Министерство науки и высшего образования Российской Федерации
Федеральное государственное бюджетное образовательное учреждение
высшего образования
«Тульский государственный университет»

Кафедра прикладной математики и информатики

УТВЕРЖДАЮ
Зав. кафедрой ПМиИ
_____ В.И. Иванов
«___» _____ 20___ г.

ПОЯСНИТЕЛЬНАЯ ЗАПИСКА

к курсовой работе по курсу

«МЕТОДЫ ОПТИМИЗАЦИИ»

на тему

Итеративный алгоритм ближайших точек (ICP-алгоритм)

Автор работы __24.12.2020_____ студент гр. _221271__ ____Окороков М.В.____
(дата, подпись) (фамилия и инициалы)

Руководитель работы _24.12.2020_____ __проф._____ ____Баранов В.П._____
(дата, подпись) (должность) (фамилия и инициалы)

Работа защищена ____24.12.2020_____ с оценкой _____
(дата)

Члены комиссии _____ __проф. _____ __Баранов В.П._____
(дата, подпись) (должность) (фамилия и инициалы)

_____ _зав. каф. ПМиИ_ __Иванов В.И._____
(дата, подпись) (должность) (фамилия и инициалы)

_____ __доц._____ ___Ларин Н.В._____
(дата, подпись) (должность) (фамилия и инициалы)

Тула 2020

УТВЕРЖДАЮ
Зав. кафедрой ПМиИ
_____ В.И. Иванов
«___» _____ 20___ г.

ЗАДАНИЕ

на курсовую работу по курсу
«МЕТОДЫ ОПТИМИЗАЦИИ»

студенту гр. _221271_____ __Окорокову Максиму Витальевичу_____
(фамилия, имя, отчество)

Тема работы __Итеративный алгоритм ближайших точек (ICP-алгоритм)_____
_____
_____

Входные данные _Содержательная постановка задачи, исходные данные для контрольных задач, требования к представлению результатов работы, список рекомендуемой литературы._____
_____
_____

Задание получил _____ ____09.09.2020_____
(подпись) (дата)

График выполнения работы _I неделя – получение задания и его изучение, заполнение бланка заданий; II-IV неделя – изучение литературы и других необходимых данных; V-XIII неделя – разработка программы, написание теоретического материала; XIV-XV неделя – оформление пояснительной записки и сдача на проверку; 24.12.2020 – защита курсовой работы._____
_____
_____

Замечания консультанта _____
_____
_____
_____

К защите. Консультант работы _____ ___24.12.2020_____
(подпись) (дата)

РЕФЕРАТ

Целью данной курсовой работы является рассмотрение итеративного алгоритма ближайших точек (ICP). Вопросы, требующие рассмотрения по данной теме, были полностью разобраны в данной курсовой работе. Пояснительная записка, общим объемом 41 стр., содержит 4 рисунка, 7 использованных источников.

Ключевые слова: экстремальная задача, оптимизируемая функция, итеративный алгоритм ближайших точек (ICP), трехмерное сканирование, облако точек, совмещение, движение, матрица вращения, вектор сдвига, метод наименьших квадратов, сингулярное разложение, взвешенная среднеквадратичная метрика, след матрицы.

СОДЕРЖАНИЕ

Введение …….. 5

1. Сферы применения результатов, полученных при решении экстремальных задач с помощью ICP-алгоритма…….. 7

2. Описание итеративного алгоритма ближайших точек (ICP)……..……. 8

2.1. ICP-алгоритм для случая, если X и Y представляют собой одно и то же зашумленное облако, но по-разному расположенное в пространстве... 9

2.2. ICP-алгоритм для случая, если X и Y похожи, но состоят из разного числа точек……..... 12

3. Пример нахождения матрицы вращения и вектора сдвига для случая, описанного в п. 2.1……..... 15

Заключение……..... 25

Библиографический список ……..... 26

Приложение..... 27

ВВЕДЕНИЕ

При решении научных и технических задач ученые и инженеры всегда стараются найти наилучшие и наиболее оптимальные пути их решения. Для анализа всего множества вариантов решения строятся математические модели, в которых на языке математики описываются поставленные задачи. Общим для этих моделей является то, что в них поиск наилучших вариантов сводится к поиску наилучшего (экстремального) значения функции (функционала). Такие модели получили название экстремальные задачи, а функции стали называться оптимизируемыми [1].

Первые экстремальные задачи были сформулированы еще в древности. Они касались максимизации площадей и объемов. Так, в легенде о финикийской царевне Дидоне, которая основала около 825 года до н.э. город Карфаген, говорится, что царевна, найдя подходящее место на берегу Средиземного моря, выкупила у местного князя столько земли, сколько можно было покрыть бычьей шкурой. Разрезав шкуру на тонкие полоски и связав их в один длинный ремень, Дидона окружила им максимальную площадь и заложила на ней крепость Бирса. Известна также экстремальная задача древнегреческого математика Евклида: в данный треугольник вписать параллелограмм наибольшей площади. В XVI веке появляются первые экстремальные задачи алгебраического содержания. Широко известной стала задача итальянского математика Н. Тартальи: разделить число восемь на два числа так, чтобы произведение этих чисел на их разность было максимальным. Только после открытий известных математиков П. Ферма, И. Ньютона, Г. Лейбница постепенно начал формироваться общий подход к решению экстремальных задач, использующий дифференцирование оптимизируемых функций [2].

Формулировка в конце XVII века И. Бернулли задачи о брахистохроне (кривой скорейшего спуска) положила начало изучению нового класса экстремальных задач – задач на условный экстремум. Основной вклад в решение таких задач внес французский математик Ж. Лагранж, предложивший знаме-

нитое правило множителей. На это правило опиралась группа советских математиков, возглавляемая Л. Понтрягиным, при разработке математической теории оптимального управления. До начала 40-х годов XX века экстремальные задачи формулировались преимущественно в области математики, физики и техники. Однако в начале 40-х годов появляются экономические экстремальные задачи. Формулируемые задачи представляли собой совершенно иной класс экстремальных задач на условный экстремум, не поддающиеся решению методом Лагранжа и, таким образом, требующие разработки новых подходов к поиску экстремальных значений оптимизируемых функций [3].

Данная курсовая работа посвящена рассмотрению итеративного алгоритма ближайших точек (ICP) для решения экстремальных задач. Цель данной курсовой работы: рассмотрение итеративного алгоритма ближайших точек (ICP) для решения экстремальных задач. В рамках данной курсовой работы будут проанализированы следующие вопросы:

– история возникновения и исследования экстремальных задач;

– сферы применения результатов, полученных при решении экстремальных задач с помощью итеративного алгоритма ближайших точек (ICP);

– описание итеративного алгоритма ближайших точек (ICP);

– пример решения экстремальной задачи с помощью итеративного алгоритма ближайших точек (ICP);

– описание программы, реализующей итеративный алгоритм ближайших точек (ICP).

Демонстрироваться данные методы будут с помощью C# – одного из самых современных языков программирования, подходящего для решения широкого класса задач. Программа будет рассмотрена в среде программирования Microsoft Visual Studio 2015.

1. СФЕРЫ ПРИМЕНЕНИЯ РЕЗУЛЬТАТОВ, ПОЛУЧЕННЫХ ПРИ РЕШЕНИИ ЭКСТРЕМАЛЬНЫХ ЗАДАЧ С ПОМОЩЬЮ ICP-АЛГОРИТМА

Для автоматизации и ускорения процесса перевода поверхности модели в цифровой формат используется трехмерное сканирование. Во время построения цифровых моделей появляется необходимость сканирования объекта с нескольких ракурсов в связи с перекрытием некоторых частей детали. Результат сканирования представляет собой поверхность объекта, которая описана с помощью облака точек. Полученный скан и модель должны быть выровнены в общую систему отсчета для дальнейшей обработки с целью измерения и классификации возможных ошибок в производстве.

В последнее время популярность устройств 3D-сканирования (например, Microsoft Kinect) привела к растущему интересу в области надежных алгоритмов совмещения. Требуется разрабатывать такие алгоритмы, которые способны иметь дело с большими объемами совмещаемых объектов и обеспечивать высокое качество сканирования [4]. Один из таких алгоритмов – итеративный алгоритм ближайших точек (ICP).

Итеративный алгоритм ближайших точек (Iterative Closest Point, ICP) предназначен для совмещения в пространстве двух множеств (облаков) точек. Эта задача актуальна, например, для получения трехмерной модели объекта путем склеивания его снимков, сделанных с разных точек 3D-камерой. Также алгоритм ближайших точек имеет широкое применение в промышленной робототехнике. Он позволяет решать задачи позиционирования манипулятора во время обработки детали. Благодаря сканеру и итеративному алгоритму ближайших точек можно вычислить траекторию движения захвата манипулятора в связанной с ним системой координат для обработки произвольно закрепленной детали [5].

2. ОПИСАНИЕ ИТЕРАТИВНОГО АЛГОРИТМА БЛИЖАЙШИХ ТОЧЕК (ICP)

Приведем математическую постановку экстремальной задачи, которая решается с помощью итеративного алгоритма ближайших точек.

Под облаком точек понимается множество $A=\{a_i\}_{i=1}^{N}\subset\mathbb{R}^3$. Будем также интерпретировать это множество как матрицу $3\times N$, в которой векторы $a_i\in\mathbb{R}^3$ записаны по столбцам. Пусть X и Y – два облака. Например, X и Y могут быть по-разному расположенными в пространстве и содержащими разное число точек точечными представлениями пересекающихся частей одной и той же поверхности. Требуется при помощи движений наложить облако X на облако Y [6]. Пример такого наложения представлен на рис. 2.1.

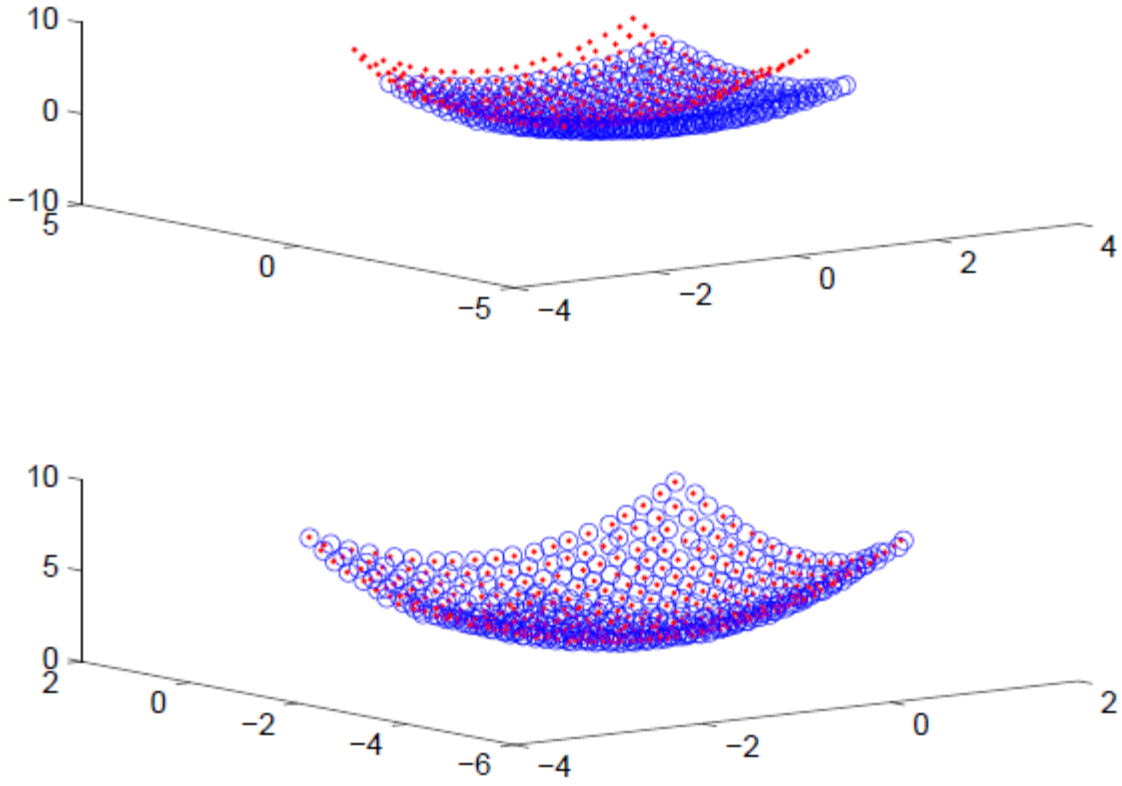

Рис. 2.1. Вверху представлены исходные облака, внизу – совмещенные.

Движением M в пространстве (преобразованием, не изменяющим длин) являются вращение и сдвиг:

$$M(x) = Rx + t, \tag{2.1}$$

где $x \in \mathbb{R}^3$ (вектор-столбец), $R \in \mathbb{R}^{3\times3}$ – матрица вращения (ортогональная матрица, для которой $RR^T = I$), $t \in \mathbb{R}^3$ – вектор сдвига.

Таким образом, если X и Y представляют собой одно и то же зашумленное облако, но по-разному расположенное в пространстве, то

$$Y = M(X) = RX + t + E \Leftrightarrow y_i = Rx_i + t + e_i,\ i = 1,...,N, \tag{2.2}$$

где e_i – ошибки. Необходимо найти движение $M = (R,t)$, для которого ошибка $E = \{e_i\}_{i=1}^N$ будет наименьшей:

$$\|E\| \to \min. \tag{2.3}$$

Здесь норма выбирается из специфики задачи. Чаще всего это среднеквадратичная норма:

$$\|E\|^2 = \frac{1}{N}\sum_{i=1}^{N}|e_i|^2. \tag{2.4}$$

Если Y и X хоть и похожи, но состоят из разного числа точек, то нужно найти такую последовательность мелких движений:

$$M_k = (R_k, t_k), k = 1,...,K, \tag{2.5}$$

чтобы $M_K(X)$ было в каком-то смысле наиболее близко к Y [6].

2.1. ICP-алгоритм для случая, если X и Y представляют собой одно и то же зашумленное облако, но по-разному расположенное в пространстве.

Вначале рассмотрим случай, когда $Y = RX + t + E$. Требуется по известным X и Y определить матрицу вращения R и вектор переноса t. Для их нахождения воспользуемся методом наименьших квадратов и будем решать экстремальную задачу:

$$\|E\|^2 = \|Y - M(X)\|^2 \to \min_M \Leftrightarrow \frac{1}{N}\sum_{i=1}^{N} |y_i - (Rx_i + t)|^2 \to \min_{R,t}, \tag{2.6}$$

где минимум берется по ортогональным матрицам R и векторам t. Матрица $\hat{R}$ и вектор $\hat{t}$, для которых этот минимум будет равен нулю, будут искомым движением:

$$\hat{M}(X,Y) = (\hat{R}, \hat{t}). \tag{2.7}$$

Целевая функция здесь называется функцией ошибки. Решим данную экстремальную задачу [6].

Для множества $X = \{x_i\}_{i=1}^N \subset \mathbb{R}^3$ через $\overline{x} = \frac{1}{N}\sum x_i$ обозначим его центр. Из равенства $Y = RX + t$ имеем:

$$\overline{y} = R\overline{x} + t, \tag{2.8}$$

откуда

$$t = \overline{y} - R\overline{x}. \tag{2.9}$$

Следовательно, достаточно найти R.

Пусть $\overline{X} = X - \overline{x}$ – центрированное множество X. Тогда

$$R\overline{X} = R(X - \overline{x}) = RX - R\overline{x} = Y - t - (\overline{y} - t) \Rightarrow \overline{Y} = R\overline{X}. \tag{2.10}$$

Для нахождения вращения R из равенства $B = RA$ решаем, как и выше, экстремальную задачу:

$$\|B - RA\|^2 = \frac{1}{N}\sum_{i=1}^{N} |b_i - Ra_i|^2 \to \min_R. \tag{2.11}$$

Для векторов $a, b \in \mathbb{R}^3$ справедливо равенство:

$$|b - Ra|^2 = b^2 + (Ra)^2 - 2bRa. \tag{2.12}$$

При этом $(Ra)^2 = a^2$ для ортогональной матрицы R (вращение не изменяет длину векторов). Таким образом, опуская отрицательный множитель и слагаемые, не зависящие от R, получаем задачу:

$$\sum_{i=1}^{N} b_i Ra_i = tr(B^T RA) = tr(RAB^T) \to \max_R. \tag{2.13}$$

Здесь $tr(M) = \sum_i m_{ii}$ – след матрицы $M = (m_{ij})$. Для него:

$$tr(M_1 M_2) = tr(M_2 M_1). \tag{2.14}$$

Пусть $C = AB^T \in \mathbb{R}^{3\times 3}$. Найдем ее сингулярное разложение (SVD-разложение)

$$C = USV^T, \tag{2.15}$$

где S – диагональная матрица сингулярных чисел матрицы C, U и V – ортогональные матрицы. Тогда

$$tr(RAB^T) = tr(RUSV^T) = tr(SV^T RU). \tag{2.16}$$

Здесь матрица $Q = V^T RU$ является ортогональной. Из равенства $QQ^T = I$ следует, что $|q_{ii}| \le 1$, причем $q_{ii} = 1\ \forall i$ только для единичной матрицы $Q = I$. Кроме того, сингулярные числа $s_{ii} \ge 0$. Поэтому

$$tr(RAB^T) = tr(SQ) = \sum_i s_{ii} q_{ii} \le \sum_i s_{ii}. \tag{2.17}$$

Равенство здесь достигается, если

$$q_{ii} = 1\ \forall i \Leftrightarrow Q = V^T RU = I \Rightarrow R = VU^T. \tag{2.18}$$

Итеративный алгоритм ближайших точек в простейшем случае может быть описан следующим образом [6].

Пусть даны X, Y, для которых $Y = RX + t$. Требуется найти R, t.

1. Определяем центры $\overline{x}, \overline{y}$.

2. Находим матрицу $C = \overline{X}\overline{Y}^T$.

3. Вычисляем сингулярное разложение (SVD-разложение) $C = USV^T$.

4. Находим искомые матрицу вращения $R = VU^T$ и вектор сдвига $t = \overline{y} - R\overline{x}$.

Можно немного обобщить приведенные результаты за счет использования взвешенной среднеквадратичной метрики:

$$\|E\|_W^2 = \sum_{i=1}^{N} w_i |e_i|^2 , \tag{2.19}$$

где $w_i \geq 0$ – заданные весовые коэффициенты. Ранее они были одинаковыми. Теперь же можно учесть степень влияния разных точек, а также исключать некоторые из них, полагая соответствующие веса равными нулю. Для задачи (2.13) при использовании взвешенной нормы получаем:

$$\sum_{i=1}^{N} w_i b_i R a_i = tr(B_W^T RA) , \tag{2.20}$$

где $B_W = \{w_i b_i\}_{i=1}^N$. Приходим к той же экстремальной задаче, заменив B на B_W.

2.2. ICP-алгоритм для случая, если X и Y похожи, но состоят из разного числа точек.

Теперь рассмотрим общий случай, когда множества X и Y состоят из разного числа точек и не существует изначального соответствия индексов точек X и Y как в случае, описанном в п. 2.1. [6]. В нем i-й точке X однозначно соответствовала i-я точка Y, что сильно облегчало задачу.

Сформулируем идею алгоритма. Облако X за несколько итераций небольшими движениями притягивается к облаку Y. Желательно, чтобы изначально облака были достаточно хорошо расположены относительно друг друга. Для этого можно, например, поместить облака в ориентируемые пря-

моугольные боксы или в общие выпуклые тела (k-dop) и совместить последние за счет сдвига в общий центр и применения вращения, рассчитанного по методу из п. 2.1.

Теперь опишем одну итерацию ICP [6]. Текущее положение облака X обозначим той же буквой. Облако Y на каждом шаге зафиксировано. Требуется найти текущее движение $M=(R,t)$, для которого облако $M(X)$ будет наиболее близко к Y. Вращение и сдвиг можно было бы определить по формулам из п. 2.1. Однако в данном случае нет соответствия между точками X и Y. Установление этого соответствия является непростой задачей. Воспользуемся способом, когда в качестве прицельной точки для $x \in X$ берется ближайшая к x точка $y^* \in Y$:

$$y^* = \arg\min_{y \in Y} |x-y|. \tag{2.21}$$

Расстояние между x и y^* обозначим через d^*. В результате получаем множество Y^* точек, соответствующих точкам X, и можем определить движение по методике из п. 2.1:

$$\left\| Y^* - M(X) \right\| \to \min_M. \tag{2.22}$$

Однако некоторые точки x и y^* могут быть слишком далеки друг от друга и вносить большие погрешности. Тогда их лучше не учитывать. Для этого паре (x, y^*) можно поставить в соответствие вес:

$$w = \begin{cases} 1, \ d^* < d_{\max}, \\ 0, \ d^* \ge d_{\max}, \end{cases} \tag{2.23}$$

где $d_{\max} > 0$ – параметр алгоритма – максимальная дистанция, когда пары точек учитываются. После этого решаем задачу:

$$\left\| Y^* - M(X) \right\|_W \to \min_M. \tag{2.24}$$

При практической реализации такого подхода самой трудоемкой частью оказывается решение задачи определения y^* для облака Y из десятков и сотен

тысяч вершин. Поэтому для ускорения поиска здесь необходимо использовать соответствующие пространственные структуры, например kd-дерево для Y. Без этого алгоритм ICP не эффективен.

3. ПРИМЕР НАХОЖДЕНИЯ МАТРИЦЫ ВРАЩЕНИЯ И ВЕКТОРА СДВИГА ДЛЯ СЛУЧАЯ, ОПИСАННОГО В П. 2.1.

Проиллюстрируем теперь наши рассуждения примером. Создадим тестовую матрицу A:

$$A = \begin{pmatrix} 0 & 3 & 3 & 3 \\ 0 & 0 & 3 & 3 \\ 0 & 0 & 0 & 3 \end{pmatrix}$$

и тестовый вектор сдвига (вектор переноса) t_0:

$$t_0 = \begin{pmatrix} 0.5 \\ -1 \\ 1.5 \end{pmatrix}.$$

Как было указано выше, в рамках данной курсовой работы также была написана соответствующая программа на современном языке программирования C#. Код программы приведен в ПРИЛОЖЕНИИ.

Определим следующие переменные:

$$r_x = t_0[0,0] = 0.5$$
$$r_y = t_0[1,0] = -1$$
$$r_z = t_0[2,0] = 1.5.$$

Образуем матрицы вращения вокруг координатных осей декартовой системы координат. Получим следующие матрицы:

$$R_x = \begin{pmatrix} 1 & 0 & 0 \\ 0 & \cos(rx) & -\sin(rx) \\ 0 & \sin(rx) & \cos(rx) \end{pmatrix} = \begin{pmatrix} 1 & 0 & 0 \\ 0 & 0.8776 & -0.4794 \\ 0 & 0.4794 & 0.8776 \end{pmatrix}$$ – матрица вращения вокруг оси x.

$$R_y = \begin{pmatrix} \cos(ry) & 0 & \sin(ry) \\ 0 & 1 & 0 \\ -\sin(ry) & 0 & \cos(ry) \end{pmatrix} = \begin{pmatrix} 0.5403 & 0 & -0.8415 \\ 0 & 1 & 0 \\ 0.8415 & 0 & 0.5403 \end{pmatrix}$$ – матрица вращения вокруг оси y.

$$R_z = \begin{pmatrix} \cos(rz) & -\sin(rz) & 0 \\ \sin(rz) & \cos(rz) & 0 \\ 0 & 0 & 1 \end{pmatrix} = \begin{pmatrix} 0.0707 & -0.9975 & 0 \\ 0.9975 & 0.0707 & 0 \\ 0 & 0 & 1 \end{pmatrix}$$ – матрица вращения вокруг оси z.

Вычислим матрицу R_0 как произведение матриц вращения R_x, R_y и R_z:

$$R_0 = R_x R_y R_z = \begin{pmatrix} 1 & 0 & 0 \\ 0 & 0.8776 & -0.4794 \\ 0 & 0.4794 & 0.8776 \end{pmatrix} \begin{pmatrix} 0.5403 & 0 & -0.8415 \\ 0 & 1 & 0 \\ 0.8415 & 0 & 0.5403 \end{pmatrix} \begin{pmatrix} 0.0707 & -0.9975 & 0 \\ 0.9975 & 0.0707 & 0 \\ 0 & 0 & 1 \end{pmatrix} =$$
$$= \begin{pmatrix} 0.5403 & 0 & -0.8415 \\ -0.4034 & 0.8776 & -0.259 \\ 0.7385 & 0.4794 & 0.4742 \end{pmatrix} \begin{pmatrix} 0.0707 & -0.9975 & 0 \\ 0.9975 & 0.0707 & 0 \\ 0 & 0 & 1 \end{pmatrix} = \begin{pmatrix} 0.0382 & -0.5389 & -0.8415 \\ 0.8469 & 0.4644 & -0.259 \\ 0.5304 & -0.7028 & 0.4742 \end{pmatrix}.$$

Образуем матрицу T_0 размерности 1×4, каждый элемент которой представляет собой вектор сдвига t_0 размерности 3×1, т.е. фактически получим матрицу размерности 3×4:

$$T_0 = \begin{pmatrix} 0.5 & 0.5 & 0.5 & 0.5 \\ -1 & -1 & -1 & -1 \\ 1.5 & 1.5 & 1.5 & 1.5 \end{pmatrix}.$$

Вычислим матрицу $B = R_0 \cdot A + T_0$:

$$B = \begin{pmatrix} 0.0382 & -0.5389 & -0.8415 \\ 0.8469 & 0.4644 & -0.259 \\ 0.5304 & -0.7028 & 0.4742 \end{pmatrix} \begin{pmatrix} 0 & 3 & 3 & 3 \\ 0 & 0 & 3 & 3 \\ 0 & 0 & 0 & 3 \end{pmatrix} + \begin{pmatrix} 0.5 & 0.5 & 0.5 & 0.5 \\ -1 & -1 & -1 & -1 \\ 1.5 & 1.5 & 1.5 & 1.5 \end{pmatrix} =$$
$$= \begin{pmatrix} 0 & 0.1146 & -1.5021 & -4.0266 \\ 0 & 2.5407 & 3.9339 & 3.1569 \\ 0 & 1.5912 & -0.5172 & 0.9054 \end{pmatrix} + \begin{pmatrix} 0.5 & 0.5 & 0.5 & 0.5 \\ -1 & -1 & -1 & -1 \\ 1.5 & 1.5 & 1.5 & 1.5 \end{pmatrix} = \begin{pmatrix} 0.5 & 0.6146 & -1.0021 & -3.5266 \\ -1 & 1.5407 & 2.9339 & 2.1569 \\ 1.5 & 3.0912 & 0.9828 & 2.4054 \end{pmatrix}.$$

Введем в рассмотрение матрицы X и Y:

$$X = A = \begin{pmatrix} 0 & 3 & 3 & 3 \\ 0 & 0 & 3 & 3 \\ 0 & 0 & 0 & 3 \end{pmatrix}; \quad Y = B = \begin{pmatrix} 0.5 & 0.6146 & -1.0021 & -3.5266 \\ -1 & 1.5407 & 2.9339 & 2.1569 \\ 1.5 & 3.0912 & 0.9828 & 2.4054 \end{pmatrix}.$$

Найдем матрицу вращения R и вектор сдвига t, для которых $Y = RX + t$. Определим центры $\overline{x}$ и $\overline{y}$. Для этого вычислим среднее арифметическое элементов каждой строки для матриц X и Y. Получим вектор-столбцы x_c и y_c:

$$x_c = \begin{pmatrix} 2.25 \\ 1.5 \\ 0.75 \end{pmatrix}; \qquad y_c = \begin{pmatrix} -0.8535 \\ 1.4079 \\ 1.9949 \end{pmatrix}.$$

Образуем матрицы X_0 и Y_0 размерности 1×4. Каждый элемент матрицы X_0 представляет собой вектор-столбец x_c размерности 3×1, т.е. фактически получим матрицу размерности 3×4. Каждый элемент матрицы Y_0 представляет собой вектор-столбец y_c размерности 3×1, т.е. фактически получим матрицу размерности 3×4. Получим матрицы X_0 и Y_0:

$$X_0 = \begin{pmatrix} 2.25 & 2.25 & 2.25 & 2.25 \\ 1.5 & 1.5 & 1.5 & 1.5 \\ 0.75 & 0.75 & 0.75 & 0.75 \end{pmatrix}; \quad Y_0 = \begin{pmatrix} -0.8535 & -0.8535 & -0.8535 & -0.8535 \\ 1.4079 & 1.4079 & 1.4079 & 1.4079 \\ 1.9949 & 1.9949 & 1.9949 & 1.9949 \end{pmatrix}.$$

Вычислим матрицы X_c и Y_c:

$$X_c = X - X_0 = \begin{pmatrix} 0 & 3 & 3 & 3 \\ 0 & 0 & 3 & 3 \\ 0 & 0 & 0 & 3 \end{pmatrix} - \begin{pmatrix} 2.25 & 2.25 & 2.25 & 2.25 \\ 1.5 & 1.5 & 1.5 & 1.5 \\ 0.75 & 0.75 & 0.75 & 0.75 \end{pmatrix} = \begin{pmatrix} -2.25 & 0.75 & 0.75 & 0.75 \\ -1.5 & -1.5 & 1.5 & 1.5 \\ -0.75 & -0.75 & -0.75 & 2.25 \end{pmatrix}$$

$$Y_c = Y - Y_0 = \begin{pmatrix} 0.5 & 0.6146 & -1.0021 & -3.5266 \\ -1 & 1.5407 & 2.9339 & 2.1569 \\ 1.5 & 3.0912 & 0.9828 & 2.4054 \end{pmatrix} - \begin{pmatrix} -0.8535 & -0.8535 & -0.8535 & -0.8535 \\ 1.4079 & 1.4079 & 1.4079 & 1.4079 \\ 1.9949 & 1.9949 & 1.9949 & 1.9949 \end{pmatrix} =$$

$$= \begin{pmatrix} 1.3535 & 1.4681 & -0.1486 & -2.6731 \\ -2.4079 & 0.1328 & 1.526 & 0.749 \\ -0.4949 & 1.0963 & -1.0121 & 0.4105 \end{pmatrix}.$$

Найдем матрицу $C = X_c \cdot Y_c^T$:

$$C = \begin{pmatrix} -2.25 & 0.75 & 0.75 & 0.75 \\ -1.5 & -1.5 & 1.5 & 1.5 \\ -0.75 & -0.75 & -0.75 & 2.25 \end{pmatrix} \begin{pmatrix} 1.3535 & 1.4681 & -0.1486 & -2.6731 \\ -2.4079 & 0.1328 & 1.526 & 0.749 \\ -0.4949 & 1.0963 & -1.0121 & 0.4105 \end{pmatrix}^T =$$

$$= \begin{pmatrix} -2.25 & 0.75 & 0.75 & 0.75 \\ -1.5 & -1.5 & 1.5 & 1.5 \\ -0.75 & -0.75 & -0.75 & 2.25 \end{pmatrix} \begin{pmatrix} 1.3535 & -2.4079 & -0.4949 \\ 1.4681 & 0.1328 & 1.0963 \\ -0.1486 & 1.526 & -1.0121 \\ -2.6731 & 0.749 & 0.4105 \end{pmatrix} = \begin{pmatrix} -4.0606 & 7.2236 & 1.4846 \\ -8.465 & 6.8252 & -1.8179 \\ -8.0192 & 2.2471 & 1.2317 \end{pmatrix}.$$

Вычислим сингулярное разложение (SVD – разложение) матрицы C [7]. Для этого получим транспонированную матрицу C^T:

$$C^T = \begin{pmatrix} -4.0606 & -8.465 & -8.0192 \\ 7.2236 & 6.8252 & 2.2471 \\ 1.4846 & -1.8179 & 1.2317 \end{pmatrix}.$$

Вычислим произведение транспонированной матрицы C^T на матрицу C:

$$C^T C = \begin{pmatrix} -4.0606 & -8.465 & -8.0192 \\ 7.2236 & 6.8252 & 2.2471 \\ 1.4846 & -1.8179 & 1.2317 \end{pmatrix} \begin{pmatrix} -4.0606 & 7.2236 & 1.4846 \\ -8.465 & 6.8252 & -1.8179 \\ -8.0192 & 2.2471 & 1.2317 \end{pmatrix} =$$

$$= \begin{pmatrix} 152.4523 & -105.1274 & -0.5171 \\ -105.1274 & 103.8132 & 1.0844 \\ -0.5171 & 1.0844 & 7.0259 \end{pmatrix}.$$

Найдем собственные значения матрицы $C^T C$:

$$\left| C^T C - \lambda E \right| = \begin{vmatrix} 152.4523 - \lambda & -105.1274 & -0.5171 \\ -105.1274 & 103.8132 - \lambda & 1.0844 \\ -0.5171 & 1.0844 & 7.0259 - \lambda \end{vmatrix} = 0.$$

Запишем характеристический многочлен матрицы:

$$-\lambda^3 + 263.2914\lambda^2 - 6573.8433\lambda + 33458.0711 = 0.$$

Решив данное кубическое уравнение, найдем собственные значения:

$$\lambda_1 = 236.0416, \lambda_2 = 20.25, \lambda_3 = 6.9998.$$

Вычислим сингулярные числа как квадратные корни из соответствующих собственных значений:

$$\rho_1 = \sqrt{\lambda_1} = \sqrt{236.0416} = 15.3636$$
$$\rho_2 = \sqrt{\lambda_2} = \sqrt{20.25} = 4.5$$
$$\rho_3 = \sqrt{\lambda_3} = \sqrt{6.9998} = 2.6457.$$

Образуем матрицу Λ:

$$\Lambda = \begin{pmatrix} \rho_1 & 0 & 0 \\ 0 & \rho_2 & 0 \\ 0 & 0 & \rho_3 \end{pmatrix} = \begin{pmatrix} 15.3636 & 0 & 0 \\ 0 & 4.5 & 0 \\ 0 & 0 & 2.6457 \end{pmatrix}.$$

Вычислим собственные векторы для соответствующих собственных значений:

$$\overline{e}_1 = \begin{pmatrix} 0.7827 \\ -0.6223 \\ -0.0047 \end{pmatrix}; \quad \overline{e}_2 = \begin{pmatrix} 0.622 \\ 0.782 \\ 0.0398 \end{pmatrix}; \quad \overline{e}_3 = \begin{pmatrix} -0.0211 \\ -0.0341 \\ 0.9992 \end{pmatrix}.$$

Образуем матрицу V^* размерности 1×3, элементы которой представляют собой собственные векторы $\overline{e}_1, \overline{e}_2, \overline{e}_3$, т.е. имеем матрицу размерности 3×3:

$$V^* = \begin{pmatrix} 0.7827 & 0.622 & -0.0211 \\ -0.6223 & 0.782 & -0.0341 \\ -0.0047 & 0.0398 & 0.9992 \end{pmatrix}.$$

Матрица V^* представляет собой сингулярный базис. Вычислим векторы $\overline{e}_1', \overline{e}_2', \overline{e}_3'$ следующим образом:

$$\overline{e}_1' = \frac{1}{\rho_1} C\overline{e}_1 = \frac{1}{15.3636} \begin{pmatrix} -4.0606 & 7.2236 & 1.4846 \\ -8.465 & 6.8252 & -1.8179 \\ -8.0192 & 2.2471 & 1.2317 \end{pmatrix} \begin{pmatrix} 0.7827 \\ -0.6223 \\ -0.0047 \end{pmatrix} =$$
$$= \begin{pmatrix} -0.2643 & 0.4702 & 0.0966 \\ -0.551 & 0.4442 & -0.1183 \\ -0.522 & 0.1463 & 0.0802 \end{pmatrix} \begin{pmatrix} 0.7827 \\ -0.6223 \\ -0.0047 \end{pmatrix} = \begin{pmatrix} -0.4999 \\ -0.7071 \\ -0.5 \end{pmatrix}$$

$$\overline{e}_2' = \frac{1}{\rho_2} C\overline{e}_2 = \frac{1}{4.5} \begin{pmatrix} -4.0606 & 7.2236 & 1.4846 \\ -8.465 & 6.8252 & -1.8179 \\ -8.0192 & 2.2471 & 1.2317 \end{pmatrix} \begin{pmatrix} 0.622 \\ 0.782 \\ 0.0398 \end{pmatrix} =$$
$$= \begin{pmatrix} -0.9024 & 1.6052 & 0.3299 \\ -1.8811 & 1.5167 & -0.404 \\ -1.782 & 0.4994 & 0.2737 \end{pmatrix} \begin{pmatrix} 0.622 \\ 0.782 \\ 0.0398 \end{pmatrix} = \begin{pmatrix} 0.7071 \\ -0.0000018 \\ -0.707 \end{pmatrix}$$

$$\overline{e}_3' = \frac{1}{\rho_3} C\overline{e}_3 = \frac{1}{2.6457}\begin{pmatrix} -4.0606 & 7.2236 & 1.4846 \\ -8.465 & 6.8252 & -1.8179 \\ -8.0192 & 2.2471 & 1.2317 \end{pmatrix}\begin{pmatrix} -0.0211 \\ -0.0341 \\ 0.9992 \end{pmatrix} =$$

$$= \begin{pmatrix} -1.5348 & 2.7303 & 0.5611 \\ -3.1995 & 2.5797 & -0.6871 \\ -3.031 & 0.8493 & 0.4655 \end{pmatrix}\begin{pmatrix} -0.0211 \\ -0.0341 \\ 0.9992 \end{pmatrix} = \begin{pmatrix} 0.5 \\ -0.707 \\ 0.5001 \end{pmatrix}.$$

Образуем матрицу U размерности 1×3, элементы которой представляют собой векторы $\overline{e}_1', \overline{e}_2', \overline{e}_3'$, т.е. имеем матрицу размерности 3×3:

$$U = \begin{pmatrix} -0.4999 & 0.7071 & 0.5 \\ -0.7071 & -0.0000018 & -0.707 \\ -0.5 & -0.707 & 0.5001 \end{pmatrix}.$$

Таким образом, мы осуществили сингулярное разложение матрицы C:

$$C = U\Lambda V^{*T}.$$

Сделаем проверку:

$$U\Lambda V^{*T} = \begin{pmatrix} -0.4999 & 0.7071 & 0.5 \\ -0.7071 & -0.0000018 & -0.707 \\ -0.5 & -0.707 & 0.5001 \end{pmatrix}\begin{pmatrix} 15.3636 & 0 & 0 \\ 0 & 4.5 & 0 \\ 0 & 0 & 2.6457 \end{pmatrix}\begin{pmatrix} 0.7827 & 0.622 & -0.0211 \\ -0.6223 & 0.782 & -0.0341 \\ -0.0047 & 0.0398 & 0.9992 \end{pmatrix}^T =$$

$$= \begin{pmatrix} -7.6803 & 3.18195 & 1.32285 \\ -10.8636 & -0.0000081 & -1.8705 \\ -7.6818 & -3.1815 & 1.3231 \end{pmatrix}\begin{pmatrix} 0.7827 & -0.6223 & -0.0047 \\ 0.622 & 0.782 & 0.0398 \\ -0.0211 & -0.0341 & 0.9992 \end{pmatrix} = \begin{pmatrix} -4.0601 & 7.2226 & 1.4845 \\ -8.4635 & 6.8242 & -1.8179 \\ -8.0194 & 2.2473 & 1.2315 \end{pmatrix}.$$

Разложение выполнено правильно, т.к. полученный результат совпадает с матрицей C с точностью до 1.5×10^{-3}.

Находим искомую матрицу вращения $R = V^* U^T$ и вектор сдвига $t = y_c - Rx_c$.

$$R = V^* U^T = \begin{pmatrix} 0.7827 & 0.622 & -0.0211 \\ -0.6223 & 0.782 & -0.0341 \\ -0.0047 & 0.0398 & 0.9992 \end{pmatrix}\begin{pmatrix} -0.4999 & 0.7071 & 0.5 \\ -0.7071 & -0.0000018 & -0.707 \\ -0.5 & -0.707 & 0.5001 \end{pmatrix}^T =$$

$$= \begin{pmatrix} 0.7827 & 0.622 & -0.0211 \\ -0.6223 & 0.782 & -0.0341 \\ -0.0047 & 0.0398 & 0.9992 \end{pmatrix}\begin{pmatrix} -0.4999 & -0.7071 & -0.5 \\ 0.7071 & -0.0000018 & -0.707 \\ 0.5 & -0.707 & 0.5001 \end{pmatrix} = \begin{pmatrix} 0.038 & -0.5385 & -0.8417 \\ 0.847 & 0.4641 & -0.2588 \\ 0.5301 & -0.7031 & 0.4739 \end{pmatrix}$$

$$t = y_c - Rx_c = \begin{pmatrix} -0.8535 \\ 1.4079 \\ 1.9949 \end{pmatrix} - \begin{pmatrix} 0.038 & -0.5385 & -0.8417 \\ 0.847 & 0.4641 & -0.2588 \\ 0.5301 & -0.7031 & 0.4739 \end{pmatrix} \begin{pmatrix} 2.25 \\ 1.5 \\ 0.75 \end{pmatrix} = \begin{pmatrix} -0.8535 \\ 1.4079 \\ 1.9949 \end{pmatrix} - \begin{pmatrix} -1.3535 \\ 2.4078 \\ 0.4935 \end{pmatrix} =$$

$$= \begin{pmatrix} 0.5 \\ -0.9999 \\ 1.5014 \end{pmatrix}.$$

Вычислим также матрицу $R_{откл}$ и вектор-столбец $t_{откл}$. Матрица $R_{откл}$ будет показывать, насколько отличаются матрицы R_0 и R, а вектор-столбец $t_{откл}$ будет показывать, насколько отличаются вектор-столбцы t_0 и t.

$$R_{откл} = |R_0 - R| = \left| \begin{pmatrix} 0.0382 & -0.5389 & -0.8415 \\ 0.8469 & 0.4644 & -0.259 \\ 0.5304 & -0.7028 & 0.4742 \end{pmatrix} - \begin{pmatrix} 0.038 & -0.5385 & -0.8417 \\ 0.847 & 0.4641 & -0.2588 \\ 0.5301 & -0.7031 & 0.4739 \end{pmatrix} \right| =$$

$$= \begin{pmatrix} 0.0002 & 0.0004 & 0.0002 \\ 0.0001 & 0.0003 & 0.0002 \\ 0.0003 & 0.0003 & 0.0003 \end{pmatrix}$$

$$t_{откл} = |t_0 - t| = \left| \begin{pmatrix} 0.5 \\ -1 \\ 1.5 \end{pmatrix} - \begin{pmatrix} 0.5 \\ -0.9999 \\ 1.5014 \end{pmatrix} \right| = \begin{pmatrix} 0 \\ 0.0001 \\ 0.0014 \end{pmatrix}.$$

Результаты работы программы представлены на рис. 2-4. Сравнивая их с результатами, полученными при аналитическом решении задачи, можно сделать вывод, что полученные при аналитическом решении матрица вращения R и вектор сдвига t совпали с результатами работы программы с точностью до 1.5×10^{-3} (более точные результаты возможно получить, если удастся избежать ошибок в округлении). Данный факт свидетельствует о том, что теоретический материал данной курсовой работы усвоен и проработан в полном объеме; написанная программа работает корректно.

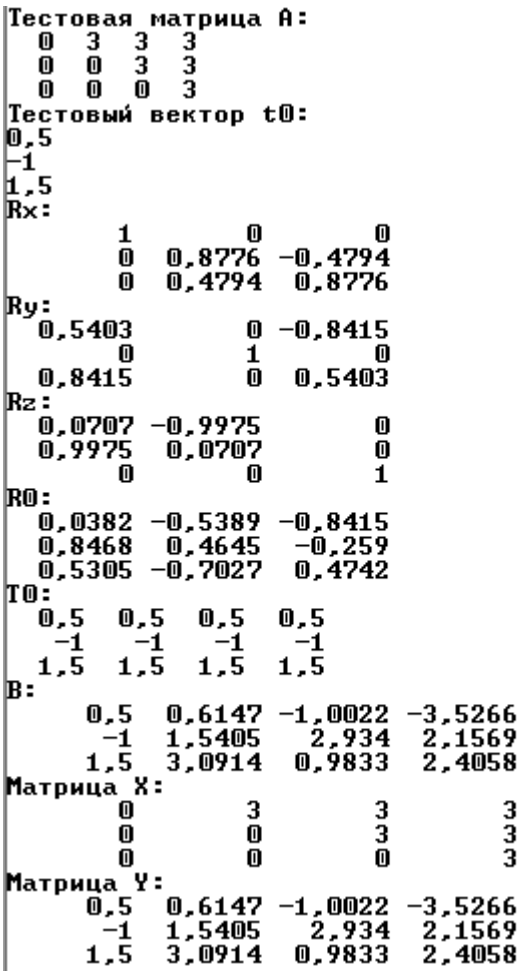

Рис. 3.1. Скриншот экрана, демонстрирующий результат работы кода, написанного в рамках данной курсовой работы и реализующего итеративный алгоритм ближайших точек (ICP) для решения задачи, описанной в п. 2.1.

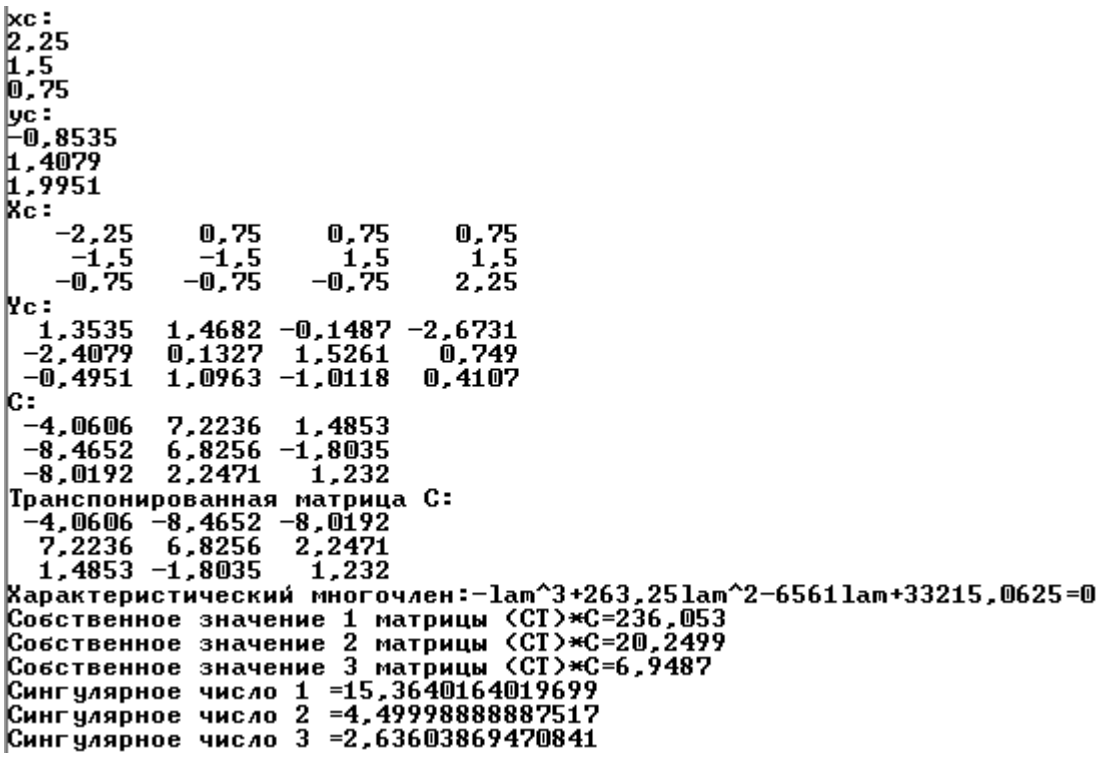

Рис. 3.2. Скриншот экрана, демонстрирующий результат работы кода, написанного в рамках данной курсовой работы и реализующего итеративный алгоритм ближайших точек (ICP) для решения задачи, описанной в п. 2.1.

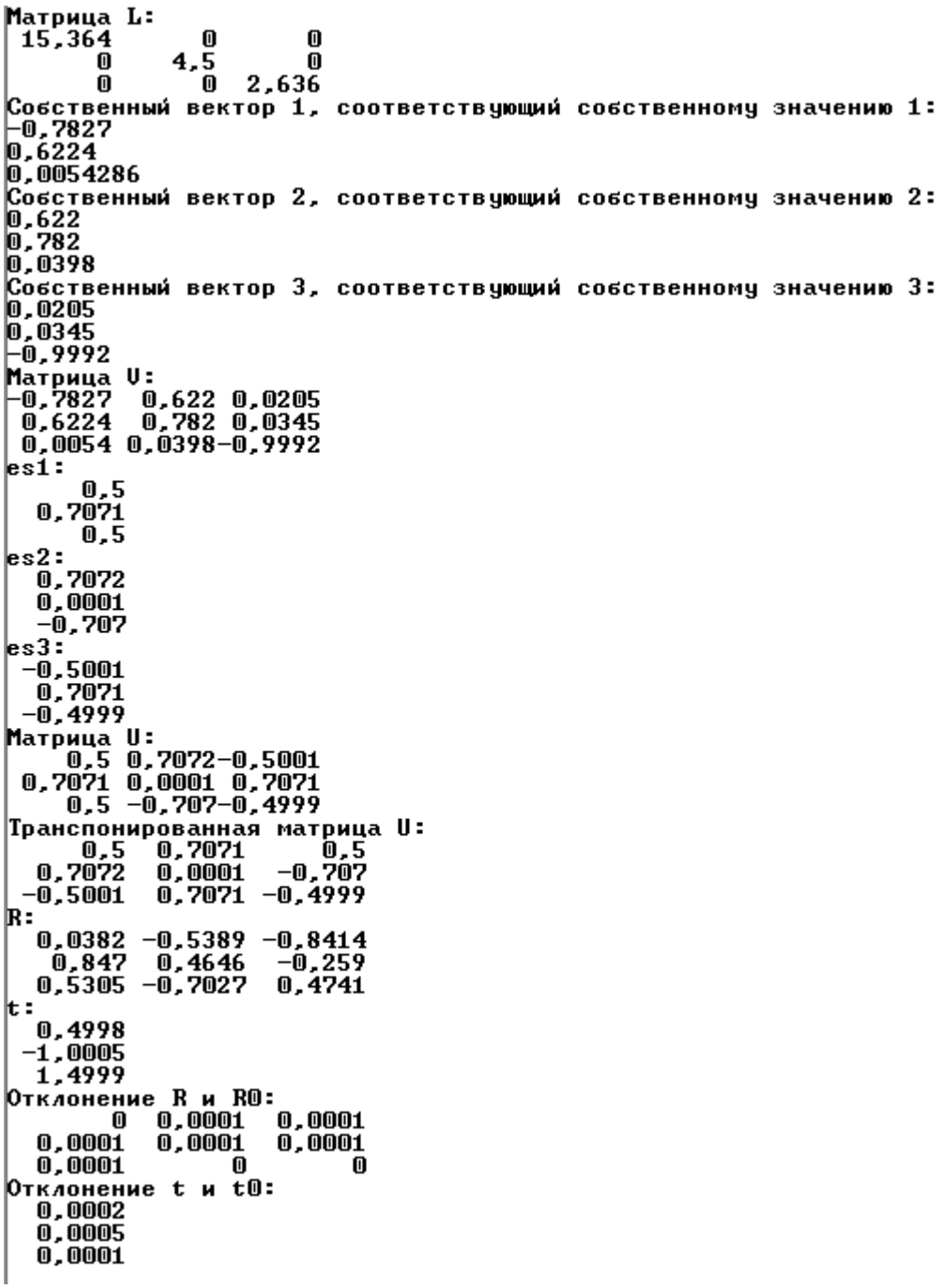

Рис. 3.3. Скриншот экрана, демонстрирующий результат работы кода, написанного в рамках данной курсовой работы и реализующего итеративный алгоритм ближайших точек (ICP) для решения задачи, описанной в п. 2.1.

ЗАКЛЮЧЕНИЕ

В рамках данной курсовой работы была рассмотрена история возникновения и исследования экстремальных задач, а также были отмечены сферы применения результатов, полученных при решении экстремальных задач с помощью итеративного алгоритма ближайших точек (ICP). В ходе работы был подробно разобран и описан итеративный алгоритм ближайших точек (ICP), а также приведена математическая постановка задачи, которую алгоритм ICP позволяет решить. В данной курсовой работе был рассмотрен пример решения экстремальной задачи с помощью итеративного алгоритма ближайших точек (ICP). Сопоставление результатов работы программы с данными, полученными при аналитическом решении экстремальной задачи, позволяет сделать вывод о том, что теоретический материал данной курсовой работы усвоен и проработан в полном объеме, написанная программа работает правильно. Приведен соответствующий код, реализующий итеративный алгоритм ближайших точек (ICP). Цель, поставленная во введении, была достигнута. Вопросы, требующие рассмотрения по данной теме, были полностью разобраны в данной курсовой работе.

БИБЛИОГРАФИЧЕСКИЙ СПИСОК

[1] Евтушенко Ю.Г. Методы решения экстремальных задач и их применение в системах оптимизации. М.: Наука, 1982. 432 с.

[2] Романовский И.В. Алгоритмы решения экстремальных задач. М.: Наука, 1977. 352 с.

[3] Тихомиров В.М. Рассказы о максимумах и минимумах. М.: Наука, 1986. 192 с.

[4] Тозик В.Т. 3ds Max. Трехмерное моделирование и анимация на примерах. СПб.: BHV, 2008. 880 с.

[5] Besl P.J., McKay N.D. A Method for Registration of 3-D Shapes // IEEE Transactions on Pattern Analysis and Machine Intelligence. 1992. V. 14, №2. P. 239-256.

[6] Горбачев Д.В. Численные методы решения экстремальных задач: учеб. пособие. Тула: Изд-во ТулГУ, 2014. 114 с.

[7] Логинов Н.В. Сингулярное разложение матриц: учеб. пособие. М.: МГАПИ, 1995. 80 с.

ПРИЛОЖЕНИЕ

Пример кода, реализующего алгоритм ICP для случая, описанного в п. 2.1.

В данном приложении представлен пример кода итеративного алгоритма ближайших точек для нахождения матрицы вращений и вектора сдвига в случае, если X и Y представляют собой одно и то же зашумленное облако, но по-разному расположенное в пространстве.

```
using System;
using System.Collections.Generic;
using System.Linq;
using System.Text;
using System.Threading.Tasks;

namespace ConsoleApplication86
{
    class Program
    {
        static void Main(string[] args)
        {
            int[,] A = new int[3, 4];
            A[0, 0] = 0;
            A[0, 1] = 3;
            A[0, 2] = 3;
            A[0, 3] = 3;
            A[1, 0] = 0;
            A[1, 1] = 0;
            A[1, 2] = 3;
            A[1, 3] = 3;
            A[2, 0] = 0;
            A[2, 1] = 0;
            A[2, 2] = 0;
            A[2, 3] = 3;
            Console.WriteLine("Тестовая матрица A:");
            for (int i = 0; i < A.GetLength(0); i++)
```

```
{
    for (int j = 0; j < A.GetLength(1); j++)
    {
        Console.Write("{0,3}", A[i, j]);
    }
    Console.WriteLine();
}
int N = A.GetLength(1);
double[] t0 = { 0.5, -1, 1.5 };
Console.WriteLine("Тестовый вектор t0:");
for (int i = 0; i < 3; i++)
{
    Console.WriteLine(t0[i]);
}
double rx = t0[0];
double ry = t0[1];
double rz = t0[2];
double[,] Rx = new double[3, 3];
Rx[0, 0] = 1;
Rx[0, 1] = 0;
Rx[0, 2] = 0;
Rx[1, 0] = 0;
Rx[1, 1] = Math.Cos(rx);
Rx[1, 2] = -Math.Sin(rx);
Rx[2, 0] = 0;
Rx[2, 1] = Math.Sin(rx);
Rx[2, 2] = Math.Cos(rx);
Console.WriteLine("Rx:");
for (int i = 0; i < 3; i++)
{
    for (int j = 0; j < 3; j++)
    {
        Console.Write("{0,8}", Math.Round(Rx[i,j], 4));
    }
    Console.WriteLine();
}
double[,] Ry = new double[3, 3];
Ry[0, 0] = Math.Cos(ry);
```

```
Ry[0, 1] = 0;
Ry[0, 2] = Math.Sin(ry);
Ry[1, 0] = 0;
Ry[1, 1] = 1;
Ry[1, 2] = 0;
Ry[2, 0] = -Math.Sin(ry);
Ry[2, 1] = 0;
Ry[2, 2] = Math.Cos(ry);
Console.WriteLine("Ry:");
for (int i = 0; i < 3; i++)
{
    for (int j = 0; j < 3; j++)
    {
        Console.Write("{0,8}", Math.Round(Ry[i, j], 4));
    }
    Console.WriteLine();
}
double[,] Rz = new double[3, 3];
Rz[0, 0] = Math.Cos(rz);
Rz[0, 1] = -Math.Sin(rz);
Rz[0, 2] = 0;
Rz[1, 0] = Math.Sin(rz);
Rz[1, 1] = Math.Cos(rz);
Rz[1, 2] = 0;
Rz[2, 0] = 0;
Rz[2, 1] = 0;
Rz[2, 2] = 1;
Console.WriteLine("Rz:");
for (int i = 0; i < 3; i++)
{
    for (int j = 0; j < 3; j++)
    {
        Console.Write("{0,8}", Math.Round(Rz[i, j], 4));
    }
    Console.WriteLine();
}
double[,] temp = new double[3, 3];
for (int i = 0; i < temp.GetLength(0); i++)
```

```
{
    for (int j = 0; j < temp.GetLength(1); j++)
    {
        temp[i, j] = 0;
    }
}
double[,] R0 = new double[3, 3];
for (int i = 0; i < Rx.GetLength(0); i++)
{
    for (int j = 0; j < Ry.GetLength(1); j++)
    {
        for (int s = 0; s < Rx.GetLength(1); s++)
        {
            temp[i, j] += Rx[i, s] * Ry[s, j];
        }
    }
}
Console.WriteLine("R0:");
for (int i = 0; i < temp.GetLength(0); i++)
{
    for (int j = 0; j < Rz.GetLength(1); j++)
    {
        for (int s = 0; s < temp.GetLength(1); s++)
        {
            R0[i, j] += temp[i, s] * Rz[s, j];
        }

        Console.Write("{0,8}", Math.Round(R0[i, j], 4));
    }
    Console.WriteLine();
}
double[,] T0 = new double[3, N];
Console.WriteLine("T0:");
for (int i = 0; i < T0.GetLength(0); i++)
{
    for (int j = 0; j < T0.GetLength(1); j++)
    {
        T0[i, j] = t0[i];
```

```
            Console.Write("{0,5}", T0[i, j]);
        }
        Console.WriteLine();
    }
    double[,] B = new double[A.GetLength(0), A.GetLength(1)];
    Console.WriteLine("B:");
    for (int i = 0; i < R0.GetLength(0); i++)
    {
        for (int j = 0; j < A.GetLength(1); j++)
        {
            for (int s = 0; s < R0.GetLength(1); s++)
            {
                B[i, j] += R0[i, s] * A[s, j];
            }
            B[i, j] += T0[i, j];
            Console.Write("{0,8}", Math.Round(B[i, j], 4));
        }
        Console.WriteLine();
    }
    double[,] X = new double[A.GetLength(0), A.GetLength(1)];
    double[,] Y = new double[B.GetLength(0), B.GetLength(1)];
    Console.WriteLine("Матрица X:");
    for (int i = 0; i < X.GetLength(0); i++)
    {
        for (int j = 0; j < X.GetLength(1); j++)
        {
            X[i, j] = A[i, j];
            Console.Write("{0,8}", Math.Round(X[i, j], 4));
        }
        Console.WriteLine();
    }
    Console.WriteLine("Матрица Y:");
    for (int i = 0; i < Y.GetLength(0); i++)
    {
        for (int j = 0; j < Y.GetLength(1); j++)
        {
            Y[i, j] = B[i, j];
            Console.Write("{0,8}", Math.Round(Y[i, j], 4));
```

```
    }
    Console.WriteLine();
}
double[,] xc = new double[X.GetLength(0), 1];
double[,] yc = new double[Y.GetLength(0), 1];
for (int i = 0; i < xc.GetLength(0); i++)
{

    xc[i, 0] = 0;

}
Console.WriteLine("xc:");
for (int i = 0; i < X.GetLength(0); i++)
{
    for (int j = 0; j < X.GetLength(1); j++)
    {
        xc[i, 0] += X[i, j];
    }
    xc[i, 0] = xc[i, 0] / X.GetLength(1);
    Console.WriteLine(Math.Round(xc[i, 0], 4));
}
for (int i = 0; i < yc.GetLength(0); i++)
{

    yc[i, 0] = 0;

}
Console.WriteLine("yc:");
for (int i = 0; i < Y.GetLength(0); i++)
{
    for (int j = 0; j < Y.GetLength(1); j++)
    {
        yc[i, 0] += Y[i, j];
    }
    yc[i, 0] = yc[i, 0] / Y.GetLength(1);
    Console.WriteLine(Math.Round(yc[i, 0], 4));
}
double[,] Temp0 = new double[3, N];
```

```
for (int i = 0; i < Temp0.GetLength(0); i++)
{
    for (int j = 0; j < Temp0.GetLength(1); j++)
    {
        Temp0[i, j] = xc[i, 0];
    }
}
double[,] Temp1 = new double[3, N];
for (int i = 0; i < Temp1.GetLength(0); i++)
{
    for (int j = 0; j < Temp1.GetLength(1); j++)
    {
        Temp1[i, j] = yc[i, 0];
    }
}
double[,] Xc = new double[Temp0.GetLength(0), Temp0.GetLength(1)];
Console.WriteLine("Xc:");
for (int i = 0; i < Xc.GetLength(0); i++)
{
    for (int j = 0; j < Xc.GetLength(1); j++)
    {
        Xc[i, j] = X[i, j] - Temp0[i, j];
        Console.Write("{0,8}", Math.Round(Xc[i, j], 4));
    }
    Console.WriteLine();
}
double[,] Yc = new double[Temp1.GetLength(0), Temp1.GetLength(1)];
Console.WriteLine("Yc:");
for (int i = 0; i < Yc.GetLength(0); i++)
{
    for (int j = 0; j < Yc.GetLength(1); j++)
    {
        Yc[i, j] = Y[i, j] - Temp1[i, j];
        Console.Write("{0,8}", Math.Round(Yc[i, j], 4));
    }
    Console.WriteLine();
}
double[,] Yctrans = new double[Yc.GetLength(1), Y.GetLength(0)];
```

```
for (int i = 0; i < Yctrans.GetLength(0); i++)
{
    for (int j = 0; j < Yctrans.GetLength(1); j++)
    {
        Yctrans[i, j] = Yc[j, i];
    }
}
double[,] C = new double[Xc.GetLength(0), Yctrans.GetLength(1)];
Console.WriteLine("C:");
for (int i = 0; i < Xc.GetLength(0); i++)
{
    for (int j = 0; j < Yctrans.GetLength(1); j++)
    {
        for (int s = 0; s < Xc.GetLength(1); s++)
        {
            C[i, j] += Xc[i, s] * Yctrans[s, j];
        }
        Console.Write("{0,8}", Math.Round(C[i, j], 4));
    }
    Console.WriteLine();
}
double[,] Ctrans = new double[Yctrans.GetLength(1), Xc.GetLength(0)];
Console.WriteLine("Транспонированная матрица C: ");
for (int i = 0; i < Yctrans.GetLength(1); i++)
{
    for (int j = 0; j < Xc.GetLength(0); j++)
    {
        Ctrans[i, j] = C[j, i];
        Console.Write("{0,8}", Math.Round(Ctrans[i, j], 4));
    }
    Console.WriteLine();
}
double[,] Proizv = new double[Ctrans.GetLength(0), C.GetLength(1)];
for (int i = 0; i < Ctrans.GetLength(0); i++)
{
    for (int j = 0; j < C.GetLength(1); j++)
    {
        for (int s = 0; s < Ctrans.GetLength(1); s++)
```

```
                {
                    Proizv[i, j] += Ctrans[i, s] * C[s, j];
                }
            }
        }
        double a1 = -1;
        double a2 = Proizv[0, 0] + Proizv[1, 1] + Proizv[2, 2];
        double a3 = -Proizv[0, 0] * Proizv[1, 1] - Proizv[0, 0] * Proizv[2, 2] - Proizv[1, 1]
* Proizv[2, 2] + Proizv[1, 2] * Proizv[2, 1] + Proizv[0, 1] * Proizv[1, 0] + Proizv[0, 2] * Pro-
izv[2, 0];
        double a4 = Proizv[0, 0] * Proizv[1, 1] * Proizv[2, 2] - Proizv[0, 0] * Proizv[1, 2]
* Proizv[2, 1] - Proizv[0, 1] * Proizv[1, 0] * Proizv[2, 2] + Proizv[0, 1] * Proizv[1, 2] * Pro-
izv[2, 0] + Proizv[0, 2] * Proizv[1, 0] * Proizv[2, 1] - Proizv[0, 2] * Proizv[2, 0] * Proizv[1,
1];
        Console.WriteLine("Характеристический многочлен:-lam^3+{0}lam^2{1}lam+{2}=0",
Math.Round(a2,3), Math.Round(a3,1), Math.Round(a4,4));
        double lam1 = 236.053;
        double lam2 = 20.2499;
        double lam3 = 6.9487;
        Console.WriteLine("Собственное значение 1 матрицы (CT)*C={0}", lam1);
        Console.WriteLine("Собственное значение 2 матрицы (CT)*C={0}", lam2);
        Console.WriteLine("Собственное значение 3 матрицы (CT)*C={0}", lam3);
        double p1 = Math.Sqrt(lam1);
        double p2 = Math.Sqrt(lam2);
        double p3 = Math.Sqrt(lam3);
        Console.WriteLine("Сингулярное число 1 ={0}", p1);
        Console.WriteLine("Сингулярное число 2 ={0}", p2);
        Console.WriteLine("Сингулярное число 3 ={0}", p3);
        double[,] L = new double[3, 3];
        for (int i = 0; i < 3; i++)
        {
            for (int j = 0; j < 3; j++)
            {
                L[i, j] = 0;
            }
        }
        L[0, 0] = p1;
        L[1, 1] = p2;
        L[2, 2] = p3;
        Console.WriteLine("Матрица L:");
        for (int i = 0; i < 3; i++)
```

```
{
    for (int j = 0; j < 3; j++)
    {
        Console.Write("{0,7}", Math.Round(L[i, j], 4));
    }
    Console.WriteLine();
}
double[,] e1 = new double[3, 1];
double[,] e2 = new double[3, 1];
double[,] e3 = new double[3, 1];
e1[0, 0] = -0.7827;
e1[1, 0] = 0.6224;
e1[2, 0] = 0.0054286;
e2[0, 0] = 0.622;
e2[1, 0] = 0.782;
e2[2, 0] = 0.0398;
e3[0, 0] = 0.0205;
e3[1, 0] = 0.0345;
e3[2, 0] = -0.9992;
Console.WriteLine("Собственный вектор 1, соответствующий собственному значению 1:");
for (int i = 0; i < 3; i++)
{
    Console.WriteLine(e1[i, 0]);
}
Console.WriteLine("Собственный вектор 2, соответствующий собственному значению 2:");
for (int i = 0; i < 3; i++)
{
    Console.WriteLine(e2[i, 0]);
}
Console.WriteLine("Собственный вектор 3, соответствующий собственному значению 3:");
for (int i = 0; i < 3; i++)
{
    Console.WriteLine(e3[i, 0]);
}
double[,] V = new double[3, 3];
V[0, 0] = e1[0, 0];
V[0, 1] = e2[0, 0];
V[0, 2] = e3[0, 0];
```

```
V[1, 0] = e1[1, 0];
V[1, 1] = e2[1, 0];
V[1, 2] = e3[1, 0];
V[2, 0] = e1[2, 0];
V[2, 1] = e2[2, 0];
V[2, 2] = e3[2, 0];
double[,] Vt = new double[3, 3];
Vt[0, 0] = e1[0, 0];
Vt[0, 1] = e1[1, 0];
Vt[0, 2] = e1[2, 0];
Vt[1, 0] = e2[0, 0];
Vt[1, 1] = e2[1, 0];
Vt[1, 2] = e2[2, 0];
Vt[2, 0] = e3[0, 0];
Vt[2, 1] = e3[1, 0];
Vt[2, 2] = e3[2, 0];
Console.WriteLine("Матрица V:");
for (int i = 0; i < 3; i++)
{
    for (int j = 0; j < 3; j++)
    {
        Console.Write("{0,7}", Math.Round(V[i, j], 4));
    }
    Console.WriteLine();
}
double[,] es1 = new double[3, 1];
double[,] es2 = new double[3, 1];
double[,] es3 = new double[3, 1];
Console.WriteLine("es1:");
for (int i = 0; i < 3; i++)
{
    for (int j = 0; j < 1; j++)
    {
        for (int s = 0; s < 3; s++)
        {
            es1[i, j] += C[i, s] * e1[s, j];
        }
        es1[i, j] = es1[i, j] / p1;
```

```
            Console.Write("{0,8}", Math.Round(es1[i, j], 4));
        }
        Console.WriteLine();
    }
    Console.WriteLine("es2:");
    for (int i = 0; i < 3; i++)
    {
        for (int j = 0; j < 1; j++)
        {
            for (int s = 0; s < 3; s++)
            {
                es2[i, j] += C[i, s] * e2[s, j];
            }
            es2[i, j] = es2[i, j] / p2;
            Console.Write("{0,8}", Math.Round(es2[i, j], 4));
        }
        Console.WriteLine();
    }
    Console.WriteLine("es3:");
    for (int i = 0; i < 3; i++)
    {
        for (int j = 0; j < 1; j++)
        {
            for (int s = 0; s < 3; s++)
            {
                es3[i, j] += C[i, s] * e3[s, j];
            }
            es3[i, j] = es3[i, j] / p3;
            Console.Write("{0,8}", Math.Round(es3[i, j], 4));
        }
        Console.WriteLine();
    }
    double[,] U = new double[3, 3];
    U[0, 0] = es1[0, 0];
    U[0, 1] = es2[0, 0];
    U[0, 2] = es3[0, 0];
    U[1, 0] = es1[1, 0];
    U[1, 1] = es2[1, 0];
```

```
U[1, 2] = es3[1, 0];
U[2, 0] = es1[2, 0];
U[2, 1] = es2[2, 0];
U[2, 2] = es3[2, 0];
Console.WriteLine("Матрица U:");
for (int i = 0; i < 3; i++)
{
    for (int j = 0; j < 3; j++)
    {
        Console.Write("{0,7}", Math.Round(U[i, j], 4));
    }
    Console.WriteLine();
}
double[,] Utrans = new double[3, 3];
Console.WriteLine("Транспонированная матрица U: ");
for (int i = 0; i < 3; i++)
{
    for (int j = 0; j < 3; j++)
    {
        Utrans[i, j] = U[j, i];
        Console.Write("{0,8}", Math.Round(Utrans[i, j], 4));
    }
    Console.WriteLine();
}
double[,] time = new double[3, 3];
for (int i = 0; i < 3; i++)
{
    for (int j = 0; j < 3; j++)
    {
        for (int s = 0; s < 3; s++)
        {
            time[i, j] += U[i, s] * L[s, j];
        }
    }
}
double[,] singular = new double[3, 3];
for (int i = 0; i < 3; i++)
{
```

```
    for (int j = 0; j < 3; j++)
    {
        for (int s = 0; s < 3; s++)
        {
            singular[i, j] += time[i, s] * Vt[s, j];
        }
    }
}
double[,] R = new double[3, 3];
Console.WriteLine("R:");
for (int i = 0; i < 3; i++)
{
    for (int j = 0; j < 3; j++)
    {
        for (int s = 0; s < 3; s++)
        {
            R[i, j] += V[i, s] * Utrans[s, j];
        }
        Console.Write("{0,8}", Math.Round(R[i, j], 4));
    }
    Console.WriteLine();
}
Console.WriteLine("t:");
double[] t = new double[3];
for (int i = 0; i < 3; i++)
{
    for (int j = 0; j < 1; j++)
    {
        for (int s = 0; s < 3; s++)
        {
            t[i] += R[i, s] * xc[s, j];
        }
        t[i] = yc[i, 0] - t[i];
        Console.Write("{0,8}", Math.Round(t[i], 4));
    }
    Console.WriteLine();
}
double[,] razR = new double[3, 3];
```

```
            double[] razT = new double[3];
            Console.WriteLine("Отклонение R и R0:");
            for (int i = 0; i < 3; i++)
            {
                for (int j = 0; j < 3; j++)
                {
                    razR[i, j] = Math.Abs(R0[i, j] - R[i, j]);
                    Console.Write("{0,8}", Math.Round(razR[i, j], 4));
                }
                Console.WriteLine();
            }
            Console.WriteLine("Отклонение t и t0:");
            for (int i = 0; i < 3; i++)
            {
                razT[i] = Math.Abs(t0[i] - t[i]);
                Console.WriteLine("{0,8}", Math.Round(razT[i], 4));
            }
            Console.ReadKey();
        }
    }
}
```