Санкт-Петербургский государственный университет

Математико-механический факультет
Специальность Астрономия

Стариков Кирилл Игоревич

Вычисление целочисленной неоднозначности в процессе точного местоопределения

Курсовая работа

Научный руководитель:
и.о. зав. кафедрой астрономии, доцент Петров С. Д.

Санкт-Петербург
2021

SAINT-PETERSBURG STATE UNIVERSITY

The Faculty of Mathematics and Mechanics
Astronomy department

Kirill Starikov

Computing integer ambiguity in process of precise point positioning

Course Work

Scientific supervisor:
docent Sergei Petrov

Saint-Petersburg
2021

Оглавление

Введение

Тема работы

В данной работе был реализован алгоритм разрешения целочисленной неоднозначности псевдофазовых измерений, являющихся частью процесса высокоточных абсолютных местоопределений в ГНСС.

Подход

Для разрешения целочисленной неоднозначности псевдофазовых измерений был выбран метод MLAMBDA [3], являющийся модификацией давно использующегося метода LAMBDA [2]. Алгоритм, излагающийся в [3], был реализован в моей работе на языке программирования C++. Для работы с матрицами был создан соответсвующий класс с базовыми методами(арифметические операции, транспонирование, инверсия). За опорную точку были взяты функции, использующиеся в библиотеке RTKLib [1].

1. Обзор метода

Ререшение целочисленной неоднозначности псефдофазовых измерений является важным этапом высокоточных местоопределений в ГНСС. Согласно схеме на рис. 1 после фильтрационной процедуры оценивания становятся известны вектор действительный оценок (float solution)

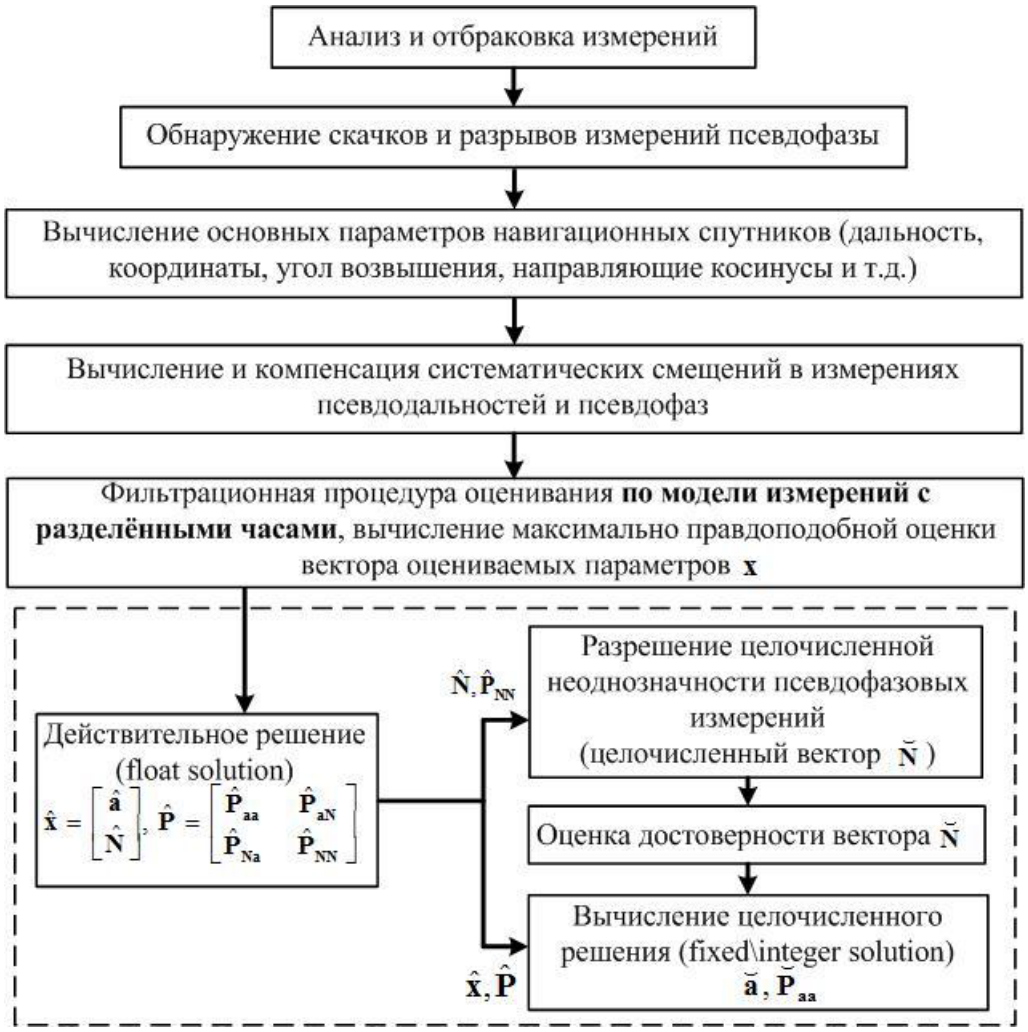

Рис. 1: Укрупненная блок-схема алгоритма высокоточного местоопределения потребителя с разрешением целочисленной неоднозначности псевдофазовых измерений (Integer PPP) [4]

$$\hat{x} = \begin{bmatrix} \hat{a} \\ \hat{N} \end{bmatrix} \tag{1}$$

и соответсвующяя ему ковариационная матрица

$$\hat{P} = \begin{bmatrix} \hat{P}_{aa} & \hat{P}_{aN} \\ \hat{P}_{Na} & \hat{P}_{NN} \end{bmatrix} \tag{2}$$

где $\hat{a}$ - вектор действительных оценок всех параметров кроме неоднозначностей, $\hat{N}$ - вектор действительных оценок целочисленных неоднозначностей $\hat{N}^j$, выраженных в циклах. Блок $\hat{P}_{NN}$ матрицы $\hat{P}$ (2) является ковариационной матрицей вектора $\hat{N} = \begin{bmatrix} \hat{N}_3^G & \hat{N}_4^G \end{bmatrix}^{-T}$, содержащего действительные значения оценок для целочисленных векторов $\hat{N}_i^G$ и $\hat{N}_j^G$, где i и j зависят от ионосферной модели разделённых часов.

В соответствии с [2] осуществляется минимизация в целых числах N следующей квадратичной формы:

$$D(N) = (N - \hat{N})^T (\hat{P}_{NN})^{-1} (N - \hat{N}) = d \longrightarrow min \tag{3}$$

В предположении, что целочисленный вектор N может принимать произвольные действительные значения квадратичная форма (3) в пространстве с координатами N задаèт уравнение эллипсоида с центром в точке $\hat{N}$ (вектор $\hat{N}$ содержит действительные оценки искомых целочисленных неоднозначностей, доступных на выходе фильтрационной процедуры оценивания, рис. 1). Как правило, ищется не единственный целочисленный вектор $\tilde{N}$, минимизирующий квадратичную форму (3), а некоторое число k целочисленных векторов, $\tilde{N}$, $i = 1, ..., k$, которые задают последовательно нарастающие значения квадратичной формы (3).

Известно, что на практике матрица $\hat{P}_{NN}$ является плохо обусловленной (отношение еè максимального и минимального собственных чисел может доходить до нескольких десятков тысяч), что порождает сильную вытянутость (или сплюснутость) эллипсоида (3). С целью повышения эффективности поисковой процедуры минимизации квадратичной формы (3) применяется линейное целочисленное унимодулярное преобразование - LAMBDA-method [3] - идея которого состоит в отображении эллипсоида (3) в другое целочисленное пространство M такое, что эллипсоид (3) в нèм преобразуется в фигуру, близкую к шару, которая определяется преобразованной квадратичной формой $D^*(M)$. Близкая к шару форма преобразованного эллипсоида резко сокращает расходы машинного времени на поиск целочисленного минимума преобразованной квадратичной формы (3). Затем над k найденными целочисленными векторами, $\tilde{M}_i$, $i = 1, ..., k$, доставляющими последовательно нарастающие целочисленные минимумы

преобразованному эллипсоиду $D^*(M)$, осуществляется обратное ЦУМП к целочисленному пространству N. Результатом разрешения целочисленной неоднозначности являются k целочисленных векторов, $\tilde{N}_i$, $i = 1, ..., k$, доставляющих последовательно нарастающие целочисленные минимумы d, $\tilde{d}_i$, $i = 1, ..., k$ квадратичной формы (3).

2. LAMBDA-method и MLAMBDA (??)

Пусть $\tilde{a} \in \mathbb{R}^n$ - веществennозначное приближение наименьших квадратов (НК) целочисленного вектора парметров $a \in \mathbb{Z}^n$ (в нашем случае целочисленный вектор неоднозначностей) и $Q_a \in \mathbb{R}^{n \times n}$ - симметричная и положительноопределённая ковариационная матрица. Приближение целочисленного МНК (ILS) $\tilde{a}$ является решением задачи минимизации:

$$\min_{a \in \mathbb{Z}^n} (a - \tilde{a})^T Q_{\tilde{a}}^{-1} (a - \tilde{a}) \tag{4}$$

Данный метод состоит из двух этапов - редукция и поиск. Опишем каждый из них.

2.1. Процесс редукции

На практике известно, что матрица $Q_{\tilde{a}}^{-1}$ является плохо обусловленной (отношение еѐ максимального и минимального собственных чисел может доходить до нескольких десятков тысяч), что порождает сильную вытянутость (или сплюснутость) эллипсоида (4). С целью повышения эффективности поисковой процедуры минимизации квадратичной формы (4) применяется линейное целочисленное унимодулярное преобразование.

Пусть $Z \in \mathbb{Z}^{n \times n}$ - унимодулярная матрица, то есть $|det(Z)| = 1$. Очевидно, что Z^{-1} тоже целочисленная. Используются преобразования:

$$z = Z^T a, \quad \hat{z} = Z^T \hat{a}, \quad Q_{\hat{z}} = Z^T Q_{\hat{z}} Z. \tag{5}$$

Они приводят 4 к задаче следующего вида:

$$\min_{z \in \mathbb{Z}^n} (z - \hat{z})^T Q_{\hat{z}}^{-1} (z - \hat{z}) \tag{6}$$

Этот переход преобразовывает эллипсоид (4) в фигуру, близкую к шару, которая определяется квадратичной формой (6). Близкая к шару форма преобразованного эллипсоида резко сокращает расходы машинного времени на поиск целочисленного минимума исходной задачи.

Пусть $L^T DL$ факторизация $Q_{\hat{a}}$ и $Q_{\hat{z}}$ соответственно имеют емеет следующий вид:

$$Q_{\hat{a}} = L^T DL; \quad Q_{\hat{z}} = Z^T L^T DLZ = \tilde{L}^T \tilde{D} \tilde{L} \tag{7}$$

Где L и $\tilde{L}$ - нижние унитреугольные матрицы, и $D = diag(d_1, ..., d_n)$, $\tilde{D} = diag(\tilde{d}_1, ..., \tilde{d}_n)$, $d_i, \tilde{d}_i > 0$. Процесс редукции начинается с $L^T DL$ факторизации $Q_{\hat{a}}$ и обновляет элементы таким образом, чтобы получить $L^T DL$ факторизацию $Q_{\hat{z}}$. В этом процессе пытаются найти унимодулярную матрицу Z для достижения двух целей, которые имеют решающее значение для эффективности процесс поиска: (i) $Q_{\hat{z}}$ диагональна настолько насколько возможно (т. е. недиагональные элементы $\tilde{L}$ имеют минимльные

значения); (ii) Диагональные элементы $\tilde{D}$ распределены в порядке убывания, если это возможно, т. е. процесс стремится к выполнению следующего условия:

$$\tilde{d}_1 \geq \tilde{d}_2 \geq ... \geq \tilde{d}_n \tag{8}$$

Обратите внимание, что первая задача - декорреляция неизвестных параметров, и это часть процесса редукции.

Здесь сделаем замечание. Алгоритм LLL также преследует две вышеупомянутые цели. Фактически, Алгоритм редукции LAMBDA основан на идеях Ленстры (Lenstra, 1981), которые были модифицированы и приведены к LLL-алгоритму (Hassibi and Boyed, 1998). Подходы, представленные Лиу (Liu et al., 1999) и Щу (Xu, 2001) в основном преследуют первую цель, а вторая цель достигается лишь частично. В методе LAMBDA унимодулярная матрица Z в (5) строится последовательностью целочисленных преобразований и перестановок Гаусса. Преобразования используются для того, чтобы сделать недиагональные элементы $\tilde{L}$ как можно меньше, в то время как перестановки устремлют процесс к выполнию условия (8).

2.1.1. Целочисленные преобразования Гаусса

Целочисленное преобразование Гаусса Z_{ij} имеет следующую форму:

$$Z_{ij} = I - \mu e_i e_j^T$$

, где μ - целое число. Легко показать, что $Z_{ij}^{-1} = I + \mu e_i e_j^T$. Применяя Z_{ij} к L справа даёт

$$\tilde{L} = LZ_{ij} = L - \mu L e_i e_j^T$$

Таким образом $\tilde{L}$ такая же, как L за исключением

$$\tilde{l}_{kj} = l_{kj} - \mu l_{ki}, \quad k = i, ..., n$$

Чтобы сделать $\tilde{l}_{kj}$ как можно меньше, берут $\mu = \lfloor l_{ij} \rceil$, обеспечивая

$$|\tilde{l}_{ij}| \leq 1/2, \quad i > j \tag{9}$$

Когда Z_{ij} применяется к L справа, одновременно следует применять Z_{ij}^T к $\hat{a}$ слева. Все остальные преобразования тоже должны быть посчитаны.

2.1.2. Перестановки

Чтобы добиться порядка (8), симметричные перестановки ковариационной матрицы $Q_{\hat{a}}$ требуют процесса редукции. Когда два диагональных элемента $Q_{\hat{a}}$ меняются местами, факторы L и D факторизации $L^T D L$ должны быть обновлены. Предполо-

жим, что мы разделяем L^TDL факторизацию $Q_{\hat{a}}$ следующим образом:

$$Q_{\hat{a}} = L^TDL = \begin{bmatrix} L_{11}^T & L_{21}^T & L_{31}^T \\ & L_{22}^T & L_{32}^T \\ & & L_{33}^T \end{bmatrix} \begin{bmatrix} D_1 & & \\ & D_2 & \\ & & D_3 \end{bmatrix} \begin{bmatrix} L_{11} & & \\ L_{21} & L_{22} & \\ L_{31} & L_{32} & L_{33} \end{bmatrix}$$

Пусть

$$P = \begin{bmatrix} 0 & 1 \\ 1 & 0 \end{bmatrix}, \quad P_{k,k+1} = \begin{bmatrix} I_{k-1} & & \\ & P & \\ & & In-k-1 \end{bmatrix}$$

Можно показать, что $P_{k,k+1}^T Q_{\hat{a}} P_{k,k+1}$ имеет L^TDL факторизацию

$$P_{k,k+1}^T Q_{\hat{a}} P_{k,k+1} = \begin{bmatrix} L_{11}^T & \tilde{L}_{21}^T & L_{31}^T \\ & \tilde{L}_{22}^T & \tilde{L}_{32}^T \\ & & L_{33}^T \end{bmatrix} \begin{bmatrix} D_1 & & \\ & \tilde{D}_2 & \\ & & D_3 \end{bmatrix} \begin{bmatrix} L_{11} & & \\ \tilde{L}_{21} & \tilde{L}_{22} & \\ L_{31} & \tilde{L}_{32} & L_{33} \end{bmatrix} \tag{10}$$

где

$$\tilde{D}_2 = \begin{bmatrix} \tilde{d}_k & \\ & \tilde{d}_{k+1} \end{bmatrix}, \quad \tilde{d}_{k+1} = d_k + l_{k+1,k}^2 d_{k+1}, \quad \tilde{d}_k = \frac{d_k}{\tilde{d}_{k+1}} d_{k+1}, \tag{11}$$

$$\tilde{L}_{22} = \begin{bmatrix} 1 & \\ \tilde{l}_{k+1,k} & 1 \end{bmatrix}, \quad \tilde{l}_{k+1,k} = \frac{d_{k+1} l_{k+1,k}}{\tilde{d}_{k+1}}, \tag{12}$$

$$\tilde{L}_{21} = \begin{bmatrix} -l_{k+1,k} & 1 \\ \frac{d_k}{\tilde{d}_{k+1}} & \tilde{l}_{k+1,k} \end{bmatrix} L_{21} = \begin{bmatrix} -l_{k+1,k} & 1 \\ \frac{d_k}{\tilde{d}_{k+1}} & \tilde{l}_{k+1,k} \end{bmatrix} L(k:k+1, 1:k-1), \tag{13}$$

$$\tilde{L}_{21} = L_{32}P = [L(k+2:n, k+1)\ L(k+2:n, 1:k)]. \tag{14}$$

Если мы имеем

$$\tilde{d}_{k+1} < d_{k+1} \tag{15}$$

(из этого следует, что $d_k < d_{k+1}$), то перестановки выполнены. Это не гарантирует, что $\tilde{d}_k \geq \tilde{d}_{k+1}$, но это делает разрыв между $\tilde{d}_k$ и $\tilde{d}_{k+1}$ меньше, чем между d_k и d_{k+1}. Стоит отметить, что между абсолютные значения элементов ниже $l_{k,k}$ и $l_{k+1,k+1}$ в L ограничены выше на $1/2$, оценки остаются в силе после перестановки, за исключением того, что $\tilde{l}_{k+1,k}$ (ур. 12) может больше не ограничиваться $1/2$.

Здесь мы хотели бы указать на важное свойство редукции LAMBDA. После завершения процесса редукции неравенство (15) не будет выполняться ни при каких k (иначе будет выполнена новая перестановка), таким образом, для $\tilde{L}$ и $\tilde{D}$, полученных в конце редукции процесса, мы должны иметь (ур. 11)

$$\tilde{d}_k + \tilde{l}_{k+1,k}^2 \tilde{d}_{k+1} \geq \tilde{d}_{k+1}, \quad k = 1, 2, ..., n-1$$

или

$$\tilde{d}_k \geq (1 - \tilde{l}^2_{k+1,k})\tilde{d}_{k+1}, \quad k = 1, 2, ..., n-1 \tag{16}$$

Это тот порядок, который может гарантировать LAMBDA редукция, хотя она и стремится к более сильному порядку (см. ур-ние 8). Легко показать, что уравнения (2.1.2) и (16) эквивалентны так называемым критериям редукции LLL, которым удовлетворяют $\tilde{L}$ и $\tilde{D}$, полученные с помощью редукции LLL-алгоритма, см. Ленстра (Lenstra et al., 1982) и Эйгерлл (Agrell et al., 2002).

2.2. Модификации процесса редукции

Рассмотрим теперь какие модификаци процесса редукции были предложенны в [3].

2.2.1. Стратегия симметричной опоры

Чтобы добиться неравенства в уравнении (8) или для достижения неравенства в уравнении (16), алгоритм редукции выполняет перестановки. В общем случае, операции перестановок, вероятно, будут самыми дорогостоящими для всего процесса редукции. Чем меньше количество перестановок, тем быстрее работает алгоритм. Мотивируясь этим, в [3] было предложено включить стратегию симметричной опоры (symmetric pivoting strategy) при вычислении L^TDL факторизации ковариационной матрицы $Q_{\hat{a}}$ в начале процесса редукции. Сначала мы рассмотрим вывод алгоритма факторизации L^TDL без опорного элемента. Предположим, что $Q \in \mathbb{R}^{n\times n}$ - симметричная положительноопределённая матрица. Разбиваем $Q = L^TDL$ следующим образом:

$$\begin{bmatrix} \tilde{Q} & q \\ q^T & q_{nn} \end{bmatrix} = \begin{bmatrix} \tilde{L}^T & l \\ & 1 \end{bmatrix} \begin{bmatrix} \tilde{D} & \\ & d_n \end{bmatrix} \begin{bmatrix} \tilde{L} & \\ l^T & 1 \end{bmatrix}$$

Поэтому

$$d_n = q_{nn}, \quad l = q/d_n, \quad \tilde{Q} - ld_nl^T = \tilde{L}^T\tilde{D}\tilde{L}$$

Эти уравнения ясно показывают, как найти d_n, l, $\tilde{L}$ и $\tilde{D}$.
Поскольку мы стремимся к неравенствам в уравнении (8) сначала симметрично переставляем наименьший диагональный элемент матрицы Q на позицию (n, n), а затем найти d_n, l и применить тот же подход к $\tilde{Q}$–ld_nl^T. Наконец, мы получаем L^TDL-факторизацию Q с перестановками. Предположим, что после первой симметричной перестановки P_1 имеем:

$$P_1^TQP_1 = \begin{bmatrix} \tilde{Q} & q \\ q^T & q_{nn} \end{bmatrix}$$

Определим $d_n = q_{nn}$ и $l = q/d_n$. Пусть $\tilde{Q} - l d_n l^T$ имеет следующую $L^T DL$-факторизацию с симметричной опорой

$$\tilde{P}^T(\tilde{Q} - l d_n l^T)\tilde{P} = \tilde{L}^T \tilde{D} \tilde{L}$$

где $\tilde{P}$ - результат перестановки матриц. Тогда легко убедиться, что

$$P^T Q P = L^T D L, \quad P = P_1 \begin{bmatrix} \tilde{P} & \\ & 1 \end{bmatrix}, \quad L = \begin{bmatrix} \tilde{L} & \\ l^T \tilde{P} & 1 \end{bmatrix}, \quad D = \begin{bmatrix} \tilde{D} & \\ & d_n \end{bmatrix},$$

давая $L^T DL$-факторизацию Q с симметричной опорой. Обратите внимание, что $L^T DL$ - факторизация с симметричной опорой аналогична факторизации Холецкого симметричной неотрицательноопределенной матрицы с симметричной опорой. (см., например, Голуб и Ван Лоан (1996), разд. 4.2.9) Но в последнем случае опорный элемент выбирается как самый большой элемент, а не как самый маленький. Следует отметить, что эта стратегия симметричной опоры также использовалась у Щу (Xu et al., 1995) и (Лю и др., 1999) с разными мотивировками.

2.2.2. Стратегия жадного выбора

После $L^T DL$-факторизации с симметричной опорой мы запускаем процесс редукции. Чтобы сделать редукцию более эффективной, мы хотели бы еще больше сократить количество перестановок. Как уже изложено выше процесс редукции в методе LAMBDA выполняется последовательно - справа-налево. Когда условие $\tilde{d}_{k+1} < d_{k+1}$ (см. уравнение 15) выполнено, происходит перестановка пары $(k, k+1)$, а затем мы возвращаемся в исходное состояние, т.е. $k = n - 1$. Интуитивно маловероятно, что такое сокращение будет очень эффективным. Когда мы достигаем критического индекса k, т.е. $d_{k+1} \ll d_k$ и $\tilde{d}_{k+1} < d_{k+1}$, выполняется перестановка на этой позиции, но вполне вероятно, что мы получим $d_{k+2} \ll d_{k+1}$ и $\tilde{d}_{k+2} < d_{k+2}$, поэтому $k+1$ становится критическим индексом и так далее. Следовательно, перестановки, выполненные до того, как мы достигли индекса k, вероятно, будут потрачены впустую.

Одно из решений этой проблемы - применить то, что мы называем стратегией жадного выбора. Вместо цикла по k от $n-1$ до 1, мы всегда выбираем индекс k так, чтобы d_{k+1} уменьшается больше всего при перестановке пары $(k, k+1)$, т.е. k определяется как

$$k = arg \min_{1 \le j \le n-1} \left\{ \tilde{d}_{j+1}/d_{j+1} : \tilde{d}_{j+1} < d_{j+1} \right\} \tag{17}$$

Если таких k не найдено, то перестановки не выполняются.

2.2.3. Стратегия ленивых преобразований

Перестановки в процессе редукции метода LAMBDA могут изменять величины недиагональных элементов L. Может случиться, что целочисленные преобразова-

ния Гаусса применяются к одним и тем же элементам L много раз из-за перестановок. В частности, если выполняется перестановка для пары $(k, k+1)$ изменяется $L(k:k+1, 1:k-1)$ (см. уравнение 13) и два столбца $L(k+1:n, k:k+1)$ (см. уравнение 14) меняются местами. Если абсолютные значения элементов $L(k:k+1, 1:k-1)$ ограничены сверху $1/2$ перед перестановкой, то после перестановки эти границы больше не гарантируются. Таким образом, для элементов $L(k:k+1, 1:k-1)$, которые теперь больше $1/2$ по величине, теряется соответствующее целочисленное преобразование Гаусса, которые было использовано, чтобы ограничить эти элементы сверху на $1/2$ перед этой перестановкой. Перестановка также влияет на $l_{k+1,k}$ (см. уравнение 12), абсолютное значение которого может не быть ограничено сверху $1/2$. Но любое целочисленное преобразование Гаусса, которое применялось для обеспечения $|l_{k+1,k}| \leq 1/2$ перед перестановкой, не тратится зря, так как это преобразование необходимо сделать для выполнения этой перестановки. Причина в следующем. Заметим, что $\tilde{d}_{k+1} = d_k + l_{k+1,k}^2 d_{k+1}$ (см. уравнение 11). Цель перестановки - уменьшить $\tilde{d}_{k+1}$, что $l_{k+1,k}$ была как можно меньше, что и реализуется целочисленным преобразованием Гаусса. Мы предлагаем сначала применять целочисленные преобразования Гаусса только к некоторым субдиагональным элементам L, а затем делать перестановки. В процессе перестановок целочисленные преобразования Гаусса будут применяться только к некоторым измененным субдиагональным элементам L. Если перестановок выполнено не будет, то применим преобразования к недиагональным элементам L. Мы называем это стратегия стратегия «ленивого» преобразования. В частности, в начале процесса редукции, целочисленное преобразование Гаусса применяется к $l_{k+1,k}$, когда выполняется следующий критерий:

$$\text{Критерий 1: } \; d_k < d_{k+1} \tag{18}$$

Когда этот критерий не выполнился, это значит, что d_k и d_{k+1} были в правильном порядке, поэтому нам не нужно делать перестановку для пары $(k, k+1)$. Позже целочисленное преобразование Гаусса будет применяется к $l_{k+1,k}$, когда удовлетворяются как критерий 1, так и следующий критерий 2:

$$\text{Критерий 2: } \; l_{k+1,k}. \tag{19}$$

Этот критерий используется для того, чтобы пропустить неизменённые субдиагональные элементы L. После перестановки для пары $(k, k+1)$, три элемента на поддиагонали L меняются. Это: $l_{k,k-1}, l_{k+1,k} l_{k+2,k+1}$. Таким образом, после перестановки, к этим эементам применяется не более трех целочисленных преобразований Гаусса. Наконец, когда перестановок выполнено не будет, целочисленное преобразование Гаусса применяется ко всем элементам в строго нижнетреугольной части L.

2.3. Процесс дискретного поиска

После процесса редукции запускается процесс дискретного поиска. Для решения задачи ЦМНК (6) стратегия дискретного поиска используется для перечисления подпространства в $\mathbb{Z}^n$, которое содержит решение. Предположим, имеется следующая оценка целевой функции в уравнении. (6):

$$f(z) \stackrel{def}{=} (z-\hat{z})^T Q_{\hat{z}}^{-1}(z-\hat{z}) \leq \chi^2 \tag{20}$$

Решение ищется над эти гиперэллипсоидом.
Подставляя $L^T DL$ факторизацию $Q_{\hat{z}}$ из (7) в (20) получим

$$f(z) = (z-\hat{z})^T \tilde{L}^{-1}\tilde{D}^{-1}\tilde{L}^{-T}(z-\hat{z}) \leq \chi^2 \tag{21}$$

Определяя

$$\tilde{z} = z - \tilde{L}^{-T}(z-\hat{z}), \tag{22}$$

получим

$$\tilde{L}^{-T}(z-\tilde{z}) = z-\hat{z},$$

то есть

$$\tilde{z}_n = \hat{z}_n, \quad \tilde{z}_i = \hat{z}_i + \sum_{j=i+1}^{n}(z_j - \tilde{z}_j)\tilde{l}_{ij}, \quad i = n-1, n-2, ..., 1 \tag{23}$$

Можно заметить, что $\tilde{z}_i$ зависит от $z_{i+1}, ..., z_n$ и первое определяется, когда последнее фикировано. За эти следует из (21) и (22), что

$$f(z) = (z-\tilde{z})^T \tilde{D}^{-1}(z-\tilde{z}) \leq \chi^2$$

Эвквивалентно можно переписать

$$f(z) = \frac{(z_1-\tilde{z}_1)^2}{\tilde{d}_1} + \frac{(z_2-\tilde{z}_2)^2}{\tilde{d}_2} + ... + \frac{(z_n-\tilde{z}_n)^2}{\tilde{d}_n} \leq \chi^2 \tag{24}$$

Очевидно, что любое z, удовлетворяющее этому неравенству, должно также удовлетворять следующем поэлементным неравенствам:

$$\tilde{z}_n - \tilde{d}_n^{1/2}\chi \leq z_n \leq \tilde{z}_n + \tilde{d}_n^{1/2}, \tag{25}$$

$$\vdots$$

$$\tilde{z}_i - \tilde{d}_i^{1/2}\left[\chi^2 - \sum_{j=i+1}^{n}\frac{(z_j-\tilde{z}_j)^2}{\tilde{d}_j}\right]^{1/2} \leq z_n \leq \tilde{z}_i + \tilde{d}_i^{1/2}\left[\chi^2 - \sum_{j=i+1}^{n}\frac{(z_j-\tilde{z}_j)^2}{\tilde{d}_j}\right]^{1/2}, \tag{26}$$

$$\vdots$$

$$\tilde{z}_1 - \tilde{d}_1^{1/2}\left[\chi^2 - \sum_{j=2}^{n}\frac{(z_j-\tilde{z}_j)^2}{\tilde{d}_j}\right]^{1/2} \leq z_1 \leq \tilde{z}_1 + \tilde{d}_1^{1/2}\left[\chi^2 - \sum_{j=2}^{n}\frac{(z_j-\tilde{z}_j)^2}{\tilde{d}_j}\right]^{1/2}, \qquad (27)$$

Обратите внимание, что неравенства в формуле (27) эквивалентны неравенству (24). На основе этих соотношений может быть получена процедура поиска. Нижняя и верхняя границы z_i определяют интервал, который мы называем уровнем i. Целые числа на этом уровне ищутся напрямую от самых маленьких до самых больших. Каждое возможное целое число на этом уровне будет опробовано, хотя бы единожды. Как только z_i определяется на уровне i, переходят к уровню $i-1$, чтобы определить z_{i-1}. Если на уровне i не может быть найдено целое число, происходит возврат на предыдущий уровень $i+1$, чтобы взять следующее возможное целое число для z_{i+1}, а затем снова происходит переход на уровень i. Как только z_1 определен на уровне 1, найден полный целочисленный вектор z. Затем мы начинаем поиск новых целочисленных векторов. Новый процесс начинается с уровня 1 для поиска всех остальных возможных целых чисел от наименьшего до наибольшего. Процесс поиска завершается, когда все найденные действительные целые числа обработаны.

Одним из важных вопросов в процессе поиска является задание положительной константы χ^2, которая определяет размер эллипсоидальной области. В пакете LAMBDA в процессе поиска размер эллипсоидальной области остается постоянным. Следовательно, производительность процесса поиска будет сильно зависят от значения χ^2. Небольшое значение χ^2 может привести к образованию эллипсоидальной области, не позволяющей минимизизировать исходную задачу (4), в то время как слишком большое значение может сделать процесс слишком трудоёмким для минимизации. Пакет LAMBDA устанавливает значение χ^2 следующим образом. Предположим, что требуется p оптимальных оценок ЦМНК. Если $p \leq n+1$, берётся $z_i = \lfloor \tilde{z}_i \rceil$ для $i = n, n-1, ..., 1$ (т.е. округление каждого $\tilde{z}_i$ до ближайшего целого) в уравнении (23), что дает первый целочисленный вектор $z^{(1)}$, который соответствует так называемой точке Бабая в литературе по теории информации, см. Бабай (Babai, 1986) и Эйрелл (Agrell et al., 2002). Тогда для каждого i $(i = 1, ..., n)$, полученное $\tilde{z}_i$ округляется до ближайшего целого числа, сохраняя $z^{(1)}$, и создаётя новый целочисленный вектор. На основе этих $n+1$ целочисленных векторов, χ^2 устанавливается как p-ое наименьшее решение целевой функции $f(z)$, что гарантирует при минимум p кандидатов в эллипсоидальной области. Если $p > n+1$, объем эллипсоида устанавливается равным p, а затем вычисляется χ^2.

В конце раздела сделаем два замечания. Процесс поиска в пакете LAMBDA фактически основан на LDL^T факторизации $Q_{\hat{a}}^{-1}$, что вычисляется из L^TDL факторизации $Q_{\hat{a}}$, то есть нижняя треугольная матрица первого получается инвертированием нижняя треугольная матрица последнего. Когда оптимальная оценка z, обозначаемая $\check{z}$, найдена, требуется обратное преобразование - $\check{a} = Z^{-1}\check{z}$ (см. уравнение 5). Для по-

дробностей об этом вычислении см. De Jonge and Tiberius (1996), Sects. 3.9 и 4.13.

2.4. Модификации поиска

Константа χ^2, играет важную роль в процессе поиска. Для поиска (единственной) оптимальной оценки ЦМНК, Тёниссен (Teunissen, 1993) предложил использовать стратегию, сокращающую область поиска. Как только целочисленный вектор z в эллипсоидальной области найден, соответствующее значение $f(z)$ в (21) принимается за новое значение для χ^2. Таким образом, эллипсоидальная область сжимается. Как отмечают Де Йонге и Тибериус (De Jonge and Tiberius, 1996), стратегия сокращения может сильно ускоить процесс поиска. Однако эта стратегия не используется в пакете LAMBDA, который находит несколько оптимальных оценок ЦМНК. Чтобы сделать процесс поиска более эффективным, предлагаем расширить стратегию сжатия на случай, когда требуется найти более одной оптимальной оценки. Прежде чем продолжить, мы опишем альтернативу (см. Тёниссен (Teunissen, 1995b, Sect. 2.4) и (De Jonge and Tiberius, 1996, Sect. 4.7)) прямому алгоритму поиска от меньшего к большему на уровне, описанного выше. Мы ищем целые числа по неубывающему расстоянию к $\tilde{z}_i$ на интервале, определённом формулой (26) на уровне i. В частности, если $\tilde{z}_i \leq \lfloor \tilde{z}_i \rceil$, мы используем следуюий порядок поиска:

$$\lfloor \tilde{z}_i \rceil, \lfloor \tilde{z}_i \rceil - 1, \lfloor \tilde{z}_i \rceil + 1, \lfloor \tilde{z}_i \rceil - 2, \dots , \tag{28}$$

иначе используем

$$\lfloor \tilde{z}_i \rceil, \lfloor \tilde{z}_i \rceil + 1, \lfloor \tilde{z}_i \rceil - 1, \lfloor \tilde{z}_i \rceil + 2, \dots . \tag{29}$$

Эта стратегия поиска была также независимо предложена Шнорром и Эйчнером (1994). Теперь мы опишем, как применить стратегию сжатия к процессу поиска, когда требуется больше, чем одна оптимальная оценка ЦМНК. Предположим, нам требуется p оптимальных оценок ЦМНК. Вначале мы устанавливаем χ^2 равным бесконечности. Очевидно, что первым кандидатом, полученным в процессе поиска является

$$z^{(1)} = [\lfloor \tilde{z}_1 \rceil, \lfloor \tilde{z}_2 \rceil, \lfloor \tilde{z}_3 \rceil, \dots , \lfloor \tilde{z}_n \rceil]^T \tag{30}$$

Обратите внимание, что $z^{(1)}$ здесь получается процессом поиска, а не отдельным процессом, что предлагает пакет LAMBDA (см. параграф 4 предыдущего раздела). Возьмем второго кандидата $z^{(2)}$, идентичного $z^{(1)}$, за исключением того, что первая запись в $z^{(2)}$ - второе ближайшее к $\tilde{z}_i$ целое число. И третий $z^{(3)}$ совпадает с $z^{(1)}$, за исключением того, что его первая запись принимается как третье ближайшее целое число к $\tilde{z}_i$ и так далее. Таким образом мы получаем p кандидатов $z^{(1)}$, $z^{(2)}$, $\cdots$, $z^{(p)}$. Очевидно, у нас есть $f(z^{(1)}) \leq f(z^{(2)}) \dots \leq f(z^{(p)})$ (см. уравнение 24). Затем эллипсоидальная область сжимается, задавая $\chi^2 = f(z^{(p)})$. Это альтернатива методу, используемому

методом LAMBDA для задания χ^2 и его главное преимущество в том, что проще определить χ^2. Также, если $p = 2$, вероятно, значение χ^2, определенное этим методом, меньше, чем определенное методом LAMBDA поскольку d_1, вероятно, больше, чем другие d_i после процесса редукции (см. уравнение 24). Затем мы начинаем искать нового кандидата. Мы возвращаемся на уровень 2 и берем следующее действительное целое число для z_2. Продолжать процесс поиска будем, пока мы не найдем нового кандидата на уровне 1. Теперь мы заменяем кандидата $z^(j)$, который удовлетворяет $f(z^(j)) = \chi^2$, новым. Снова сжимаем эллипсоидальную область. Следующее χ^2 принимается как $\underset{1 \le i \le p}{max} f(z^(i))$. Затем мы продолжаем описанный выше процесс до тех пор, пока не сможем найти нового кандидата. Наконец, мы получаем p оптимальных оценок ЦМНК.

3. Реализация

В программе используются три класса: Matrix.h, Lambda.cpp/.h, LambdaTest.cpp/.h. Кратко рассмотрим каждый класс по отдельности.

3.1. Matrix.h

Класс Matrix - простой математический матричный класс, написанный на C++ для хранения и обработки данных в методах класса LAMBDA. Класс Matrix содержит базовые операции, такие как сложение, умножение, доступ к элементам, ввод и вывод, создание единичной матрицы и простые методы решения линейных систем.
Замечание: в файле Matrix.cpp/.h реализован шаблон класса, а не сам класс.
Обзор функций класса изложен в Приложении 1.

3.2. Lambda.cpp/.h

Класс Lambda - класс, реализующий алгоритмы LAMBDA [2]
и MLAMBDA [3] - линейного целочисленного унимодулярного преобразования неоднозначности методом наименьших квадратов. Их математическая трактовка изложена в предыдущей главе(В ПРОЦЕССЕ).
Обзор функций класса изложен в Приложении 2.

3.3. LambdaTest.cpp/.h

Класс LambdaTest тестирует класс Lambda. Внутри LambdaTest.cpp пользователем вводятся веществennозначный вектор неоднозначности и его ковариационная матрица. Реализован вывод результатов на консоль.

Заключение

...

Приложение

Приложение 1. Функции класса Matrix.

Публичное поле:

Matrix<T>(int, int) - шаблонный конструктор, принимающий на вход два целых числа - число строк и столбцов.

Matrix<T>(T **, int, int) - шаблонный конструктор, принимающий на вход массив указателей на другую матрицу и два целых числа - число строк и столбцов.

Matrix<T>() - шаблонный конструктор по умолчанию.

Matrix<T>() - шаблонный деструктор по умолчанию.

Matrix<T>(const Matrix<T> &) - оператор копирования.

Matrix<T> operator=(const Matrix<T> &) - оператор присваивания.

inline T operator()(int x, int y) { return p[x][y]; } - оператор круглых скобок реализует обращение к конкретному элементу матрицы через обращение к массиву указателей. (inline означает, что компилятор при вызове оператора скобок заменяет его на обращение к массиву указателей этой матрицы).

inline T operator()(int x, int y) const { return p[x][y]; } - константный оператор круглых скобок.

Matrix<T> operator+=(const Matrix<T> &) - оператор сложения.

Matrix<T> operator-=(const Matrix<T> &) - оператор вычетания.

Matrix<T> operator*=(const Matrix<T> &) - оператор умножения.

Matrix<T> operator*=(T) - оператор умножения на число.

Matrix<T> operator/=(T) - оператор деления на число.

void swapRows(int, int) - функция, меняющая два ряда матрицы местами.

Matrix<T> pushBackRow(const std::vector<T> &) - метод, добавляющий в матрицу одну строку снизу. Принимает на вход вектор из стандартной библиотеки.

Matrix<T> pushBackColumn(const std::vector<T> &) - метод, добавляющий в матрицу один столбец справа. Принимает на вход вектор из стандартной библиотеки.

Matrix<T> transpose() - метод, транспонирующий матрицу.

static Matrix<T> createIdentity(int) - статитечский метод, создающий единичную матрицу. Принимает на вход целое число - размер квадратичной матрицы.

static Matrix<T> augment(Matrix<T>, Matrix<T>) - статитечский метод, расширающий первую матрицу второй.

Matrix<T> gaussianEliminate() - метод, который находит решение методом Гаусса.

Matrix<T> rowReduceFromGaussian() - метод, уменьшающий число строк в гауссиане.

Matrix<T> inverse() - метод, инвертирующий матрицу.
int getRows() const - константная функция возвращающая число строк матрицы. Обязательно вовращает значение.

int getCols() const - константная функция возвращающая число столбцов матрицы. Обязательно вовращает значение.

Приватное поле:

int rows_{}, cols_{} - число строк и столбцов матрицы.
T **p{} - массив указателей на данные матрицы.

void allocSpace() - функция, выделяющая место для хранения данных матрицы.

Приложение 2. Функции класса Lambda.

Matrix<int> Lambda::computeIntegerSolution
(Matrix<double> &floatAmbiguity,
Matrix<double> &ambiguityCovarianceMatrix) - функция получает на вход вещественнозначный вектор оценки неоднозначности и его ковариационную матрицу и возвращает вектор целочисленного решения.

int Lambda::lambda(const int &m, const Matrix<double> &a,
Matrix<double> &Q, Matrix<double> &F, Matrix<double> &s)
m - число целочисленных решений, a - вещественнозначный вектор оценки неоднозначности, Q - ковариационная матрица a, F - $n \times m$ матрица целочисленных решений, s - сумма квадратов невязок целочисленных решений $m \times 1$. Этот алгоритм вычисляет матрицу целочисленных решений F и вектор невязок s, используя все описанные ниже методы.

int validateSolution(const Matrix<double> S) - функция по сумме квадратов невязок проверяет решение по порогу, который задаёт пользователь.

int Lambda::search(const int &m, const Matrix<double> &L,
const Matrix<double> &D, Matrix<double> &zs, Matrix<double> &zn,
Matrix<double> &s)
m - число целочисленных решений, L и D - части L^TDL факторизации ковариационной матрицы $Q_{\hat{a}}$ размера $n \times n$, zs - $n \times 1$, zn - $n \times 1$, s - сумма квадратов невязок целочисленных решений $m \times 1$. Этот алгоритм находит m оптимальных МНК оценок целочисленного решения.
$Z_b = Z - L^{-T}(Z - Z_s)$
Краткое описание: После процесса редукции запускается процесс дискретного поиска. Чтобы решить проблему МНК, стратегия дискретного поиска используется для перевода текущего подпространства в пространство целых чисел, которое содержит решение. Применяется процедура сжатие эллипсоида. Опишем, как применить стратегию сжатия к процессу поиска, когда требуется более одной оптимальной оценки МНК. Предположим, нам требуется m оптимальных оценок МНК. Вначале мы установили $maxDist$ (т.ч. $f(z) = (z - z_s)^T Q_{z_s}^{-1}(z - z_s) \leq maxDist$) равным бесконечности Очевидно, что первым кандидатом, полученным в процессе поиска, является округлённый до целого значения вектор решений z. Затем вычисляем второго кандидата равного второму ближайшему целому числу к z_i. И так далее. Таким образом мы получаем m кандидатов $z^{(1)}, z^{(2)}, ..., z^{(m)}$. Очевидно, что $f(z^{(1)}) \leq f(z^{(2)}) \leq ... \leq f(z^{(m)})$. Затем эллипсоидальную область сжимают, полагая $maxDist = f(z^{(m)})$. Затем мы

начинаем поиск нового кандидата. Мы возвращаемся на уровень 2 и берем следующее действительное целое число для $z^{(2)}$. Процесс поиска продолжается, пока мы не найдем нового кандидата на уровне 1. Теперь мы заменяем кандидата $z^{(j)}$, который удовлетворяет $f(z^{(j)}) = maxDist$, новым. Снова сжимаем эллипсоидальную область. Более новый $maxDist$ принимается как $max_{1 \le i \le m} f(z^{(i)})$. Затем мы продолжаем описанный выше процесс до тех пор, пока не сможем найти нового кандидата. Наконец, мы получаем m оптимальных оценок МНК.

void Lambda::reduction(Matrix<double> &L, Matrix<double> &D, Matrix<double> &Z) - LAMBDA-редукция. Пусть $Q_{\hat{a}}$ ковариационная матрица вещественной оценки $\hat{a}$ решения размера. Алгоритм вычисляет целочисленную униодулярную матрицу Z и $L^T DL$ фактроизацию $Q_{\hat{z}} = Z^T Q_{\hat{a}} Z = L^T DL$, где L и D - части $L^T DL$ факторизации ковариационной матрицы $Q_{\hat{a}}$ размера $n \times n$, этот алгоритм также вычисляет $\hat{z} = Z^T \hat{a}$
Краткое описание. Процесс редукции начинается с предпоследнего столбца L и последней пары диагональных элементов D и пытается достичь первого столбца L и первой пары диагональных элементов D. Когда алгоритм впервые встречает индекс k, алгоритм сначала выполняет целочисленное преобразование Гаусса на L такое, что абсолютные значения элементов ниже l_{jj} ограничены сверху числом $1/2$, и тогда для пары $(k, k+1)$ происходит перестановка, если для $del = \hat{d}_{j+1} = d_j + l^2_{j+1,j} d_{j+1}$ выполняется условие, что $del < d_{j+1}$. После перестановки алгоритм перезапускается, т.е. возвращается обратно в исходное положение. Алгоритм использует переменную k для отслеживания тех столбцов, чьи недиагональные элементы по величине уже ограничены сверху $1/2$ из-за предыдущих целочисленных преобразований Гаусса, чтобы в процессе перезапуска никакие новые преобразования для этих столбцов не выполнялись.

void Lambda::permutations(Matrix<double> &L,
Matrix<double> &D, int j, double del, Matrix<double> &Z) - функция, реализующая перестановки. L и D - части $L^T DL$ факторизации матрицы Q размера $n \times n$, j - индекс, del - скаляр (из $L^T DL$ преобразования $del = \hat{d}_{k+1} = d_k + l^2_{k+1,k} d_{k+1}$), Z - (целочисленная) матрица размера $n \times n$.
Краткое описание. Чтобы добиться неубывания элементов матрицы D по возрастанию индекса, нужны симметричные перестановки матрицы $Q_{\hat{a}}$. Для этого в функции переставляются элементы факторизации матрицы $Q_{\hat{a}}$ - L и D. eta, lam, $temp1$, $temp2$ - времнные локальные перемынные для рассчётов.

void Lambda::gaussTransformation(Matrix<double> &L,
Matrix<double> &Z, int i, int j) целочисленное преобразование Гаусса. L - нижняя треугольная матрица размера $n \times n$, $(i,\ j)$ - пара индексов, Z - (целочисленная) мат-

рица размера $n \times n$.
Краткое описание. На входе имеется пара индексов. Сперва округляется l_{ij}. Далее к L применяется целочисленное преобразование Гаусса Z_{ij}, такое что $|(LZ)(i,j)| < 1/2$.

int Lambda::factorization(const Matrix<double> &Q,
Matrix<double> &L, Matrix<double> &D) - $L^T DL$ факторизация матрицы Q ($Q = L^T DL$, где L - нижняя треугольная матрица и $D = diag(d_1, ..., d_n),\ d_i > 0$)
Краткое описание. Создаётся копия матрицы Q. Сначала диагональные элементы записываются в вектор D. Потом расчитывается нижняя треугольная матрица L.

Список литературы

[1] RTKLib. Library // An Open Source Program Package for GNSS Positioning. — URL: http://www.rtklib.com/.

[2] Teunissen P. J. G. The least-squares ambiguity decorrelation adjustment: a method for fast GPS integer ambiguity estimation. — Journal of Geodesy, 1995. — Research Gate : https://www.researchgate.net/publication/226909388_The_least - squares_ambiguity_decorrelation_adjustment_Its_performance_on_short_GPS_baselines_

[3] X.-W. Chang X. Yang T. Zhou. MLAMBDA: A modified LAMBDA method for integer least-squares estimation. — Journal of Geodesy, 2005. — Research Gate : https://www.researchgate.net/publication/225518977_MLAMBDA_a_modified_LAMBDA_ squares_estimation.

[4] Подкорытов Андрей Николаевич. Высокоточное местоопределение в глобальных навигационных спутниковых системах в абсолютном режиме за счёт разрешения неоднозначности псевдофазовых измерений. — Моск. гос. авиац. ин-т, 2014.