的属性通常用来表示元素内容的类型、格式、单位等（如角度的单位是弧度还是度）。

6.2 XML 文件数据解析

源代码 XmlInterpreter.cpp 用来将读取的 XML 的文件内容进行解析，并赋值到相应的结构体中。

6.3 轨迹处理和位置计算

接收机轨迹用类 CTrajectory 来维护，该类维护了 ECEF 坐标系下的初始位置、速度以及本地坐标系下的初始速度，并且用一个链表组织了所有的轨迹段。在初始化的时候，轨迹类首先初始化初始位置和初始速度，然后调用 AppendTrajectory()在链表末端添加新的轨迹段。该函数计算上一个轨迹段的结束位置和速度作为新添加的轨迹段的起始位置速度，并用相应的参数用 InitSegment()函数对轨迹段进行初始化后添加到链表的末尾。

通常设置一个轨迹段的可以有多种参数类型组合，如匀加速直线运动就可以任意取持续时间，加速度和末速度种任意的两个进行组合，而相应轨迹段的 SetSegmentParam()函数会将不同的参数组合转换为统一的参数组合对轨迹段进行初始化。

在提取轨迹中的坐标时，首先调用 ResetTrajectoryTime()将当前时刻初始化为轨迹的起始时刻，然后调用 GetNextPosVelECEF()或 GetNextPosVelLLA()获得给定间隔时间后的位置和速度。计算位置和速度的计算方法是首先将当前时刻向后推给定的时间间隔，然后计算新的当前时刻对应的轨迹段和在轨迹段内相对于轨迹段起始时刻的时间偏移，最后根据该轨迹段的起始位置和速度以及时间偏移计算当前时刻的位置和速度。

所有的轨迹段以链表的形式进行组织，不同类型的轨迹段都是虚基类 CTrajectorySegment 的派生类。基类中记录了该轨迹段的起始位置、速度以及轨迹段的时间长度，不同的轨迹段类型派生类分别记录相应的其他参数，并且也对设置参数的虚函数 SetSegmentParam()以及计算轨迹段内某时刻位置速度的函数 GetPosVel()提供了相应的具体实现。

6.4 导航电文读取和合成

导航电文的读取由 CNavData 类实现。该类通过读取 RINEX 文件获得星历数据并维护了一个动态的 pool 用于存放星历，同时可以根据给定的时间查找合适使用的星历参数。

对于生成原始观测量的应用来说，仅用到导航电文的读取功能。对于基带数据产生和中频数据产生的应用，因为涉及到数据比特的调制，因此还会需要用到导航电文的合成功能。

导航电文和合成是由 NavBit 类实现的。这是一个包含纯虚函数的虚基类，提供了用于导航电文合成所需要的基本底层函数和对外的接口虚函数。不同的导航电文是由 NavBit 类的不同的派生类来实现的。不同的派生类在合成导航电文码流的时候，使用基类提供的统一的函数接口。

对于包含导频通道的信号来说，导频通道的 PRN 码上往往包含 Secondary Code 作为二次调制的信号。这部分功能由 Pilot Bit 调制函数产生。

7. SignalSim 主要接口函数说明

下面将 SignalSim 软件包提供的主要接口函数分为几大类，分别是和场景配置文件处理相关的函数、接收机轨迹相关的函数、和原始观测量产生相关的函数、和导航电文的读取和合成相关的函数以及辅助类函数等，下面分别加以说明。

7.1 场景配置文件处理函数

对 XML 文件的读取和解析由 CXmlElementTree 类实现，最常使用的成员函数为：

- parse()：输入参数为 XML 文件名，实现对文件内容的读取和解析。读取的 XML 文件存放在以树组织的 CXmlElement 类的实例中。
- getroot()：返回存储的 XML 文件内容树的根节点。

在 XML 文件中，每一个元素（或节点）是以以下形式表示的：<TagName attribute1=”attr1” attribute2=”attr2” …>Text</TagName>，其中 attribute1、attribute2 等等为属性的 key，attr1、attr2 等为对应属性的值。属性值可以没有或者有多个。如果元素节点是叶子节点，则 Text 表示元素内容；如果元素节点不是叶子节点，则 Text 的内容是各个子节点的列表。元素类为 CXmlElement，常用的类成员函数为：

- GetElement()：输入参数为子节点索引，返回指向子节点的实例指针。如果索引值大于子节点个数则返回空指针。
- GetTag()：返回对应元素的 TagName 字符串。

对场景文件的解析由以下函数实现：

- AssignStartTime()：参数如下：
  - ✓ Element：指向包含起始时间的元素指针。
  - ✓ UtcTime：起始时间的引用，用于存放解析的起始时间。
- SetTrajectory()：参数如下：
  - ✓ Element：指向包含接收机轨迹的元素指针。
  - ✓ StartPos：初始位置的引用，用于存放解析的初始位置。
  - ✓ StartVel：初始速度的引用，用于存放解析的初始速度。
  - ✓ Trajectory：接收机轨迹实例的引用，用于存放解析的接收机轨迹列表。
- SetOutputParam()：参数如下：
  - ✓ Element：指向包含输出控制的元素指针。
  - ✓ OutputParam：输出控制结构体的引用。
- SetPowerControl()：参数如下：
  - ✓ Element：指向包含功率控制的元素指针。
  - ✓ PowerControl：功率控制类实例的引用。
- SetDelayConfig()：参数如下：
  - ✓ Element：指向包含延时参数的元素指针。

- ✓ DelayConfig：延时参数结构体的引用。

7.2 和接收机轨迹相关的函数

和接收机轨迹相关的函数由 CTrajectory 类实现，类接口函数

- ResetTrajectoryTime()：重置接收机轨迹时间到轨迹的起始。
- GetNextPosVelECEF()：得到下一个时间点的接收机位置和速度。第一个参数是步进的时间间隔，以秒为单位。第二个参数是 ECEF 坐标系下位置和速度结构体的引用，用于存放接收机的位置和速度。

7.3 和原始观测量产生相关的函数

原始观测量的产生主要依赖于 SATELLITE_PARAM 结构体，结构体中包含卫星状态以及用于计算原始观测量的相关信息。部分信息依赖于接收机位置。结构体的成员变量如下：

- system：卫星星座
- svid：卫星号
- FreqID：GLONASS 卫星的频点号
- CN0：卫星的载噪比，以 0.01dB 为单位
- PosTimeTag：通过外推预测方式计算卫星位置时外推起始的时间
- PosVel：卫星当前位置速度
- Acc：卫星当前加速度
- TravelTime： 卫星信号传播时间，包含除电离层延时外其他所有改正量：对流层、星钟、相对论改正等
- IonoDelay：电离层延时
- GroupDelay：主频点群延时以及其他频点相对主频点的相对延时
- Elevation：卫星仰角
- Azimuth：卫星方位角
- RelativeSpeed： 卫星和接收机的相对距离变化率
- LosVector：Line of Sight 单位矢量

用于得到上述结构体的值的函数包括：

- GetSatelliteParam()：计算卫星信息结构体，参数包括：
  - ✓ PositionEcef：ECEF 坐标系下的接收机位置（主要用于计算几何距离和相对速度）
  - ✓ PositionLla：接收机位置的经纬高（主要用于计算对流层电离层延时），最好和接收机位置的 ECEF 坐标一致
  - ✓ time：当前 GPS 时间
  - ✓ system：卫星星座

- ✓ Eph：指向星历结构体的指针，如果是 GLONASS，需要将指针类型做强制转换
- ✓ IonoParam：指向 Ionosphere 改正参数结构体的指针
- ✓ SatelliteParam：指向存放返回值的 SATELLITE_PARAM 结构体指针

- GetSatelliteCN0()：计算卫星载噪比 ，参数包括：
  - ✓ PowerListCount：卫星载噪比调整列表长度
  - ✓ PowerListCount：卫星载噪比调整列表
  - ✓ DefaultCN0：CN0 默认值
  - ✓ Adjust：CN0 随仰角调整方式
  - ✓ SatelliteParam：指向存放返回值的 SATELLITE_PARAM 结构体指针，其中的星座、svid 和仰角用于计算当前 CN0，得到的结果会用来为 CN0 字段赋值
- GetTravelTime()：返回卫星信号传播时间（秒），参数包括：
  - ✓ SatelliteParam：指向存放卫星状态参数结构体的指针
  - ✓ FreqIndex：信号频点，在 SatelliteParam.h 中定义
- GetCarrierPhase()：返回卫星信号载波相位（周），参数同上
- GetDoppler()：返回卫星即时多普勒（Hz），参数同上

7.4 导航电文读取和合成

导航电文的读取由 CNavData 类实现。该类通过读取 RINEX 文件获得星历数据并维导航电文的读取由 CNavData 类实现。该类通过读取 RINEX 文件获得星历数据并维护了一个动态的 pool 用于存放星历，同时可以根据给定的时间查找合适使用的星历参数。

CNavData 类提供了如下常用的成员函数：

- ReadNavFile()：参数为 RINEX 格式 navigation 数据文件的文件名，用于从文件中加载星历、电离层改正参数等内容。
- FindEphemeris()：参数为卫星星座、GPS 时、svid，返回星历 pool 中适合的星历。如果没有找到则返回空指针。
- FindGloEphemeris()：参数为 GPS 时、卫星 slot，返回星历 pool 中适合的 GLONASS 星历。如果没有找到则返回空指针。
- GetGlonassSlotFreq()：参数为 slot，返回 GLONASS 卫星相应 slot 对应的频点索引。
- GetGpsIono(), GetBdsIono(), GetGalileoIono()：返回相应卫星系统电离层改正参数的指针。
- GetGpsUtcParam(), GetBdsUtcParam(), GetGalileoUtcParam()：返回相应卫星系统 UTC 改正和闰秒参数的指针。

导航电文合成目前实现了 GPS 的 LNAV、BDS 的 D1/D2 和 B-CNAV1、Galileo 的 I/NAV 和 GLONASS 的 GNAV 数据调制的合成。通过以下基类统一的成员函数接口得到导航电文的调制数据：

- SetEphemeris()：参数为 svid 和星历指针，如果是 GNAV，则第一个参数为 slot，第二个参数的指针类型需要进行强制转换。
- SetAlmanac()：参数为 svid 和历书指针，对于不同的历书类型，指针类型需要进行强制转换。
- SetIonoUtc()：参数为指向电离层改正参数和 UTC 参数的指针。
- GetFrameData()：得到相应的 subframe/page 的调制比特流。参数 StartTime 为相应系统的系统时间（GLONASS 在 Week 中存放天数，在 MilliSeconds 中存放整数的天内毫秒），参数 svid 为卫星号（slot），NavBits 为存放 0/1 序列的数组，数组需要在调用函数中分配，数组大小由相应的导航电文类型决定。系统时间不要求对齐一个 subframe/page 的边沿，函数会返回给定时间点所在的一个完整的 subframe/page。原则上每一次合成的导航电文长度为包含完整校验和交织的一段导航电文，如下表中说明：

导航电文类型	电文长度	说明
LNAV	300	一个子帧长度
CNAV	600	
CNAV2	1800	
D1/D2	300	
B-CNAV1	1800	
B-CNAV2	600	
B-CNAV3	1000	
I/NAV	250	一个 page 的长度
F/NAV	500	
GNAV	200	进行 meander code 调制后的一个 page

7.5 卫星信号生成

不同星座和频点的卫星具有不同的调制方式、导航电文、功率分配等特性。因此 CSatelliteSignal 类用于产生对应卫星的信号。生成相应的卫星信号前需要先调用函数 SetSignalAttribute()进行配置，函数的参数如下：

- ✓ System：卫星星座（BasicTypes.h 中定义）
- ✓ FreqIndex：信号的频点（SatelliteParam.h 中定义）
- ✓ pNavData：指向对应的导航电文合成类实例的指针。实例的类型需要和相应的星座和频点一致，否则函数会返回 FALSE。如果不需要电文调制，可以提供空指针，则产生的信号的数据通道的电文调制为全零。
- ✓ svid：卫星 SVID（GLONASS 卫星为 slot number）

配置好信号属性后，就可以调用函数 GetSatelliteSignal()产生相应时间点上的信号。由于信号码周期和数据/导频调制的最小间隔为 1ms，因此函数产生的是相应时间点上在数据/导频信号上的调制幅度和相位。函数的第一个参数是 GPS 时，其他的卫星系统会自动进行相应的转换以得到各自系统的时间。第二个和第三个参数分别是数据和导频上的调制数据变量的引用，调制数据都是复数。产生的规则是：对于相位来说，如果该信号只有数据分量，则数据调制为 0 和 1 对应的相位分别是 0 和π。如果存在导频分量，则导频上的二次编码为 0 和 1 对应的导频信号的相位分别是 0 和π，数据信号的相位按照与导频信号的相位关系确定。对于幅度来说，按照信号总能量为 1 时，导频和数据各自的能量分配确定各自的幅度，如果还有 BOC(6,1)调制，则会在总能量中扣除 BOC(6,1)分量的能量。

7.6 其他辅助函数

其他辅助函数包括对根据星历计算卫星位置/速度/加速度的函数（Coordinate.h）、UTC 时间和各个卫星系统时间转换的函数（GnssTime.h）、对 RINEX 文件进行读写和解析的函数（Rinex.h）等，是上述软件包的底层支持函数，也可以直接进行调用。

8. SignalSim 对 RINEX 4 文件的支持

新版的 RINEX 4 文件扩展了对不同频率下的不同格式的导航电文的支持，从原来每个卫星系统只有一种导航电文扩展到了支持所有频点的不同格式的导航电文。对于 GPS 来说，新增加了 L5 频点上的 CNAV 和 L1C 频点上的 CNAV2 格式；对于 BDS，增加了 B1C 频点上的 CNAV1、B2a 频点上的 CNAV2 和 B2b 频点上的 CNAV3 格式；Galileo 除了在 source 字段指明数据来源于 I/NAV 还是 F/NAV 以外，也在电文头中给出了指示。

新的导航电文相比于原来的导航电文，在卫星轨道和星钟参数上基本一致，但是有以下三个比较重要的不同点：第一是卫星轨道参数新增了 A_dot 和 delta_n_dot 两个新的参数；第二是除了 TGD 以外，还增加了信号间延迟改正项 ISC；第三是一些标志位，包括健康标志、精度标志等存在不同。

针对上述的不同，SignalSim 也做了相应的升级，以支持的新的格式的导航电文的文件以及对星历数据的使用。

8.1 RINEX 文件的读取

在读取导航电文文件的时候，程序会自动判断数据段是原来 RINEX 3 格式的电文数据还是以 RINEX 4 格式数据头开始的数据段，然后根据读取的结果自动判断导航电文类型并进行赋值。为了统一起见，除 GLONASS 和 SBAS 的导航电文外，所有其他的星历数据使用同样的星历数据结构体，同时 QZSS 的相应的电文类型由于和 GPS 基本一致，因此不做区分。不同格式的星历数据在 RINEX 导航电文文件中的排列如下表：

		GPS/QZSS			Galileo	BDS				NavIC
line		LNAV	CNAV	CNV2	I/FNAV	D1/D2	CNAV1	CNAV2	CNAV3	LNAV
0	Time	t_{oc}	t_{oc}	t_{oc}	t_{oc}	t_{oc}	t_{oc}	t_{oc}	t_{oc}	t_{oc}
	0	af0	af0	af0	af0	af0	af0	af0	af0	af0
	1	af1	af1	af1	af1	af1	af1	af1	af1	af1
	2	af2	af2	af2	af2	af2	af2	af2	af2	af2
1	3	IODE	A_{dot}	A_{dot}	IOD_{nav}	AODE	A_{dot}	A_{dot}	A_{dot}	IODEC
	4	C_{rs}	C_{rs}	C_{rs}	C_{rs}	C_{rs}	C_{rs}	C_{rs}	C_{rs}	C_{rs}
	5	Δn	Δn0	Δn0	Δn	Δn	Δn0	Δn0	Δn0	Δn
	6	M0	M0	M0	M0	M0	M0	M0	M0	M0
2	7	C_{uc}	C_{uc}	C_{uc}	C_{uc}	C_{uc}	C_{uc}	C_{uc}	C_{uc}	C_{uc}
	8	e	e	e	e	e	e	e	e	e
	9	C_{us}	C_{us}	C_{us}	C_{us}	C_{us}	C_{us}	C_{us}	C_{us}	C_{us}
	10	sqrtA	sqrtA	sqrtA	sqrtA	sqrtA	sqrtA	sqrtA	sqrtA	sqrtA
3	11	t_{oe}	t_{op}	t_{op}	t_{oe}	t_{oe}	t_{oe}	t_{oe}	t_{oe}	t_{oe}
	12	C_{ic}	C_{ic}	C_{ic}	C_{ic}	C_{ic}	C_{ic}	C_{ic}	C_{ic}	C_{ic}
	13	Ω_0	Ω_0	Ω_0	Ω_0	Ω_0	Ω_0	Ω_0	Ω_0	Ω_0
	14	C_{is}	C_{is}	C_{is}	C_{is}	C_{is}	C_{is}	C_{is}	C_{is}	C_{is}
4	15	i_0	i_0	i_0	i_0	i_0	i_0	i_0	i_0	i_0
	16	C_{rc}	C_{rc}	C_{rc}	C_{rc}	C_{rc}	C_{rc}	C_{rc}	C_{rc}	C_{rc}
	17	ω	ω	ω	ω	ω	ω	ω	ω	ω
	18	Ω_{dot}	Ω_{dot}	Ω_{dot}	Ω_{dot}	Ω_{dot}	Ω_{dot}	Ω_{dot}	Ω_{dot}	Ω_{dot}
5	19	i_{dot}	i_{dot}	i_{dot}	i_{dot}	i_{dot}	i_{dot}	i_{dot}	i_{dot}	i_{dot}
	20	CodeL2	$\Delta n0_{dot}$	$\Delta n0_{dot}$	source		$\Delta n0_{dot}$	$\Delta n0_{dot}$	$\Delta n0_{dot}$	
	21	Wn	URA_{NED0}	URA_{NED0}	Wn	Wn	SatType	SatType	SatType	Wn
	22	L2P	URA_{NED1}	URA_{NED1}			t_{op}	t_{op}	t_{op}	
6	23	Accuracy	URA_{ED}	URA_{ED}	SISA	Accuracy	$SISAI_{oe}$	$SISAI_{oe}$	$SISAI_{oe}$	URA
	24	health	health	health	health	health	$SISAI_{ocb}$	$SISAI_{ocb}$	$SISAI_{ocb}$	health
	25	T_{GD}	T_{GD}	T_{GD}	B_{GDE5a}	T_{GD1}	$SISAI_{oc1}$	$SISAI_{oc1}$	$SISAI_{oc1}$	T_{GD}
	26	IODC	URA_{NED2}	URA_{NED2}	B_{GDE5b}	T_{GD2}	$SISAI_{oc2}$	$SISAI_{oc2}$	$SISAI_{oc2}$	
7	27	T_{tr}	ISC_{L1CA}	ISC_{L1CA}	T_{tr}	T_{tr}	ISC_{B1Cd}		SISMAI	T_{tr}
	28	fit interval	ISC_{L2C}	ISC_{L2C}		AODC		ISC_{B2ad}	health	
	29		ISC_{L5I}	ISC_{L5I}			TGD_{B1Cp}	TGD_{B1Cp}	B2b flag	
	30		ISC_{L5Q}	ISC_{L5Q}			TGD_{B2ap}	TGD_{B2ap}	T_{GDB2b}	
8	31		T_{tr}	ISC_{L1Cd}			SISMAI	SISMAI	T_{tr}	
	32		Wn_{op}	ISC_{L1Cp}			health	health		
	33						B1C flag	Integ. Flag		
	34						IODC	IODC		
9	35			T_{tr}			T_{tr}	T_{tr}		
	36			Wn_{op}						
	37									
	38						IODE	IODE		

Same meaning
Similar meaning
Bit field

根据新的导航电文的不同之处，首先在星历结构体中增加了 A_dot 和 delta_n_dot 两个参数。对于不存在这两个参数的导航电文会给两个参数赋值为 0，在通过星历计算卫星位置的程序中，这两个值为 0 不会影响原先的计算公式和计算结果。

对于新增加的信号间延迟改正项，根据各个卫星系统以及各个信号的 ICD 中给出的计算公式，不同频点上卫星时钟误差的计算公式统一使用$(\Delta t_{SV})_{(i)} = \Delta t_{SV} - T_{GD(i)}$其中下标 i 表示不同的频点和信号通道。因为信号和通道的增加，因而T_{GD}的个数也需要相应地增加。为了保持和原来的程序以及结构体的兼容性，原有的 tgd 和 tgd2 的字段保持不变，新增了 tgd_ext[5]数组用于存储不同信号通道的群延迟，数组元素对应的信号通道将在后面说明。

对于不同信号的标志，星历结构体将 flag 和 health 两个字段从原来的 8bit 变成 16bit，对于不同的卫星系统，两个字段的标志排列如下表所示：

flag

	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
GPS													fit	L2P	Code L2	
Galileo							TOC ref							source		
BDS						B2b integ. flags			B2a integ. flags			B1C integ. flags			SatType	

health

	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
GPS							L1C	L1	L2	L5	LNAV health					
Galileo								E5b HS/DVS			E5a HS/DVS			E1B HS/DVS		

8.2 不同来源导航电文的相互转换

对于 SignalSim 的用户来说，有可能输入的导航电文的数据格式不全，比如只有 BDS D1/D2 和 B2a 的导航电文，但是需要仿真 B1C 上的信号；或者只有 GPS RINEX 3 的导航数据文件，但是需要仿真 GPS 全频点的信号。在这种情况下，某一数据来源的星历会需要补全数据用于其他频点的电文调制。

对于星钟和轨道参数来说，补全或忽略 A_dot 和 delta_n_dot 后，就可以直接计算调制电文了。对于标志字段来说，flag/health 是针对整个卫星系统的，因此也可以直接使用。需要特别注意的是T_{GD}的部分。当读取星历数据的时候，RINEX 数据处理程序会补齐其他所需要的群延迟数据，这样在进行相应频点的观测量计算，以及在用相应的数据产生所需的调制电文的时候，就可以直接使用或者进行推算了。

下标给出了不同卫星系统的不同信号通道所使用的相应的变量值，以及在读取 RINEX 星历进行赋值的时候的计算公式：

			tgd_ext				
	tgd	tgd2	0	1	2	3	4
	L1CA B1I E1	L2C B2I	L1Cd B1Cd	L1Cp B1Cp	L5I B2ad E5a	L5Q B2ap	B2bI E5b
LNAV	T_{GD}	γT_{GD}	T_{GD}	T_{GD}	$\gamma_{L5} T_{GD}$	$\gamma_{L5} T_{GD}$	-
CNAV	$T_{GD} - \mathrm{ISC}_i$	$T_{GD} - \mathrm{ISC}_i$	$T_{GD} - \mathrm{ISC}_i$	$T_{GD} - \mathrm{ISC}_i$	$T_{GD} - \mathrm{ISC}_i$	$T_{GD} - \mathrm{ISC}_i$	-
CNAV2	$T_{GD} - \mathrm{ISC}_i$	$T_{GD} - \mathrm{ISC}_i$	$T_{GD} - \mathrm{ISC}_i$	$T_{GD} - \mathrm{ISC}_i$	$T_{GD} - \mathrm{ISC}_i$	$T_{GD} - \mathrm{ISC}_i$	-
D1/D2	T_{GD1}	T_{GD2}	T_{GD1}	T_{GD1}	$\gamma_{L5} T_{GD1}$	$\gamma_{L5} T_{GD1}$	$\gamma_{B2b} T_{GD1}$
CNAV1	TGD_{B1C}	$\gamma_{B2I} TGD_{B1C}$	$TGD_{B1C} - \mathrm{ISC}$	TGD_{B1C}	TGD_{B2a}	TGD_{B2a}	$\gamma_{B2b} TGD_{B1}$
CNAV2	TGD_{B1C}	$\gamma_{B2I} TGD_{B1C}$	TGD_{B1C}	TGD_{B1C}	$TGD_{B2a} - \mathrm{ISC}$	TGD_{B2a}	$\gamma_{B2b} TGD_{B1}$
CNAV3	TGD_{B2bI}/γ_{B2b}	TGD_{B2bI}	TGD_{B2bI}/γ_{B2b}	TGD_{B2bI}/γ_{B2b}	TGD_{B2bI}	TGD_{B2bI}	TGD_{B2bI}
I/NAV	$BGD_{(E1,E5a)}$	$BGD_{(E1,E5b)}$	-	-	$\gamma_{E5a} BGD_b$	-	$\gamma_{E5b} BGD_b$
F/NAV	$BGD_{(E1,E5a)}$	$BGD_{(E1,E5a)}$	-	-	$\gamma_{E5a} BGD_a$	-	$\gamma_{E5b} BGD_a$

表中绿色表示采用RINEX4电文中相应的值进行赋值，蓝色表示如果没有相应电文，则用于替代的值，-表示没有相应频点或者不进行赋值。

8.3 SignalSim 函数的调用方式

由于一颗卫星在不同频点上存在着多组不同的电文，所以用户在调用 SignalSim 软件包的相关函数时也增加了灵活性和复杂度。

在调用 SignalSim 的函数完成所需功能时，在读取了 RINEX 文件以后，对于产生不同频点上的信号的导航电文时，需要首选来自于本频点信号的星历电文。因此 CNavData 类的成员函数 FindEphemeris()增加了一个额外的参数表示首选的星历来源以匹配星历结构体中的 source 字段的内容。调用时会优先匹配来自同一频点的星历数据。如果不指定该参数，则使用缺省值 0，相应的定义为原来的 RINEX 3 文件中的 LNAV/D1D2 的星历数据，与原有的接口和功能兼容。

下面的代码给出为 B1I 和 B1C 两个频点的信号生成相应导航电文的示例：

```
for (i = 1; i <= TOTAL_BDS_SAT; i ++)
{
    BdsEphemeris = NavData.FindEphemeris(BdsSystem, CurTime, i);
    NavBitArray[0]->SetEphemeris(i, BdsEph[i - 1]);
    BdsEphemeris = NavData.FindEphemeris(BdsSystem, CurTime, i,
EPH_SOURCE_CNV1);
    NavBitArray[1]->SetEphemeris(i, BdsEph[i - 1]);
}
```

上述代码遍历卫星号，分别为指向 D1D2NavBit 类实例的指针 NavBitArray[0]和 BCNavBit 类实例的指针 NavBitArray[1]设置相应格式的星历数据。相应的类指针可以在为相应信号 CSatelliteSignal 的实例调用函数 SetSignalAttribute()时作为参数进行传递。这样就优先选择相应来源的星历用于相应频点了。

而对于不同频点的观测量生成，由于模型中卫星位置在某一时刻的真值只有一个，也即是说只能用其中的一组星历参数，因此需要用户根据自己的需求，为一颗卫星选择唯一的一组星历参数将指针填入 Eph[]数组并传送给 GetVisibleSatellite()函数，得到可见星列表并获得从中挑选的可见卫星的星历指针 EphVisible[]。调用函数 GetSatelliteParam()的时候也使用同样的星历作为参数 Eph。

这是因为，不同频点播发的星历参数计算得到的卫星位置可能存在微小的差异。如果不同频点的参数采用不同的星历参数计算卫星位置的话，会为模型带来未知的误差。也就是频点间的延迟不仅来源于模型参数本身计算得到的群延迟差和依靠电离层模型计算的延迟差，还有不同轨道参数带来的卫星位置差。而后者在观测量处理和解算的时候是不补偿的。

Annex A 原始观测量生成及验证

原始观测量生成程序 XmlObsGen 实现了从配置文件读取信息，并且产生原始观测量以及参考位置输出。两个配置文件 test_pos.xml 和 test_obs.xml 分别用来产生用于参考的接收机位置文件 reference.pos 和原始观测量文件 test.o。参考接收机位置是通过轨迹计算得到的参考位置，一方面可以用来验证轨迹设置的正确性，另一方面可以用来作为定位解算结果的参考。

参考位置文件可以直接被 RTKLIB 工具集中的 RTKPLOT 工具显示。原始观测量文件 test.o 可以通过 RTKPOST 进行处理解算，配置为单点定位输出，得到的定位结果 test.pos 也可以在 RTKPLOT 中进行显示。下图是 RTKPLOT 工具给出的参考位置和原始观测量经过计算得到的定位结果的差，两者的差异非常小（造成差异的主要原因是原始观测量生成和 RTKLIB 所采用的对流层模型不同）。