Задание на курсовой проект
по дисциплине
“Аппаратура потребителей СРНС”

1 Введение

В курсовом проекте рассматривается модель навигационной системы, включающая в себя модель космического сегмента — навигационных спутников, и модель наземного потребителя. Общая модель системы разбита на множество блоков, выполняющих отдельные функции, такие, как слежение за фазой, декодирование навигационных данных и т.д. Необходимо реализовать некоторые из этих функций в соответствии со своим вариантом задания.

1.1 Задание

Цель работы — создать в среде Matlab программу, реализующую модель одного из блоков системы спутниковой навигации (тип блока в соответствии с вариантом).

Для достижения поставленной цели работы необходимо решить следующие *задачи*:

1. изучить принципы работы модели системы навигации в целом;
2. изучить принципы работы рассматриваемого блока;
3. изучить программный интерфейс блока;
4. разработать модель рассматриваемого блока;
5. провести моделирование;
6. показать работоспособность модели общей системы при использовании разработанного блока;
7. оформить и защитить результаты.

Структура отчёта должна соответствовать поставленным задачам.

Варианты заданий приведены в табл. 1.

2 Описание системы

Рассматривается гипотетическая система спутниковой радионавигации, основанная на тех же принципах, что и реальная система ГЛОНАСС/GPS, но в упрощённом виде.

2.1 Системная шкала времени

В каждой системе спутниковой радионавигации используется единая для всех спутников системная шкала времени. Формирование сигналов и привязка эфемеридных данных осуществляется по этой шкале времени.

В данной работе эфемериды спутников представлены в виде данных в формате TLE. Эфемериды в формате TLE привязаны к номеру секунды с начала года. Поэтому в качестве системной шкалы времени будет использоваться номер текущей секунды в пределах года.

№	Рассматриваемый блок
1	Расчёт положения НС на основе данных TLE
2	Формирование сообщения
3	Расчёт дальности и формирование сигнала
4	Радиочастотный блок
5	Коррелятор
6	Блок поиска
7	Некогерентный режим слежения
8	Когерентный режим слежения
9	Символьная синхронизация и демодуляция
10	Кадровая синхронизация
11	Измерение псевдодальности и расчёт положения НС
12	Блок решения навигационной задачи
13	Кодирование и декодирование CRC-8
14	Расчёт направления на НС

Таблица 1: Варианты заданий

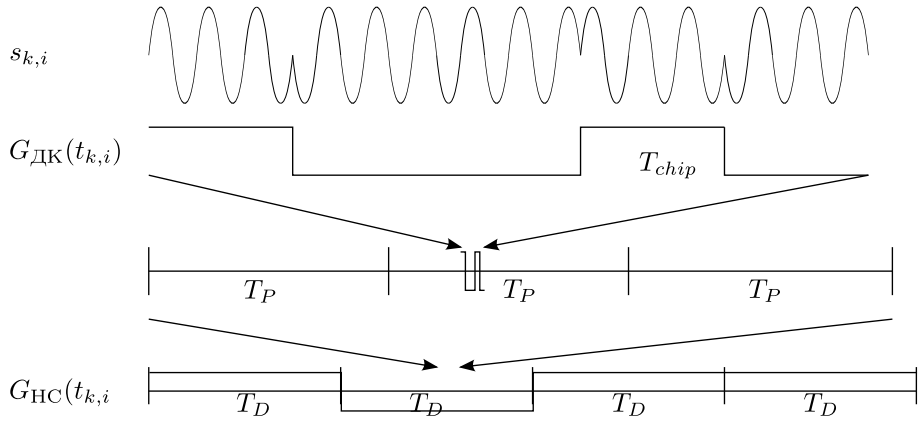

Рис. 1: Структура навигационного сигнала

2.2 Структура навигационного сигнала

Навигационный сигнал имеет структуру следующего вида:

$$s_{k,i} = AG_{\text{ДК}}(t_{k,i} - \tau_k)cos((\omega_0 + \omega_{\text{доп},k})t_{k,i} + \varphi_k + \pi \cdot \theta_{\text{НС}}(t_{k,i} - \tau_k))$$

где A — амплитуда сигнала, $G_{\text{ДК}}(t)$ — функция модуляции дальномерным кодом, ω_0 — номинальная частота сигнала, τ_k — задержка сигнала на k-ом шаге, $\omega_{\text{доп},k}$ — частота сигнала на k-ом шаге, φ_k — фаза сигнала на k-ом шаге, $\theta_{\text{НС}}(t)$ — функция модуляции навигационным сообщением. В данном описании сигнала использована двойная отсчётов сигнала, $t_{k,i} = T \cdot k + T_{\text{Д}} \cdot i$, где T — интервалы времени, на которых параметры сигнала считаются постоянными, $T_{\text{Д}}$ — интервалы дискретизации.

Структура сигнала приведена на рис. 1. Длина ПСП составляет $T_P = 1$ мс, количество символов ПСП $N = 1023$, значит продолжительность одного символа составляет $T_{chip} \approx 978$ мкс. Номинальная несущая частота сигнала $f_0 = 1575.42$ МГц.

Формирование навигационного сигнала строго привязано к системной шкале времени:

- длительность ПСП составляет $T_P = 1$ мс, поэтому положение в пределах ПСП можно получить, разделив текущее время по модулю T_p;
- длительность символа данных составляет $T_D = 5$ мс, положение символа можно получить, разделив текущее время по модулю T_D;
- длительность кадра сообщения составляет 1200 символов (6 с), поэтому положение в пределах кадра можно получить, разделив время по модулю 6 с.

Иначе говоря, нулевой кадр начинается в момент времени $t = 0$ с, первый кадр начинается в момент времени $t = 6$ с. В момент времени $t = 60300$ с начинается 100500-й кадр сообщения.

2.3 Структура навигационного сообщения

Навигационное сообщение имеет следующий состав:

- 63 бит: преамбула;
- 25 бит: текущее время на начало кадра;
- 1104 бит: сообщение в формате TLE;
- 8 бит: контрольная сумма.

Второе поле — текущее время на начало кадра по системной шкале времени в секундах. Количество бит достаточно для кодирования времени в пределах года. Момент начала формирования кадра привязан к началу соответствующей секунды, поэтому дробная часть времени отсутствует.

3 Описание модели

3.1 Форма представления модели

Модель написана на языке системы Matlab и может быть скачана с сайта http://srns.ru.

Модель разбита на множество отдельных функций, каждая из которых расположена в отдельном файле с расширением “*.p”. Это закодированные файлы Matlab, которые можно запустить на исполнение, но невозможно прочитать. В рамках курсового проекта нужно написать данные функции в формате “*.m” таким образом, чтобы они могли полностью подменять предоставленные в задании файлы “*.p”.

Главный файл, связывающий всю модель вместе, называется “main.m”. Для запуска модели в целом нужно запускать именно этот файл. Файл “main.m” предоставляется в виде исходного кода, его можно анализировать и изменять.

При запуске программы появляется окно, изображённое на рис. 2.

3.2 Соответствие между функциями и вариантами заданий

Список функций, которые нужно реализовать в каждом из вариантов, приведён в табл. 2.

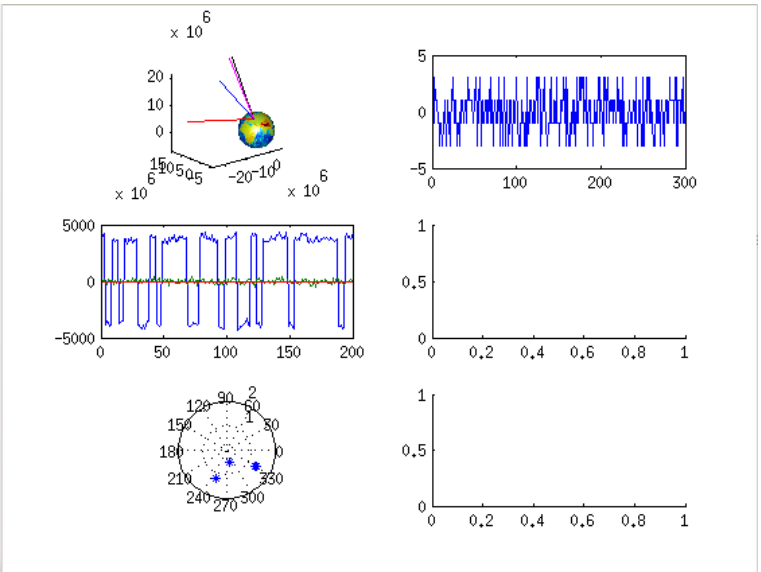

Рис. 2: Интерфейс программы

№	Рассматриваемый блок	Названия функций
1	Обработка данных TLE	readTLE(), decodeTLE()
2	Формирование сообщения	encodeMsg(), TLE2msg()
3	Расчёт дальности и формирование сигнала	makeSignal()
4	Радиочастотный блок	FrontEnd(), initFrontEnd()
5	Коррелятор	correlator()
6	Блок поиска	acq()
7	Некогерентный режим слежения	FllDll()
8	Когерентный режим слежения	PllDll()
9	Символьная синхронизация и демодуляция	SymSync()
10	Кадровая синхронизация	FrameSync()
11	Расчёт положения НС	getSatPos()
12	Блок решения навигационной задачи	NavSol()
13	Кодирование и декодирование CRC-8	CRC8()
14	Расчёт направления на НС	getElAz()

Таблица 2: Варианты заданий

4 Описание отдельных функций

4.1 Обработка файлов TLE

4.1.1 Чтение файла TLE — readTLE()

Функция readTLE() считывает строки из файла TLE. Интерфейс функции следующий:

```
  function [tle] = readTLE(filename)
  % function [str1, str2] = readTLE(filename)
  %
  %Функция для чтения файла TLE
  %Входные параметры
  % filename   - название файла TLE
  %Выходные параметры
  % tle        - список данных из файла TLE
```

На вход подаётся строка filename, содержащая имя файла для считывания. На выходе функции получается массив типа cell(), элементами которого являются структуры с двумя полями: str1 и str2, содержащие строки из файла TLE для каждой записи.

4.1.2 Декодирование файла TLE — decodeTLE()

Функция decodeTLE() берёт данные из файла TLE в том виде, в котором их выдаёт функция readTLE() и отправляет их в функцию twoline2rv() с сайта http://celestrak.com, которая декодирует данные.

```
  function [ephemeris] = decodeTLE(tle)
  % function [ephemeris] = decodeTLE(tle)
  %
  %Функция для чтения файла TLE
  %Входные параметры
  % tle        - список данных TLE
  %Выходные параметры
  % ephemeris - эфемериды
```

На выходе получается массив cell() структур ephemeris, содержащих орбитальные элементы.

4.2 Формирование и навигационного сообщения

4.2.1 Формирование сообщения — encodeMsg()

Данная функция объединяет преамбулу, текущее время и биты навигационных данных, рассчитывает контрольную сумму CRC8 и формирует 1200 символов сообщения.

```
  function [msg] = encodeMsg(t, tle)
  % function [msg] = encodeMsg(t, tle)
```

```
  %
  %Данная функция формирует навигационное сообщение
  %Входные параметры
  %t    - момент времени начала сообщения
  %tle  - передаваемые данные
  %Выходные параметры
  %msg  - кодированное сообщение (+- 1)
```

На входе t — текущее время, номер секунды на момент начала кадра, tle — строка из файла TLE в том виде, в котором её выдаёт функция readTLE().

На выходе — массив из 1200 чисел ±1, содержащий сообщение.

4.2.2 Преобразование строк TLE в биты данных — TLE2msg()

Данная функция преобразует две строки из файла TLE в массив символов ±1.

```
  function [msg] = TLE2msg(tle)
  %function [msg] = TLE2msg(tle)
  %
  %Данная функция формирует бинарное сообщение из записи TLE
  %Входные данные
  %tle - структура данных TLE две (строки по 69 байт)
  %Выходные данные
  %msg - биты данных (2*69*8 = 1104 бит)
```

На входе tle — запись из файла TLE в том виде, в котором её выдаёт функция readTLE(). Данная запись имеет вид структуры, содержащий два поля — str1 и str2. Длина каждой строки 69 байт. Функция преобразует эти байты в массив битов (старший бит идёт первым), а потом представляет их в виде чисел ±1.

4.3 Формирование сигнала

Данная функция формирует навигационных сигнал, осуществляя фазовую модуляцию несущей дальномерным кодом и навигационным сообщением.

```
  function [s, t, SgnParams] = makeSignal(t, Nd, SgnParams)
  %
  %Блок формирования сигнала
  %Входные параметры
  %t          - текущий момент времени
  %Nd         - количество отсчётов, которое требуется сформировать
  %SgnParams  - параметры формируемого сигнала
  %Выходные параметры
  %s          - сформированная выборка сигнала
  %t          - момент времени окончания сигнала
  %SgnParams  - параметры формируемого сигнала
```

На входе функции — начальный момент времени, число формируемых отсчётов и параметры сигнала. Параметры сигнала представлены в виде структуры SgnParams.

Функция должна рассчитать начальный момент времени для текущего кадра, определяет, не требуется ли обновить навигационное сообщение. Если его нужно обновить, сообщение обновляется с помощью функции encodeMsg(). Далее формируется несущая сигнала и моделируется сообщением и дальномерным кодом.

На выходе формируется выборка данных в виде массива s.

4.4 Радиочастотный блок — FrontEnd()

Модель радиочастотного блока содержит модель собственных шумов приёмника, фильтр и модель 2-разрядного АЦП.

```
  function [y, FrontEndParams] = FrontEnd(s, FrontEndParams)
  % function [out, FrontEndParams] = FrontEnd(s, FrontEndParams)
  %
  % Модель радиочастотного блока
  % Входные параметры
  % s              - выборка входного сигнала
  % FrontEndParams - параметры фронтэнда-
  % Выходные параметры
  % y              - выходной сигнал
  % FrontEndParams - параметры фронтэнда-
```

На вход поступает смесь входных сигналов s. Параметры фронт-энда содержатся в структуре FrontEndParams. Для инициализации данной структуры служит специальная функция initFrontEnd().

```
  function [FrontEndParams] = initFrontEnd(FrontEndParams)
  % function [FrontEndParams] = initFrontEnd(FrontEndParams)
  %
  % Данная функция инициализирует фронтэнд-
  % Входные параметры:
  % FrontEndParams     - структура с данными фронтэнда-
  % FrontEndParams.fd  - частота дискретизации
  % FrontEndParams.fi  - промежуточная частота
  % FrontEndParams.BW  - полоса фильтра
  % Выходные параметры
  % FrontEndParams     - структура с данными фронтэнда-
```

На выходе формируется выборка наблюдений в виде массива y.

4.5 Модель коррелятора — correlator()

Модель коррелятора умножает входной сигнал на опорные сигналы и накапливает корреляционные интегралы синхронно с границами дальномерного кода.

```
  function [CorrRegs, CorrParams] = correlator(y, CorrRegs,
      CorrParams)
  % function [CorrRegs, CorrParams] = correlator(t, y, CorrRegs,
      CorrParams)
  %
```

```
  % Модель коррелятора
  % Входные параметры
  % y ---------- - выборка входного сигнала
  % CorrRegs --- - внутренее состояние коррелятора
  % CorrParams - - внутреннее состояние коррелятора
  % Выходные параметры
  % y ---------- - оставшаяся часть сигнала
  % CorrRegs --- - внутреннее состояние коррелятора
  % CorrParams - - внутреннее состояние коррелятора
```

На вход поступают наблюдения y. Параметры коррелятора (частота дискретизации, номинальная частота сигнала и т.д.) хранятся в структуре CorrParams. В структуре CorrRegs хранятся данные, эквивалентные регистрам реального коррелятора. Часть из них используется для управления коррелятором (freq, dtau), часть содержит накопленные корреляционные интегралы (I, Q).

4.6 Блок поиска — acq()

Блок поиска перестраивает задержку опорного сигнала до тех пор, пока не обнаружит сигнал на входе.

```
  function [detected, CorrRegs, AcqParams] = acq(CorrRegs, ...
      AcqParams)
  % function [detected, CorrRegs, AcqParams] = acq(CorrRegs, ...
      AcqParams)
  %
  % Алгоритм последовательного поиска сигнала
  % Входные параметры
  % CorrRegs -- внутренее состояние коррелятора
  % AcqParams - параметры системы поиска
  % Выходные параметры
  % detected -- признак обнаружения сигнала (1 - обнаружен, 0 - нет)
  % CorrRegs -- внутренее состояние коррелятора
  % AcqParams - параметры системы поиска
```

На вход блока поиска поступают данные из коррелятора в виде структуры CorrRegs и параметры блока поиска в виде структуры AcqParams.

Блок поиска на каждом шаге изменяет задержку опорного дальномерного кода, используюя CorrRegs.dtau и сравнивает огибающую сигнала на выходе коррелятора с порогом. При превышении порога сигнал считается обнаруженным.

На выходе формируется переменная detected, свидетельствующая об успехе процесса поиска.

4.7 Некогерентные режим слежения — FllDll()

В некогерентном режиме осуществляется слежение за частотой из задержкой сигнала.

```
  function [CorrRegs, FllParams] = FllDll(CorrRegs, FllParams)
```

```
% function [CorrRegs, FllParams] = FllDll(CorrRegs, FllParams)
%
% Система некогерентного слежения за частотой и задержкой
% Входные параметры
%   CorrRegs   - внутренее состояние коррелятора
%   FllParams - параметры системы поиска
% Выходные параметры
%   detected   - признак обнаружения сигнала (1 - обнаружен, 0 - нет)
%   CorrRegs   - внутренее состояние коррелятора
%   FllParams - параметры системы поискаДанная

 функция содержит в себе ЧАП 2 порядка и ССЗ 1 порядка. На вход
    поступает структура
CorrRegs с отсчётами на выходе коррелятора. В структуре FllParams
    хранятся внутренниеданные
 функции ---- параметры дальномерного кода, параметры и векторы
    состояния фильтров. В

 процессе работы функция управляет коррелятором, изменяя поля
    структуры CorrRegs:
CorrRegs.freq ---- частота опорного сигнала, CorrRegs.dtau ---- сдвиг
    задержки дальномерного кода.
```

4.8 Когерентный режим слежения — PllDll()

В когерентном режиме слежения осуществляется слежение за фазой и задержкой сигнала.

```
  function [CorrRegs, PllParams] = PllDll(CorrRegs, PllParams)
% function [CorrRegs, PllParams] = PllDll(CorrRegs, PllParams)
%
% Система некогерентного слежения за частотой и задержкой
% Входные параметры
%   CorrRegs   - внутренее состояние коррелятора
%   FllParams - параметры системы поиска
% Выходные параметры
%   detected   - признак обнаружения сигнала (1 - обнаружен, 0 - нет)
%   CorrRegs   - внутренее состояние коррелятора
%   FllParams - параметры системы поиска
```

Данная функция содержит в себе ФАП 3 порядка и ССЗ 1 порядка. На вход поступает структура CorrRegs с отсчётами на выходе коррелятора. В структуре PllParams хранятся внутренние данные функции — параметры дальномерного кода, параметры и векторы состояния фильтров.

Структура ССЗ аналогична той, что используется в некогерентном режиме слежения.

В процессе работы функция управляет коррелятором, изменяя поля структуры CorrRegs: CorrRegs.freq — частота опорного сигнала, CorrRegs.dtau — сдвиг задержки дальномерного кода.

4.9 Система символьной синхронизации — SymSync()

Данная функция обнаруживает границы символов, поддерживает символьную синхронизацию после обнаружения границ символов и накапливает данные на длительности символов.

```
function [syncDone, II, eph, SyncParams] = SymSync(CorrRegs, SyncParams)
% function [syncDone, SyncParams] = SymSync(CorrRegs, SyncParams)
%
% Блок символьной синхронизации и демодуляции сообщения
% Входные параметры
%   CorrRegs    - регистры коррелятора
%   SyncParams  - параметры блока демодуляции
% Выходные параметры
%   syncDone    - признак прихода нового кадра
%   II          - накопленный символ данных
%   eph         - номер эпохи, соответствующий началу символа
%   SyncParams  - параметры блока демодуляции
```

На вход поступают данные из коррелятора в виде структуры CorrRegs. Среди прочего, эта структура содержит поле CorrRegs.I, представляющее собой корреляционный интеграл, накопленный на длительности одной эпохи (1 мс). Длительность символа данных равна 5 мс, поэтому для демодуляции символа данных нужно накопить 5 отсчётов с выхода коррелятора.

Но перед этим нужно определить границы символов — один из 5 возможных вариантов сдвига с шагом 1 мс. Для выполнения данной задачи функция SymSync накапливает сигналы со всеми возможными 5 вариантами сдвига и накапливает гистограмму. После сбора некоторой статистики определяется максимум гистограммы, соответствующий сдвигу данных.

На выходе функции формируется признак синхронизации syncDone. Кроме того, формируется накопленный на 5 мс символ II. Этот символ формируется раз в 5 эпох, в остальное время II — пустой массив.

4.10 Кадровая синхронизация — FrameSync()

Для декодирования сообщения необходимо установить кадровую синхронизацию.

```
function [new_frame, frame, FrameParams, R] = FrameSync(I, epoch, FrameParams)
% function [syncDone, frame, FrameParams] = FrameSync(I, epoch, FrameParams)
%
```

```
% Блок кадровой синхронизации
% Входные параметры
%   I           - накопленный символ
%   eph         - номер эпохи, соответствующий началу символа
%   FrameParams - параметры блока демодуляции
% Выходные параметры
%   new_frame   - признак прихода нового кадра
%   frame       - данные, содержащиеся в кадре
%   FrameParams - параметры блока демодуляции
```

Данная функция получает накопленные символы данных по одному на входе I. Далее символы накапливаются во внутреннем массиве, являющемся полем структуры FrameParams.II. После сбора достаточного числа символов (1200) входной накопленные данные умножаются на опорный сигнал — преамбулу и рассчитывается модуль корреляционного интеграла. На каждом вызове функции FrameSync происходит сдвиг данных на 1 символ. При превышении порога принимается решение о нахождении границы символа. После этого данные перемещаются в массив frame и выдаются в качестве результата работы функции, после чего продолжается накопление следующего кадра.

4.11 Расчёт координат спутников — getSatPos()

Данная функция рассчитывает положение спутника на требуемый момент времени.

```
function [X, V] = getSatPos(t, ephemeris)
% function [X, V] = getSatPos(t, ephemeris)
%
% Данная функция рассчитывает положение спутника с использованием
    модели SGP4
% Входные параметры
%   t           - массив моментов времени от начала эпохи эфемерид
%   ephemeris - эфемериды данного спутника
% Выходные параметры
%   X           - координаты: строки - время, столбцы - координаты (x,
    y, z), м
%   V           - скорость:   строки - время, столбцы - компоненты (Vx,
    Vy, Vz), мс/
```

На вход поступают эфемериды спутника в том виде, в котором их выдаёт функция decodeTLE(). Кроме того, параметром является текущий момент времени с начала эпохи эфемерид t.

Функция основывается на вызове функции sgp4() с сайта http://celestrak.com, реализующей официальный алгоритм расчёта положения НС по модели SGP4.

Выходные данные представляются в виде вектора координат X и вектора скоростей V.

4.12 Блок решения навигационной задачи — NavSol()

Данная функция определяет положение потребителя.

```
function [UserPos, DOP, dPR, dX, Niter] = NavSol(InitPos, SatPos, PR, varargin)
% function [UserPos, DOP, dPR, dX, Niter] = NavSol(InitPos, SatPos, PR, [MaxErr], [MaxIter])
%
% Данная функция решает навигационную задачу
% Входные параметры:
%   - InitPos    - начальное приближение положения потребителя [x; y; z; tau]
%   - SatPos     - положение НС столбцы( - спутники, строки - [x; y; z; tau]
%   - PR         - псевдодальности 1*N
%   - [MaxErr]   - погрешность
%   - [MaxIter]  - максимальное число итераций
% Выходные параметры
%   - UserPos  - вычисленное положение
%   - DOP      - геометрический фактор
%   - dPR      - невязки по спутникам
%   - dX       - остаточная ошибка в псевдодальностях
%   - Niter    - число итераций
```

На вход поступает начальное приближение координат (это может быть нулевой вектор), массив координат спутников, участвующих в решении и массив псевдодальностей.

На выходе рассчитывается вектор координат UserPos, геометрический фактор и вспомогательные данные.

4.13 Расчёт контрольной суммы — CRC8()

Данная функция рассчитывает контрольную сумму CRC8 и используется как при формировании сигнала, так и при декодировании сообщения.

```
function [CRC] = CRC8(msg)
% function [CRC] = CRC8(msg)
%
% Расчёт контрольной суммы CRC8
% Входные данные
%   msg - сообщение в формате +- 1
% Выходные данные
%   CRC - контрольная сумма
```

На входе — сообщение в виде массива чисел ± 1. На выходе — результат вычисления контрольной суммы, массив из 8 чисел ± 1.

Алгоритм расчёта может быть найден в Википедии.

4.14 Расчёт направлений на НС — getElAz()

Данная функция рассчитывает угол места и азимут спутника по декартовым координатам спутника и геодезическим координатам потребителя.

```
function [El, Az] = getElAz(Sat, lat, lon, h)
% function [El, Az] = getElAz(Sat, lat, lon, h)
%
% Данная функция рассчитывает направление на спутник
% Входные параметры:
%   Sat - координаты спутника в декартовой системе координат WGS84
%   lat - широта наблюдателя
%   lon - долгота наблюдателя
%   h   - высота наблюдателя
% Выходные параметры:
%   El  - угол места
%   Az  - азимут
```

Образец алгоритма расчёта может быть найден по адресу http://www.mathworks.com/matlabcentral/fileexchange/13439-orbital-mechanics-library/content/topo.m Его надо переоформить, чтобы программный интерфейс соответствовал представленному выше.