Министерство образования Республики Беларусь
БЕЛОРУССКИЙ НАЦИОНАЛЬНЫЙ ТЕХНИЧЕСКИЙ УНИВЕРСИТЕТ

Кафедра «Робототехнические системы»

МАТЕМАТИЧЕСКОЕ ОБЕСПЕЧЕНИЕ ПРОМЫШЛЕННЫХ РОБОТОВ

Учебно-методический комплекс
для студентов специальности
1-53 01 06 «Промышленные роботы
и робототехнические комплексы»

В 2 частях

Часть 2

КУРСОВАЯ РАБОТА

Минск
БНТУ
2013

УДК 007.52:51(076.5)
ББК 32.816я7
М34

Авторы части:
А. Р. Околов, Е. Р. Новичихина, Г. С. Свидерский,
А. А. Шевченко, С. И. Шахнер

Рецензенты:
Н. Н. Гурский, С. Н. Павлович

Околов, А. Р.

М34 Математическое обеспечение промышленных роботов : учебно-методический комплекс для студентов специальности 1-53 01 06 «Промышленные роботы и робототехнические комплексы» : в 2 ч. / А. Р. Околов [и др.]. – Минск : БНТУ, 2012–2013. – Ч. 2 : Курсовая работа. – 2013. – 60 с.

ISBN 978-985-550-227-3 (Ч. 2).

Издание является второй частью учебно-методическо-го комплекса. Цель выполнения курсовой работы – повторение студентами лекционного материала и закрепление его на практике в виде курсовой работы. Рассмотрены основные вопросы, связанные с математическим описанием динамики и кинематики промышленного робота, планированием и моделированием траектории движения робота, а также решением прямой и обратной задач кинематики и динамики.

Учебно-методический комплекс адресуется студентам, инженерам и преподавателям, занимающимся проектированием и эксплуатацией промышленных роботов.

Часть 1 «Лабораторные работы» (авторы А. Р. Околов, Е. Р. Новичихина, Г. С. Свидерский) вышла в БНТУ в 2012 г.

УДК 007.52:51(076.5)
ББК 32.816я7

ISBN 978-985-550-227-3 (Ч. 2)
ISBN 978-985-550-093-4

1. ТЕМАТИКА КУРСОВОЙ РАБОТЫ

Целью курсовой работы (КР) является разработка программных средств, обеспечивающих выполнение задач управления движением манипулятора промышленного робота (ПР). К таким задачам относятся: планирование и оптимизация желаемой траектории движения манипулятора в различных системах (пространствах) координат, вычисление линейных и угловых скоростей движения рабочего органа манипулятора и скоростей его обобщенных координат с использованием прямой и обратной матриц Якоби и т.д.

Разработке программных средств, как правило, предшествует математическое решение поставленной задачи.

2. ЗАДАНИЕ ПО КУРСОВОЙ РАБОТЕ

Задание по КР выдается студенту преподавателем в начале семестра. Варианты заданий могут отличаться друг от друга как исходными данными, так и характером решаемых задач. Содержание заданий по курсовой работе приведено в разделе 7, варианты заданий – в Приложении 1.

3. СОДЕРЖАНИЕ КУРСОВОЙ РАБОТЫ

Расчетно-пояснительная записка к курсовой работе должна содержать:

– титульный лист;
– задание по курсовой работе;
– аннотацию;
– введение;
– формулировку задачи;
– математическое решение задачи;
– листинг (текст) программы;
– распечатку исходных данных и результатов выполнения программы;
– список использованных источников;
– приложение (если требуется).

4. ТЕХНИЧЕСКИЕ И ПРОГРАММНЫЕ СРЕДСТВА ДЛЯ ВЫПОЛНЕНИЯ РАБОТЫ

Курсовая работа выполняется на IBM-совместимом персональном компьютере (ПК) с использованием языка программирования ООП Visual C# 4.0. Допускается выполнение работы на других ПК и использование различных языков программирования.

5. ЗАЩИТА КУРСОВОЙ РАБОТЫ

При защите КР студент делает сообщение продолжительностью 5–10 минут, в котором показывает соответствие полученных результатов требованиям работы. При этом следует выделить основные этапы выполнения работы, отметить стандартные и оригинальные приемы решения поставленной задачи.

Вопросы, задаваемые студенту, могут касаться как содержания работы, так и различных разделов курса лекций по программному обеспечению робототехнических систем. При определении оценки за работу учитываются:

– правильность полученных результатов;
– обоснованность выбора и оригинальность решения;
– оформление расчетно-пояснительной записки.

6. ОБЩИЕ ТРЕБОВАНИЯ К ОФОРМЛЕНИЮ РАСЧЕТНО-ПОЯСНИТЕЛЬНОЙ ЗАПИСКИ

Расчетно-пояснительная записка должна выполняться в соответствии с требованиями, предъявляемыми к текстовым документам. Она должна быть написана от руки на одной стороне писчей бумаги формата А4 с полями шириной 20 мм для подшивки четким разборчивым почерком, напечатана на пишущей машинке или принтере. Материал должен быть изложен кратко, аргументированно и в логической последовательности.

Содержание записки разделяется на разделы и подразделы. Разделы записки нумеруются арабскими цифрами с точкой. Подразделы должны иметь сквозные порядковые номера в пределах каждого

раздела. Номера подразделов содержат две цифры: первая указывает номер раздела, вторая – номер подраздела, после каждой цифры ставятся точки. Содержание каждого раздела и подраздела можно разбивать на пункты, а пункты – на подпункты.

Каждый раздел рекомендуется начинать с нового листа; его наименование пишется прописными буквами. Переносы слов в заголовках не допускаются. Точку в конце заголовка не ставят. Не допускается подчеркивание заголовка.

Формулы, на которые в записке делается ссылка, должны иметь сквозную нумерацию. Номер формулы ставят с правой стороны листа в круглых скобках на уровне формулы. Ссылка в тексте на порядковый номер формулы дается в круглых скобках. Не следует нумеровать формулы, на которые в дальнейшем по тексту нет ссылки. Под формулой приводится значение каждого символа с новой строки в последовательности, в которой они приводятся в формуле. Первая строка расшифровки должна начинаться со слова «где» без двоеточия после него.

Иллюстрации (схемы, чертежи и т.п.) в записке именуются рисунками. Рисунки нумеруются в пределах раздела арабскими цифрами. Номер рисунка должен состоять из номера раздела и порядкового номера рисунка. Рисунки должны размещаться сразу после ссылки на них в тексте. Каждый рисунок должен сопровождаться содержательной подписью.

Цифровой или иной материал рекомендуется оформлять в виде таблиц. Каждая таблица должна иметь заголовок, который помещают под словом «Таблица». Это слово с номером таблицы пишется в правом верхнем углу таблицы. Номер таблицы состоит из номера раздела и порядкового номера таблицы.

В конце записки следует привести список использованной литературы. Литературные источники указывают в алфавитном порядке по фамилии авторов. При этом каждый источник должен иметь порядковый номер, фамилию и инициалы авторов, год издания, количество страниц. При ссылке на литературный источник в тексте записки в квадратных скобках указывается его номер в перечне литературы.

7. ЗАДАНИЯ ПО КУРСОВОЙ РАБОТЕ

Целью КР является моделирование движения манипулятора, приведенного на рис. П.1 Приложения 1, в различных системах координат с различными видами интерполяции траектории и его графическое отображение на дисплее ПК.

Варианты заданий приведены в табл. П.1 Приложения 1, исходные данные – в табл. П.2 Приложения 1, пример выполнения курсовой работы – в Приложении 2.

8. МЕТОДИКА ВЫПОЛНЕНИЯ КУРСОВОЙ РАБОТЫ

8.1. При планирования траектории движения манипулятора **в системе обобщенных координат** необходимо:

а) по заданным значениям углов Эйлера φ, θ, ψ и положениям схвата ($x_{\text{сх}}, y_{\text{сх}}$) в начальной, конечной и промежуточных точках составить матрицы однородных преобразований $P_{\text{схн}}$, $P_{\text{схк}}$, $P_{\text{схпр}}$, описывающих положение и ориентацию системы координат схвата относительно базовой системы координат;

б) для начального $P_{\text{схн}}$, конечного $P_{\text{схк}}$ и промежуточных $P_{\text{схпр}}$ положений схвата манипулятора определить соответствующие значения трех обобщенных координат i ($Q_{i\text{н}}$, $Q_{i\text{пр}}$, $Q_{i\text{к}}$), решив обратную задачу кинематики;

в) в соответствии с заданным видом интерполяции построить зависимости, характеризующие положения $Q_i(t)$, скорости $\dot{Q}_i(t)$ и ускорения $\ddot{Q}_i(t)$ каждой обобщенной координаты i в текущие моменты времени t, используя найденные ранее точки ($Q_{i\text{н}}$, $Q_{i\text{пр}}$, $Q_{i\text{к}}$) в качестве опорных (см. [6] лаб. раб. № 4, 7).

При построении графиков скорости и ускорения используйте метод приближенного дифференцирования графика положения, скорости;

г) написать программу визуализации движения манипулятора по сформированному закону движения.

Программа должна решать прямую задачу кинематики и определять текущие значения матриц однородного преобразования $P_i(t)$, описывающих положения звеньев *i* в текущие моменты времени *t* по сформированным значениям обобщенных координат $Q_i(t)$. Используя значения последних столбцов (размерностью 3×1) матриц однородных преобразований $P_i(t)$, строится проволочная модель робота. По матрицам вращения (размерностью 3×3) матриц однородных преобразований $P_i(t)$ строятся оси подвижной системы координат, связанной со схватом. В результате циклического выполнения программы на экране дисплея должна отображаться движущаяся проволочная модель манипулятора.

8.2. При планировании траектории движения манипулятора **в системе декартовых координат** необходимо:

а) по заданным значениям углов Эйлера φ, θ, ψ и положениям схвата ($x_{\text{сх}}$, $y_{\text{сх}}$) в начальной, конечной и промежуточных точках составить матрицы однородных преобразований $P_{\text{схн}}$, $P_{\text{схк}}$, $P_{\text{схпр}}$;

б) в соответствии с заданным видом интерполяции определить значения матриц однородных преобразований $P_{\text{сх}}(t)$ в текущие моменты времени *t*, используя начальное $P_{\text{схн}}$, конечное $P_{\text{схк}}$ и промежуточные $P_{\text{схпр}}$ положения рабочего органа в качестве опорных точек (см. [6] лаб. раб. № 5–7);

в) выполнить поточечное преобразование траектории в систему обобщенных координат манипулятора путем решения обратной задачи кинематики для текущих значений положений рабочего органа $P_{\text{сх}}(t)$ и построить зависимости, характеризующие положения $Q_i(t)$, скорости $\dot{Q}_i(t)$ и ускорения $\ddot{Q}_i(t)$ каждой обобщенной координаты *i* в текущие моменты времени *t*.

При построении графиков скорости и ускорения используйте метод приближенного дифференцирования графика положения;

г) написать программу визуализации движения манипулятора по сформированному закону движения.

Программа должна решать прямую задачу кинематики и определять текущие значения матриц однородного преобразования $P_i(t)$,

описывающих положения звеньев i в текущие моменты времени t по сформированным значениям обобщенных координат $Q_i(t)$. Используя значения последних столбцов (размерностью 3×1) матриц однородных преобразований $P_i(t)$, строится проволочная модель робота. По матрицам вращения (размерностью 3×3) матриц однородных преобразований $P_i(t)$ строятся оси подвижной системы координат, связанной со схватом. В результате циклического выполнения программы на экране дисплея должна отображаться движущаяся проволочная модель манипулятора.

ЛИТЕРАТУРА

1. Фу, К. Робототехника / К. Фу, Р. Гонсалес, К. Ли. – М.: Мир, 1989. – 624 с.

2. Шахинпур, М. Курс робототехники / М. Шахинпур. – М.: Мир, 1990. – 527 с.

3. Механика промышленных роботов / под ред. К. В. Фролова, Е. И. Воробьева. – Кн. 1: Кинематика и динамика. – М.: Высшая школа, 1988. – 304 с.

4. Корн, Г. Справочник по математике / Г. Корн, Т. Корн. – М.: Мир, 1974. – 832 с.

5. Шаньгин, Е. С. Управление роботами и робототехническими системами: конспект лекций / Е. С. Шаньгин. – Уфа, 2005.

6. Околов, А. Р. Математическое обеспечение промышленных роботов: учебно-методический комплекс для студентов специальности 1-53 01 06 «Промышленные роботы и робототехнические комплексы»: в 2 ч. / А. Р. Околов, Е. Р. Новичихина, Г. С. Свидерский. – Минск: БНТУ, 2012. – Ч. 1: Лабораторные работы. – 80 с.

ПРИЛОЖЕНИЯ

Приложение 1.
Варианты заданий курсовой работы

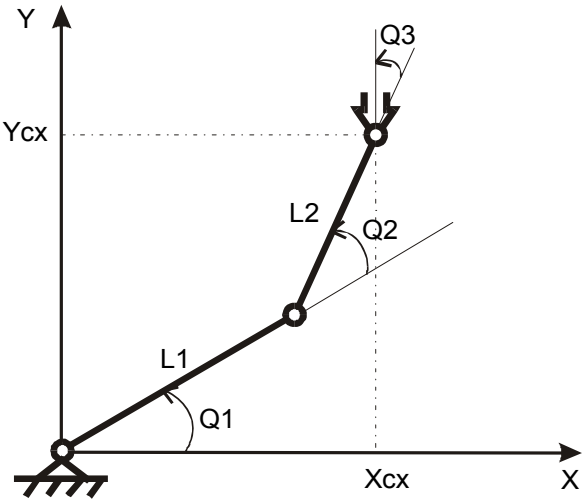

Рис. П.1. Трехстепенной манипулятор

Таблица П.1

Варианты заданий

Номер варианта	Система координат		Интерполяция					Исходные данные
	Обобщенная	Декартовая	Линейная	Круговая	Со сглаживанием отрезков	Лагранжа	Сплайн	
1	+	–	+	–	–	–	–	п/вар № 1
2	+	–	+	–	–	–	–	п/вар № 2
3	+	–	–	–	+	–	–	п/вар № 7
4	+	–	–	–	+	–	–	п/вар № 8
5	+	–	–	–	–	+	–	п/вар № 9
6	+	–	–	–	–	+	–	п/вар № 10
7	+	–	–	–	–	–	+	п/вар № 11
8	+	–	–	–	–	–	+	п/вар № 12
9	–	+	+	–	–	–	–	п/вар № 3
10	–	+	+	–	–	–	–	п/вар № 4
11	–	+	–	+	–	–	–	п/вар № 5
12	–	+	–	+	–	–	–	п/вар № 6
13	–	+	–	–	+	–	–	п/вар № 7
14	–	+	–	–	+	–	–	п/вар № 8
15	+	–	–	–	–	+	–	п/вар № 13
16	+	–	–	–	–	+	–	п/вар № 14
17	+	–	–	–	–	–	+	п/вар № 15
18	+	–	–	–	–	–	+	п/вар № 16

Таблица П.2

Исходные данные к разделу 7

№ подварианта	a	v	L_1	L_2	φ-const	θ-const	z-const
1	7000	80	5	3	0	0	0
2	6000	50	4	2	0	0	0
3	6500	70	6	3	0	0	0
4	800	80	6	3	0	0	0
5	7000	75	5	2	0	0	0
6	9500	85	4	2	0	0	0

№ подварианта	$\psi^{н}$	$x^{н}$	$y^{н}$	$\psi^{к}$	$X^{к}$	$Y^{к}$	$\psi^{пр}$	$X^{пр}$	$Y^{пр}$
1	0	3,3	5,5	–180	3,4	1,2	–	–	–
2	–180	–2,8	3,9	–90	–3	0,6	–	–	–
3	0	5	–5,6	90	4,3	–1,2	–	–	–
4	90	6	3,7	180	0,6	4,8	–	–	–
5	5	2,9	5,1	–25	5	3	–122	3,1	1
6	–122	2,9	–2	–65	4,5	–3,5	–80	2,9	–5

Продолжение табл. П.2

№ подварианта	L_1	L_2	t^0	t^1	t^2	t^3	φ-const	θ-const	z-const
7	5	4	0	1	2	3	0	0	0
8	5	3	0	1	2	4	0	0	0
9	4	2	0	1	2	3	0	0	0
10	4	3	0	1	2	4	0	0	0
11	6	3	0	1	2	3	0	0	0
12	6	3	0	1	2	4	0	0	0
13	6	3	0	1	2	3	0	0	0
14	6	3	0	1	2	4	0	0	0
15	4	2	0	1	2	3	0	0	0
16	4	3	0	1	2	4	0	0	0

№ подварианта	ψ^0	ψ^1	ψ^2	ψ^3	t^{acc}
7	–15	0	90	10	0.15
8	90	40	0	–90	0.1
9	–100	–60	–40	–80	–
10	180	170	121	90	–
11	0	20	60	90	–
12	90	150	210	270	–
13	0	20	60	90	–
14	90	150	210	270	–
15	–100	–60	–40	–80	–
16	180	170	121	90	–

Окончание табл. П.2

№ подварианта	x^0	x^1	x^2	x^3	y^0	y^1	y^2	y^3	v^0	v^1	v^2	v^3
7	–0,5	–1,8	–1	–4,7	2,2	4,9	3,3	5,3	–	–	–	–
8	3,3	3,8	4,1	3,4	5,5	3,5	2,1	1,2	–	–	–	–
9	–2,8	–3,4	–3,3	–3	3,9	3,2	1,4	0,6	–	–	–	–
10	–3,7	–4,6	4,0	–2,4	–3,9	–3,1	–0,8	–0,1	–	–	–	–
11	5	5,8	5,4	4,3	–5,6	–4,2	–2	–1,2	0	30	50	0
12	6	4,6	1,9	0,6	3,7	5,6	5,7	4,8	0	20	40	0
13	5	5,8	5,4	4,3	–5,6	–4,2	–2	–1,2	–	–	–	–
14	6	4,6	1,9	0,6	3,7	5,6	5,7	4,8	–	–	–	–
15	–2,8	–3,4	–3,3	–3	3,9	3,2	1,4	0,6	0	30	50	0
16	–3,7	–4,6	4,0	–2,4	–3,9	–3,1	–0,8	–0,1	0	20	40	0

Приложение 2.
Пример выполнения курсовой работы

Министерство образования Республики Беларусь

БЕЛОРУССКИЙ НАЦИОНАЛЬНЫЙ ТЕХНИЧЕСКИЙ УНИВЕРСИТЕТ

Кафедра «Робототехнические системы»

ПОЯСНИТЕЛЬНАЯ ЗАПИСКА

к курсовой работе по дисциплине:
«Математическое обеспечение промышленных роботов»
Планирование траекторий ПР

(наименование темы)

Исполнитель ***Иванов И.И.***

Руководитель ***Петров П.П.***

Минск 2013

СОДЕРЖАНИЕ

АННОТАЦИЯ

В данной работе отображены вопросы планирования траектории для плоскостного трехзвенного манипулятора, в частности планирование траектории в обобщенных координатах методом интерполяции полиномом Лагранжа. Результатом выполнения работы является построение графиков положения, скорости и ускорения звеньев манипулятора в обобщенной системе координат и визуализация движения робота в декартовой системе координат.

ВВЕДЕНИЕ

Опираясь на законы кинематики и динамики манипулятора, можно так управлять приводами сочленений, чтобы манипулятор двигался вдоль некоторой траектории, обеспечивающей выполнение поставленной задачи. Перед началом движения манипулятора важно знать: во-первых, существуют на его пути какие-то препятствия, и, во-вторых, накладываются ли какие-то ограничения на характер его траектории.

Кривую, вдоль которой схват манипулятора движется из начальной точки в конечную, называют его траекторией. Задача состоит в разработке математического аппарата для выбора и описания желаемого движения манипулятора между начальной и конечной точками траектории. Как правило, траектория, соединяющая начальное и конечное положение схвата, не единственна. Возможно, например, движение схвата вдоль прямой, соединяющей начальную и конечную точки (прямолинейная траектория), а также вдоль некоторой гладкой кривой, удовлетворяющей ряду ограничений на положение и ориентацию схвата на начальном, конечном и промежуточных участках траектории (сглаженная траектория). При этом не смотря на то, что виды траектории будут различными между опорными точками, тем не менее в опорных точках положения будут одинаковыми, равными заданным значениям.

Кроме того, что могут различаться виды траекторий, способы получения траекторий также могут быть различными. Различают способы планирования в декартовых координатах (ДК) и обобщенных координатах (ОК).

Первый состоит в том, что исследователь задает точный набор ограничений для звеньев (заданные значения положения, скорости, ускорения и непрерывность их) в некоторых узловых точках траектории. Планировщик траектории выбирает функцию, проходящую через узловые точки и удовлетворяющую заданным ограничениям, таким образом формирую все промежуточные положения звеньев. После этого решается прямая задача кинематики(ПЗК), т.е. по полученным значениям положений сочленений (звеньев) определяется положение схвата относительно базовой системы координат во всех точках траектории в ДК.

Второй подход состоит в том, что исследователь задает желаемую траекторию манипулятора в виде некоторой аналитически описываемой функции в декартовых координатах. Планировщик после этого решает обратную задачу кинематики (ОЗК), т.е. по заданному положение схвата относительно базовой системы координат определяются положения сочленений (звеньев) в ОК, используемые для управления приводами движения.

Известно, что отработка траектории манипулятора производится в пространстве обобщенных координат. Поэтому, если планирование траектории осуществляется в декартовых координатах, то для ее отработки необходимо решать обратную задачу кинематики о положении манипулятора. Если такое преобразование необходимо выполнять в реальном масштабе времени, то оно является серьезной вычислительной нагрузкой для системы управления. Кроме того законы движения звеньев манипулятора будут далеки от идеальных, возможны обрывы и скачки траектории в ОК. Преимуществом планирования траектории в пространстве декартовых координат является наглядность и предсказуемость траектории движения схвата манипулятора, так как непосредственно планируется траектория, отрабатываемая схватом.

Преимуществом планирования траектории в пространстве обобщенных координат является:

– планирование непосредственно траектории, которую должны отрабатывать приводы звеньев манипулятора;

– планирование траектории требует небольшого количества вычислений;

– траекторию легче планировать, так как отсутствует понятие ориентации.

Недостатком планирования в ОК является то, что итоговая траектория, получаемая после решения ПЗК, является непредсказуемой и может сильно отличатся от желаемой.

Сейчас в микропроцессорных системах управления промышленных роботов наиболее часто применяются линейная и круговая интерполяция в декартовой системе координат и линейная интерполяция в пространстве обобщенных координат. При линейной интерполяции в обобщенных координатах манипулятор совершает движение за минимально необходимое время за счет минимально необходимых перемещений звеньев. Закон движения звеньев тра-

пецеидальный для скорости. При линейной интерполяции в декартовых координатах схват манипулятора движется по прямой линии, ориентация его также меняется по линейному закону. В последнее время все большее применение находит сплайн-интерполяция. Сущность ее заключается в представлении траектории между узловыми точками кривой, описываемой степенными полиномами третьей, четвертой или пятой степени. При этом коэффициенты полинома выбираются так, чтобы обеспечить отработку ограничений, накладываемых на траекторию.

В общем случае выбор типа движения определяется решаемой технологической задачей. Если требуется выполнить дуговую сварку, окраску или нанесение покрытия, то необходимо отрабатывать траекторию в декартовых координатах. В этом случае траектория движения манипулятора имеет первостепенное значение, хотя некоторые звенья манипулятора совершают большие перемещения, чем необходимо для перемещения в конечную точку, и могут возникнуть излишние ускорения (торможения) звеньев. При этом скорость движения самого схвата должна поддерживаться постоянной, игнорируя ограничения на скорость и ускорение в шарнирах манипулятора. С другой стороны, если манипулирование объектами осуществляется в упорядоченной детерминированной среде, когда отрабатываются только опорные точки и не важна траектория между ними, планирование в декартовых координатах можно не использовать, поскольку удобнее применять интерполяцию в обобщенных координатах, как более быструю и удобную для приводов.

1. ПОСТАНОВКА ЗАДАЧИ

Целью задания является моделирование движения манипулятора, приведенного на рисунке 1.1 в обобщенной системе координат (СК) методом интерполяции полиномом Лагранжа траектории и графическое отображение его движения на дисплее ПК.

Вариант задания приведен в таблице 1.1, исходные данные – в таблице 1.2.

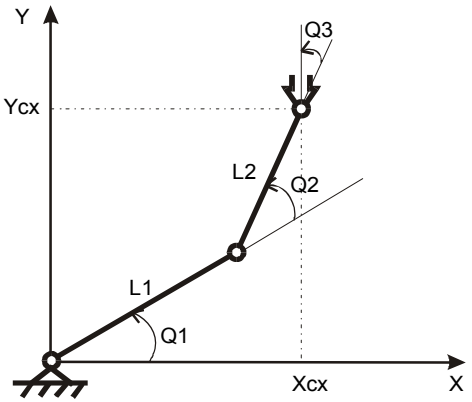

Рисунок 1.1. Кинематическая схема манипулятора

Таблица 1.1. Вариант задания

Система координат		**Вид интерполяции**				
обобщенная	декартовая	линейная	круговая	со сглаживанием отрезков	Лагранжа	сплайн
+	–	–	–	–	+	–

Таблица 1.2. Исходные данные

t	$t_0 = 0$ с	$t_1 = 1$ с	$t_2 = 2$ с	$t_3 = 4$ с
x	–3,7	–4,6	4,0	–2,4
y	–3,9	–3,1	–0,8	–0,1
v	0	20	40	0
ψ	180	170	121	90
θ	0	0	0	0
φ	0	0	0	0

$$L1 = 4, L2 = 3,$$

где t – время прохождения промежуточных опорных точек;

x, y – положение схвата в опорные моменты времени;

v – скорость сочленений (звеньев) в опорные моменты времени;

ψ, θ, φ – углы Эйлера, задающие ориентацию схвата.

2. РЕШЕНИЕ ЗАДАЧИ В ОБЩЕМ ВИДЕ

При планировании траектории движения манипулятора в системе обобщенных координат необходимо:

а) по заданным значениям углов Эйлера φ, θ, ψ и положениям схвата ($x_{\text{сх}}, y_{\text{сх}}$) в начальной, конечной и промежуточных точках составить матрицы однородных преобразований $P_{\text{схн}}$, $P_{\text{схк}}$, $P_{\text{схпр}}$, описывающих положение и ориентацию системы координат схвата относительно базовой системы координат;

б) для начального $P_{\text{схн}}$, конечного $P_{\text{схк}}$ и промежуточных $P_{\text{схпр}}$ положений схвата манипулятора определить соответствующие значения трех обобщенных координат i ($Q_{i\text{н}}$, $Q_{i\text{пр}}$, $Q_{i\text{к}}$), решив обратную задачу кинематики;

в) в соответствии с заданным видом интерполяции построить зависимости, характеризующие положения $Q_i(t)$, скорости $\dot{Q}_i(t)$ и ускорения $\ddot{Q}_i(t)$ каждой обобщенной координаты i в текущие моменты времени t, используя найденные ранее точки ($Q_{i\text{н}}$, $Q_{i\text{пр}}$, $Q_{i\text{к}}$) в качестве опорных.

При построении графиков скорости и ускорения используйте метод приближенного дифференцирования графика положения, скорости;

г) написать программу визуализации движения манипулятора по сформированному закону движения.

Программа должна решать прямую задачу кинематики и определять текущие значения матриц однородного преобразования $P_i(t)$, описывающих положения звеньев i в текущие моменты времени t по сформированным значениям обобщенных координат $Q_i(t)$. Используя значения последних столбцов (размерностью 3×1) матриц однородных преобразований $P_i(t)$, строится проволочная модель робота. По матрицам вращения (размерностью 3×3) матриц однородных преобразований $P_i(t)$ строятся оси подвижной системы координат, связанной со схватом. В результате циклического выполнения программы на экране дисплея должна отображаться движущаяся проволочная модель манипулятора.

3. МАТЕМАТИЧЕСКОЕ РЕШЕНИЕ ЗАДАЧИ

Первый этап решения – составление матриц однородных преобразований для начальной, конечной и промежуточных точек.

Матрица однородного преобразования, описывающая положение и ориентацию схвата в базовой системе координат $Q_i(t)$, имеет вид:

$$P=\begin{bmatrix} \cos\varphi\cos\theta\cos\psi-\sin\varphi\sin\psi & -\cos\varphi\cos\theta\sin\psi-\sin\varphi\cos\psi & \cos\varphi\sin\theta & x \\ \sin\varphi\cos\theta\cos\psi+\cos\varphi\sin\psi & -\sin\varphi\cos\theta\sin\psi+\cos\varphi\cos\psi & \sin\varphi\sin\theta & y \\ -\sin\theta\cos\psi & \sin\theta\sin\psi & \cos\theta & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix}. \quad (2.1)$$

Подставляя значения углов Эйлера и координат схвата в формулу (2.1), получаем матрицы однородных преобразований для начальной, конечной и промежуточных точек траектории:

$$P_0=\begin{bmatrix} 1 & 0 & 0 & -3,7 \\ 0 & 1 & -1,225 & -3,9 \\ 0 & 1,225 & -1 & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix},$$

$$P_1=\begin{bmatrix} 1 & 0 & 0 & -4,6 \\ 0 & 0,985 & -0,174 & -3,1 \\ 0 & 0,174 & -0,985 & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix},$$

$$P_2=\begin{bmatrix} 1 & 0 & 0 & 4 \\ 0 & 0,515 & -0,857 & -0,8 \\ 0 & 0,857 & -0,515 & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix},$$

$$P_3 = \begin{bmatrix} 1 & 0 & 0 & -2,4 \\ 0 & -6,123 & -1 & -0,1 \\ 0 & 1 & 6,123 & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix}.$$

Второй этап – решение обратной задачи кинематики для каждого звена в начальной, конечной и промежуточных точках. Решим ее с точки зрения алгебраического подхода.

Добавляем к манипулятору еще одну степень подвижности (поворот схвата на угол Q_3). Кинематические параметры полученного трехзвенного манипулятора приведены в таблице 2.1.

Таблица 2.1. Кинематические параметры манипулятора

I	***α***	***a***	***d***	***Q***
1	0	ℓ_1	0	Q_1
2	0	ℓ_2	0	Q_2
3	0	0	0	Q_3

Матрица, описывающая положение третьего звена (схвата) относительно базовой системы координат, выглядит следующим образом:

$$\overset{b}{w}T = \overset{0}{3}T = \begin{bmatrix} c123 & -s123 & 0 & \ell_1 c1 + \ell_2 c12 \\ s123 & c123 & 0 & \ell_1 s1 + \ell_2 s12 \\ 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix}, \qquad (2.2)$$

где $c123 = \cos(Q_1 + Q_2 + Q_3)$;
$s123 = \sin(Q_1 + Q_2 + Q_3)$;
$c1 = \cos(Q_1)$;
$c12 = \cos(Q_1 + Q_2)$;
$s1 = \sin(Q_1)$;

$s12 = \sin(Q_1 + Q_2)$.

Для рассматриваемого манипулятора матрицу ${}^0_3T$ можно записать в виде:

$$ {}^0_3T = \begin{bmatrix} n_x & o_x & a_x & p_x \\ n_y & o_y & a_y & p_y \\ n_z & o_z & a_z & p_z \\ 0 & 0 & 0 & 1 \end{bmatrix} = \begin{bmatrix} c\psi & -s\psi & 0 & x \\ s\psi & c\psi & 0 & y \\ 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix}, \qquad (2.3) $$

где $c\psi = \cos\psi$;

$s\psi = \sin\psi$.

Приравняем выражения (2.2) и (2.3) и получим четыре нелинейных уравнения, которые необходимо решить относительно Q_1, Q_2, Q_3.

$$ \begin{cases} c\psi = c123, \\ s\psi = s123, \\ x = \ell_1 c1 + \ell_2 c12, \\ y = \ell_1 s1 + \ell_2 s12. \end{cases} \qquad (2.4) $$

Возведя в квадрат, левые и правые части двух последних уравнений истемы (2.4) и складывая их, получим:

$$ (x^2 + y^2) = \ell_1^2 (c1^2 + s1^2) + \ell_2^2 (c12^2 + s12^2) + $$
$$ +2\ell_1\ell_2 (c1c12 + s1s12) = \ell_1^2 + \ell_2^2 + 2\ell_1\ell_2 c2, \qquad (2.5) $$

где $c2 = cosQ_2$.

Решая уравнение (2.5) относительно $c2$, получим

$$ c2 = \frac{x^2 + y^2 - \ell_1^2 - \ell_2^2}{2\ell_1\ell_2}. \qquad (2.6) $$

Для существования решения правая часть уравнения (2.6) должна лежать в пределах [–1; +1]. Если условие выполнено, то находим

$$\sin Q_2 = s2 = \pm\sqrt{1 - c2^2}\ . \qquad (2.7)$$

Выбор знака у $\sin Q_2$ соответствует одному из возможных решений. Принимаем положение манипулятора «локоть вниз», для которого $\sin Q_2$ в данном случае принимает положительные значения. При определении угла Q_2 воспользуемся функцией *ATAN*2, которая обеспечивает выбор всех решений и выбор квадранта. Отсюда

$$Q_2 = ATAN2(s2, c2). \qquad (2.8)$$

Зная Q_2, воспользуемся двумя последними уравнениями системы (2.4) для определения угла Q_1. Перепишем их в следующем виде:

$$\begin{cases} x = k_1 c1 - k_2 s1, \\ y = k_1 s1 + k_2 c1, \end{cases} \qquad (2.9)$$

где $k_1 = \ell_1 + \ell_2 c2$;

$k_2 = \ell_2 s2$.

Для решения уравнений системы (2.9) выполним замену переменных:

$$r = \sqrt{k_1^2 + k_2^2}\ ;\ \ \gamma = ATAN2(k_2, k_1),$$

тогда

$$k_1 = r\cos\gamma\ ; \qquad k_2 = r\sin\gamma\ .$$

Теперь систему уравнений (2.9) можно записать следующим образом:

$$\begin{cases} \dfrac{x}{r} = \cos\gamma\cos Q_1 - \sin\gamma\sin Q_1, \\ \dfrac{y}{r} = \cos\gamma\sin Q_1 + \sin\gamma\cos Q_1. \end{cases} \tag{2.10}$$

Откуда находим

$$\begin{cases} \dfrac{x}{r} = \cos(\gamma + Q_1), \\ \dfrac{y}{r} = \sin(\gamma + Q_2). \end{cases} \tag{2.11}$$

Используя функцию *ATAN2*, имеем

$$\begin{gathered} \gamma + Q_1 = ATAN2(y, x), \\ Q_1 = ATAN2(y, x) - ATAN2(k_2, k_1). \end{gathered} \tag{2.12}$$

Из первых двух уравнений системы (2.4) находим

$$Q_3 = \psi - Q_2 - Q_1. \tag{2.13}$$

Подставляя исходные данные в выражения (2.6)–(2.13) и проведя соответствующий расчет, находим значения обобщенных координат для начальной, конечной и промежуточных точек:

$$\begin{array}{llll} Q_1^0 = 13,01^\circ & Q_1^1 = 2,31^\circ & Q_1^2 = -54,89^\circ & Q_1^3 = -45,97^\circ \\ Q_2^0 = 80,65^\circ; & Q_2^1 = 76,01^\circ; & Q_2^2 = 110,39^\circ; & Q_2^3 = 143,25^\circ. \\ Q_3^0 = 86,25^\circ & Q_3^1 = 91,6^\circ & Q_3^2 = 65,5^\circ & Q_3^3 = -7,28^\circ \end{array}$$

Третий этап решения – планирование траектории движения манипулятора в обобщенных координатах с учетом интерполяции полиномом Лагранжа, сущность которой заключается в том, что теоретически можно провести непрерывную и гладкую кривую точно через последовательность узловых точек, используя формулу , интерполяции Лагранжа:

$$y(x) = \frac{(x-x_1)(x-x_2)...(x-x_n)}{(x_0-x_1)(x_0-x_2)...(x_0-x_n)} y_0 +$$
$$+ \frac{(x-x_0)(x-x_2)...(x-x_n)}{(x_1-x_0)(x_1-x_2)...(x_1-x_n)} y_1 + ...$$
$$... + \frac{(x-x_0)(x-x_1)...(x-x_{n-1})}{(x_n-x_0)(x_n-x_1)...(x_n-x_{n-1})} y_n,$$

где x_n – значения аргумента в опорных точках траектории;

y_n – значения функции в этих точках.

В этом случае мы получаем высокую плавность траектории, без каких либо скачков скорости и ускорения (рис. 2.1), однако при этом методе степень полинома растет пропорционально числу точек, что увеличивает время расчетов и колебательность.

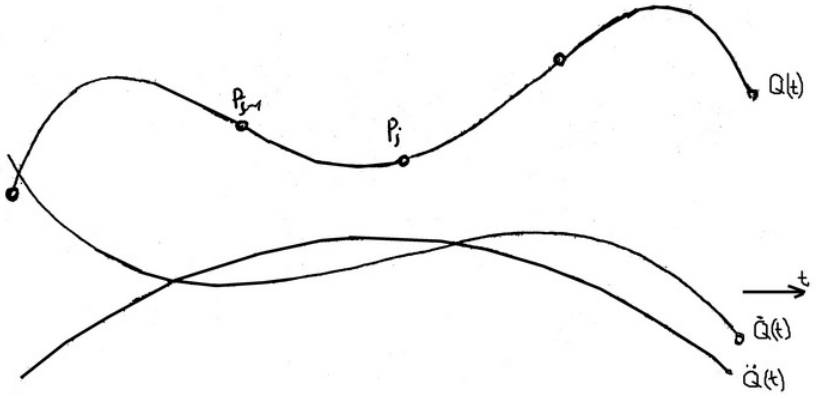

Рисунок 2.1. Вид интерполяции по формуле Лагранжа

Кроме того, желаемую траекторию между узловыми точками очень трудно предугадать. По вышеперечисленным причинам метод интерполяции по формуле Лагранжа не нашел практического применения.

Четвертый этап решения – поточечное преобразование спланированной в обощенной системе координат траектории в систему декартовых координат путем решения прямой задачи кинематики. Прямую задачу будем решать геометрическим методом.

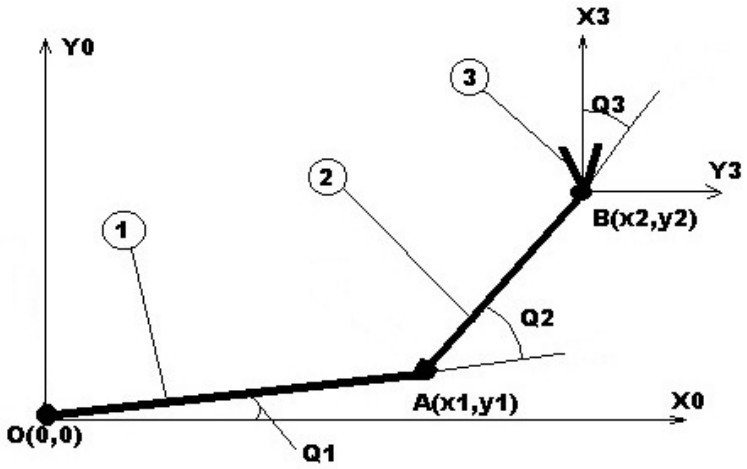

Рисунок 2.2. Решение ПЗК для манипулятора:
1 – первое звено; *2* – второе звено; *3* – третье звено (схват)

Декартовы координаты точек A(x_1, y_1) и B(x_2, y_2) относительно центра базовой системы координат O(0, 0) вычисляются с помощью геометрических зависимостей:

$$x_1 = l_1 cos Q_1; \qquad x_2 = x_1 + l_2 \cos(Q_1 + Q_2);$$
$$y_1 = l_1 sin Q_1; \qquad y_2 = y_1 + l_2 \sin(Q_1 + Q_2).$$

Таким образом, определили все необходимые для написания программы и моделирования движения манипулятора формулы.

4. МОДЕЛИРУЮЩАЯ ПРОГРАММА

4.1 Описание программы

Моделирующая программа написана на языке ООП Visual C# 4.0 [5] и выполняет следующие операции:

– решение обратной задачи кинематики для заданных контрольных точек (начальной, конечной и промежуточных);

– планирование сглаженной траектории в обобщенных координатах методом интерполяции полиномом Лагранжа;

– решение прямой задачи кинематики в каждой точке геометрическим методом;

– визуализацию движения трехзвенного манипулятора;

– построение графиков зависимостей обобщенных координат q_1, q_2, q_3, а также их скоростей и ускорений от времени.

Программа построена по блочно-модульному типу с использованием многократно вызываемых методов и событий. Моделирующая программа включает в себя следующие основные методы и события:

– *TimeIntervals* – метод создает массив значений времени, в соответствии со временем прохождения каждого участка (всего 80 элементов);

– *Result4x4* – метод, по заданным значениям углов Эйлера φ, θ, ψ и положениям схвата ($x_{\text{сх}}, y_{\text{сх}}$) в начальной, конечной и промежуточных точках составляются матрицы однородных преобразований;

– *OZK* – метод для решения обратной задачи кинематики, с точки зрения алгебраического подхода, для каждого звена в начальной, конечной и промежуточных точках;

– *Lagrange* – метод для реализации интерполяции с помощью формулы Лагранжа;

– *LagrangeSpeed* – метод для реализации интерполяции с помощью формулы Лагранжа (нахождения скорости – первая производная полинома по времени);

– *LagrangeAcceleration* – метод для реализации интерполяции с помощью формулы Лагранжа (нахождения ускорения – вторая производная полинома по времени);

– *PZK* – метод для решения прямой задачи кинематики в каждой точке геометрическим методом;

– *DrawDimension* –графический метод, производит разметку осей графиков с обобщенными координатами в соответствии с масштабом графиков;

– *DrawManipulator* – графический метод осуществляет визуализацию движения трехзвенного манипулятора в декартовых координатах;

– *DrawPosition* – графический метод осуществляет построение графиков положения всех звеньев в обобщенных координатах;

– *DrawSpeed* – графический метод осуществляет построение графиков скорости всех звеньев в обобщенных координатах;

– *DrawAcceleration* – графический метод осуществляет построение графиков ускорений всех звеньев в обобщенных координатах;

– *buttonDecart_Click* – обработчик события Click, запускает метод *DrawManipulator*;

– *buttonObobs_Click* – обработчик события Click, запускает метод *DrawPosition*;

– *buttonSkor_Click* – обработчик события Click, запускает метод *DrawSpeed*;

– *buttonUskor_Click* – обработчик события Click, запускает метод *DrawAcceleration*;

Для хранения в памяти значений обобщенных координат, их скоростей и ускорений используются массивы типа double[,]. Размер массива определен заранее, исходя из шага квантования $dt = 0{,}05$ с и общего времени по заданию $T = 4$ с (всего 80 элементов).

Построение графических объектов (манипулятора, графиков зависимостей) осуществляется в режиме 600×500 точек.

4.2. Текст моделирующей программы

```
using System;
using System.Collections.Generic;
using System.ComponentModel;
using System.Data;
using System.Drawing;
using System.Drawing.Drawing2D;
using System.Linq;
using System.Text;
using System.Windows.Forms;

namespace курсачМОПР
{
   public partial class Form1 : Form
   {

      public Form1()
      {
         InitializeComponent();
      }

      private void Form1_Load(object sender, EventArgs e)
      {

      }

      //Массив для координат X, Y, взятых из условия
      double[,] realXY = new double[2, 4] {
      { -3.7, -4.6, 4.0, -2.4 },
      { -3.9, -3.1, -0.8, -0.1 }
      };

      //Длины суставов манипулятора
      double l1 = 4.0;
      double l2 = 3.0;

      int p1 = 20;
```

```
int p2 = 20;
int p3 = 40;

//Углы psi для разных t
double[] psi = { 180.0, 170.0, 121.0, 90.0 };

double[] time;

//Массив времени t
double[] t = { 0, 1.0, 2.0, 4.0 };

int nx, ny;

#region Методы

//Разбить все время выполнения на одинаковые интервалы
public void TimeIntervals()
{
   time = new double[p1 + p2 + p3];
   time[0] = t[0];
   time[p1 - 1] = t[1];
   time[p1 + p2 - 1] = t[2];
   time[p1 + p2 + p3 - 1] = t[3];

   for (int i = 1; i < p1 - 1; i++)
   {
      time[i] = time[i - 1] + (t[1] - t[0]) / p1;
   }
   for (int i = p1; i < p1 + p2 - 1; i++)
   {
      time[i] = time[i - 1] + (t[2] - t[1]) / p2;
   }
   for (int i = p1 + p2; i < p1 + p2 + p3 - 1; i++)
   {
      time[i] = time[i - 1] + (t[3] - t[2]) / p3;
   }
}
```

```
        //Создание матрицы однородных преобразований
        public double[,] Result4x4(double psi, double x, double y)
        {
            int fi = 0;
            int eta = 0;
            psi = psi * Math.PI / 180;

            double[,] arrresult = new double[4, 4];
            arrresult[0, 0] = Math.Cos(fi) * Math.Cos(eta);
            arrresult[0, 1] = Math.Cos(fi) * Math.Sin(eta) * Math.Sin(psi) -
Math.Sin(fi) * Math.Cos(psi);
            arrresult[0, 2] = Math.Cos(fi) * Math.Sin(eta) * Math.Cos(psi) +
Math.Sin(fi) * Math.Sin(psi);
            arrresult[1, 0] = Math.Sin(fi) * Math.Cos(eta);
            arrresult[1, 1] = Math.Sin(fi) * Math.Sin(eta) * Math.Sin(psi) -
Math.Cos(fi) * Math.Cos(psi);
            arrresult[1, 2] = Math.Sin(fi) * Math.Sin(eta) * Math.Cos(psi) -
Math.Cos(fi) * Math.Sin(psi);
            arrresult[2, 0] = -Math.Sin(eta);
            arrresult[2, 1] = Math.Cos(eta) * Math.Sin(psi);
            arrresult[2, 2] = Math.Cos(eta) * Math.Cos(psi);

            arrresult[0, 3] = x;
            arrresult[1, 3] = y;
            arrresult[2, 3] = 0;
            arrresult[3, 0] = 0;
            arrresult[3, 1] = 0;
            arrresult[3, 2] = 0;
            arrresult[3, 3] = 1;

            return arrresult;
        }

        //Обратная задача кинематики(найти углы Q1, Q2, Q3 в граду-
сах)
        public double[,] OZK()
        {
```

```
            double c2, s2;
            double[,] qr = new double[3, realXY.GetLength(1)];

            for (int i = 0; i < realXY.GetLength(1); i++)
            {

                c2 = (realXY[0, i] * realXY[0, i] + realXY[1, i] * realXY[1, i]
- l1 * l1 - l2 * l2) / (2 * l1 * l2);
                s2 = Math.Sqrt(1 - c2 * c2);
                if (c2 > 0) qr[1, i] = 180 / Math.PI * Math.Atan(s2 / c2);
                if (c2 < 0) qr[1, i] = 180 + 180 / Math.PI * Math.Atan(s2 / c2);
                qr[0, i] = 180 / Math.PI * Math.Atan(realXY[1, i] / realXY[0,
i]) - 180 / (Math.PI) * Math.Atan((l2 * s2) / (l1 + l2 * c2));
                qr[2, i] = psi[i] - qr[0, i] - qr[1, i];

            }
            return qr;
        }

        //Интерполяция полиномом Лагранжа
        public double[,] Lagrange(double[,] arr)
        {
            double[,] lagQR = new double[3, p1 + p2 + p3];

            TimeIntervals();

            //Присвоить координаты Y
            for (int j = 0; j < 3; j++)
            {
                for (int i = 0; i < p1 + p2 + p3; i++)
                {

                    lagQR[j, i] = ((time[i] - t[1]) * (time[i] - t[2]) * (time[i] -
t[3])) /
                        ((t[0] - t[1]) * (t[0] - t[2]) * (t[0] - t[3])) * arr[j, 0] +

                        ((time[i] - t[0]) * (time[i] - t[2]) * (time[i] - t[3])) /
                        ((t[1] - t[0]) * (t[1] - t[2]) * (t[1] - t[3])) * arr[j, 1] +
```

```
                ((time[i] - t[0]) * (time[i] - t[1]) * (time[i] - t[3])) /
                ((t[2] - t[0]) * (t[2] - t[1]) * (t[2] - t[3])) * arr[j, 2] +

                ((time[i] - t[0]) * (time[i] - t[1]) * (time[i] - t[2])) /
                ((t[3] - t[0]) * (t[3] - t[1]) * (t[3] - t[2])) * arr[j, 3];

            }
        }
        return lagQR;
    }

    //Нахождение координат скорости
    public double[,] LagrangeSpeed(double[,] arr)
    {

        double[,] lagQR = new double[3, p1 + p2 + p3];

        TimeIntervals();

        //Присвоить координаты Y
        for (int j = 0; j < 3; j++)
        {
            for (int i = 0; i < p1 + p2 + p3; i++)
            {

                lagQR[j, i] = (3 * time[i] * time[i] - 2 * (t[3] + t[2] +
t[1])*time[i] + t[2]*t[3] + t[1]*t[3] + t[2]*t[1]) /
                ((t[0] - t[1]) * (t[0] - t[2]) * (t[0] - t[3])) * arr[j, 0] +

                (3 * time[i] * time[i] - 2 * (t[3] + t[2] + t[0]) * time[i] +
t[2] * t[3] + t[0] * t[3] + t[2] * t[0]) /
                ((t[1] - t[0]) * (t[1] - t[2]) * (t[1] - t[3])) * arr[j, 1] +

                (3 * time[i] * time[i] - 2 * (t[3] + t[0] + t[1]) * time[i] +
t[1] * t[3] + t[0] * t[3] + t[0] * t[1]) /
                ((t[2] - t[0]) * (t[2] - t[1]) * (t[2] - t[3])) * arr[j, 2] +
```

```
                (3 * time[i] * time[i] - 2 * (t[0] + t[2] + t[1]) * time[i] +
t[2] * t[1] + t[1] * t[0] + t[2] * t[0]) /
                ((t[3] - t[0]) * (t[3] - t[1]) * (t[3] - t[2])) * arr[j, 3];

            }
        }
        return lagQR;
    }

    //Нахождение координат ускорения
    public double[,] LagrangeAcceleration(double[,] arr)
    {

        double[,] lagQR = new double[3, p1 + p2 + p3];

        TimeIntervals();

        //Присвоить координаты Y
        for (int j = 0; j < 3; j++)
        {
            for (int i = 0; i < p1 + p2 + p3; i++)
            {

                lagQR[j, i] = (6 * time[i] - 2 * (t[3] + t[2] + t[1])) /
                    ((t[0] - t[1]) * (t[0] - t[2]) * (t[0] - t[3])) * arr[j, 0] +

                    (6 * time[i] - 2 * (t[3] + t[2] + t[0])) /
                    ((t[1] - t[0]) * (t[1] - t[2]) * (t[1] - t[3])) * arr[j, 1] +

                    (6 * time[i] - 2 * (t[3] + t[0] + t[1])) /
                    ((t[2] - t[0]) * (t[2] - t[1]) * (t[2] - t[3])) * arr[j, 2] +

                    (6 * time[i] - 2 * (t[0] + t[2] + t[1])) /
                    ((t[3] - t[0]) * (t[3] - t[1]) * (t[3] - t[2])) * arr[j, 3];

            }
        }
        return lagQR;
```

```
    }

    //Прямая задача кинематики(найти координаты точек манипу-
лятора)
    public double[,] PZK(double[,] arr)
    {
        double[,] pzkXY = new double[8, p1 + p2 + p3];

        //Перевод градусов в радианы
        for (int i = 0; i < 3; i++)
        {
            for (int j = 0; j < p1+p2+p3; j++)
            {
                arr[i, j] = arr[i, j] * Math.PI / 180;
            }
        }

        for (int i = 0; i < p1 + p2 + p3; i++)
        {
            //Координаты точки 1
            pzkXY[0, i] = l1 * Math.Cos(arr[0, i]);
            pzkXY[1, i] = l1 * Math.Sin(arr[0, i]);
            //Координаты точки 2
            pzkXY[2, i] = l1 * Math.Cos(arr[0, i]) + l2 * Math.Cos(arr[0,
i] + arr[1, i]);
            pzkXY[3, i] = l1 * Math.Sin(arr[0, i]) + l2 * Math.Sin(arr[0, i]
+ arr[1, i]);
            //Нужны для построения оси Y точки 2
            pzkXY[4, i] = l1 * Math.Cos(arr[0, i]) + l2 * Math.Cos(arr[0,
i] + arr[1, i]) + Math.Sin(arr[2, i]);
            pzkXY[5, i] = l1 * Math.Sin(arr[0, i]) + l2 * Math.Sin(arr[0, i]
+ arr[1, i]) + Math.Cos(arr[2, i]);
            //Нужны для построения оси X точки 2
            pzkXY[6, i] = l1 * Math.Cos(arr[0, i]) + l2 * Math.Cos(arr[0,
i] + arr[1, i]) + Math.Sin(arr[2, i] - 90 * Math.PI / 180);
            pzkXY[7, i] = l1 * Math.Sin(arr[0, i]) + l2 * Math.Sin(arr[0, i]
+ arr[1, i]) + Math.Cos(arr[2, i] - 90 * Math.PI / 180);
        }
```

```
            return pzkXY;
        }

        //Примеры вызова PZK
        //double[,] mass = PZK(Lagrange(OZK()));

        #endregion

        #region Рисование

        public void DrawDimension(Graphics g, Pen blackpen, SolidBrush
brush)
        {
            Font font = new Font("Arial", 9);
            string x, y;

            int ii = 0;
            int j = 19;

            for(int k = 0; k < 4; k++) {
                g.DrawLine(blackpen, new Point(ii + 25 + nx, 270), new
Point(ii + 25 + nx, 280));

                x = time[j].ToString();
                g.DrawString(x, font, brush, ii + 25 + nx - 5, 290.0F);

                ii = ii + nx;
                j = j + 20;
            }

            ii = 275;
            for (int k = 0; k < 6; k++)
            {
              g.DrawLine(blackpen, new Point(20, ii - ny), new Point(30, ii - ny));

                y = (k + 1).ToString();
                g.DrawString(y, font, brush, 5, ii - ny - 5);
```

```
            ii = ii - ny;
        }

        ii = 275;
        for (int k = 0; k < 6; k++)
        {
            g.DrawLine(blackpen, new Point(20, ii + ny), new Point(30, ii
+ ny));

            y = (- k - 1).ToString();
            g.DrawString(y, font, brush, 5, ii + ny - 5);

            ii = ii + ny;
        }

        x = "0";
        g.DrawString(x, font, brush, 5, 270.0F);

        font.Dispose();
    }

    private void DrawManipulator(object sender, EventArgs e)
    {
        nx = 40;
        ny = 40;

        double[,] promD = PZK(Lagrange(OZK()));

        int[,] prom = new int[promD.GetLength(0),
promD.GetLength(1)];
        for (int i = 0; i < promD.GetLength(1); i++) {
            prom[0, i] = (int)(promD[0, i] * nx) + 300;
            prom[1, i] = 275 - (int)(promD[1, i] * ny);

            prom[2, i] = (int)(promD[2, i] * nx) + 300;
            prom[3, i] = 275 - (int)(promD[3, i] * ny);

            prom[4, i] = (int)(promD[4, i] * nx) + 300;
```

```
            prom[5, i] = 275 - (int)(promD[5, i] * ny);

            prom[6, i] = (int)(promD[6, i] * nx) + 300;
            prom[7, i] = 275 - (int)(promD[7, i] * ny);
        }

        using (Graphics g = this.CreateGraphics())
        {
            Pen blackpen = new Pen(Color.Black, 5);
            Pen bluepen = new Pen(Color.Blue, 1);
            Pen bp = new Pen(Color.Black, 1);
            Pen red = new Pen(Color.Red, 2);
            Font font = new Font("Arial", 14);
            Font f = new Font("Arial", 8);
            SolidBrush brush = new SolidBrush(Color.Black);

            string osx = "X";
            string osy = "Y";
            string op;


            for (int i = 0; i < p1+p2+p3; i++)
            {
                g.Clear(Color.White);

                //Начертить звенья
                g.DrawLine(blackpen, new Point(300, 275), new
Point(prom[0, i], prom[1, i]));
                g.DrawLine(blackpen, new Point(prom[0, i], prom[1, i]),
new Point(prom[2, i], prom[3, i]));
                g.DrawLine(bluepen, new Point(prom[2, i], prom[3, i]), new
Point(prom[4, i], prom[5, i]));
                g.DrawLine(bluepen, new Point(prom[2, i], prom[3, i]), new
Point(prom[6, i], prom[7, i]));

                //Начертить оси
                g.DrawLine(bp, new Point(300, 25), new Point(300, 475));
                g.DrawLine(bp, new Point(25, 275), new Point(575, 275));
```

```
            //Стрелки
            g.DrawLine(bp, new Point(300, 25), new Point(288, 50));
            g.DrawLine(bp, new Point(300, 25), new Point(312, 50));
            g.DrawLine(bp, new Point(575, 275), new Point(550, 263));
            g.DrawLine(bp, new Point(575, 275), new Point(550, 287));
            //Подпись
            g.DrawString(osx, font, brush, 580.0F, 275.0F);
            g.DrawString(osy, font, brush, 305.0F, 10.0F);

            //Подписать доп. оси
            g.DrawString(osx, f, brush, prom[4, i] + 5, prom[5, i] - 5);
            g.DrawString(osy, f, brush, prom[6, i] + 5, prom[7, i] - 5);

            //Обозначить пересечения звеньев
            g.DrawEllipse(blackpen, prom[2, i], prom[3, i], 5.0F, 5.0F);
            g.DrawEllipse(blackpen, 300, 275, 5.0F, 5.0F);
            g.DrawEllipse(blackpen, prom[0, i], prom[1, i], 5.0F, 5.0F);

            //Нарисовать схват
            g.DrawLine(blackpen, new Point(prom[2, i], prom[3, i]),
new Point(prom[2, i] - 10, prom[3, i] - 10));
            g.DrawLine(blackpen, new Point(prom[2, i], prom[3, i]),
new Point(prom[2, i] + 10, prom[3, i] - 10));

            //Подписать оси X
            for (int ii = 300 + nx; ii <= 575; ) {
               g.DrawLine(bp, new Point(ii, 270), new Point(ii, 280));
               op = ((ii - 300)/nx).ToString();
               g.DrawString(op, f, brush, ii - 5, 290);
               ii += nx;
            }
            for (int ii = 300 - nx; ii >= 25; )
            {
               g.DrawLine(bp, new Point(ii, 270), new Point(ii, 280));
               op = ((ii - 300) / nx).ToString();
               g.DrawString(op, f, brush, ii - 5, 290);
               ii -= nx;
            }
```

```
            //Подписать оси Y
            for (int ii = 275 - ny; ii >= 25; )
            {
               g.DrawLine(bp, new Point(295, ii), new Point(305, ii));
               op = (Math.Abs(ii - 275) / ny).ToString();
               g.DrawString(op, f, brush, 280, ii - 5);
               ii -= ny;
            }
            for (int ii = 275 + ny; ii <= 475; )
            {
               g.DrawLine(bp, new Point(295, ii), new Point(305, ii));
               op = ((- ii + 275) / ny).ToString();
               g.DrawString(op, f, brush, 280, ii - 5);
               ii += ny;
            }
            op = "0";
            g.DrawString(op, f, brush, 280, 260);

            System.Threading.Thread.Sleep(30);
         }

         //Траектория движения и опорные точки
         for (int i = 0; i < p1 + p2 + p3; i++ )
         {
            g.DrawEllipse(red, prom[2, i], prom[3, i], 2.0F, 2.0F);
         }
         g.DrawEllipse(blackpen, prom[2, 0], prom[3, 0], 6.0F, 6.0F);
         g.DrawEllipse(blackpen, prom[2, p1 - 1], prom[3, p1 - 1],
6.0F, 6.0F);
         g.DrawEllipse(blackpen, prom[2, p1 + p2 - 1], prom[3, p1 + p2
- 1], 6.0F, 6.0F);
         g.DrawEllipse(blackpen, prom[2, p1 + p2 + p3 - 1], prom[3, p1
+ p2 + p3 - 1], 6.0F, 6.0F);

         red.Dispose();
         bp.Dispose();
         blackpen.Dispose();
         bluepen.Dispose();
```

```
            font.Dispose();
            brush.Dispose();
        }
    }

    private void DrawPosition(object sender, EventArgs e)
    {
        nx = 135;
        ny = 70;

        double[,] promD = Lagrange(OZK());

        int[,] prom = new int[promD.GetLength(0) * 2,
promD.GetLength(1)];
        for (int i = 0; i < promD.GetLength(1); i++)
        {
            prom[0, i] = 25 + (int)(time[i] * nx);
            prom[1, i] = 275 - (int)(promD[0, i] * (Math.PI / 180) * ny);

            prom[2, i] = 25 + (int)(time[i] * nx);
            prom[3, i] = 275 - (int)(promD[1, i] * (Math.PI / 180) * ny);

            prom[4, i] = 25 + (int)(time[i] * nx);
            prom[5, i] = 275 - (int)(promD[2, i] * (Math.PI / 180) * ny);
        }

        using (Graphics g = this.CreateGraphics()) {

            Pen blackpen = new Pen(Color.Black, 1);
            Pen redpen = new Pen(Color.Red, 2);
            Pen bluepen = new Pen(Color.Blue, 2);
            Pen greenpen = new Pen(Color.Green, 2);
            Font font = new Font("Arial", 14);
            SolidBrush brush = new SolidBrush(Color.Black);

            string osx = "t,c";
            string osy = "Q(t),рад";
            string one = "Звено 1";
```

```
            string two = "Звено 2";
            string three = "Звено 3";

            g.Clear(Color.White);

            g.DrawLine(blackpen, new Point(25, 25), new Point(25, 475));
           g.DrawLine(blackpen, new Point(25, 275), new Point(575, 275));

            g.DrawLine(blackpen, new Point(25, 25), new Point(13, 50));
            g.DrawLine(blackpen, new Point(25, 25), new Point(37, 50));
           g.DrawLine(blackpen, new Point(575, 275), new Point(550, 263));
           g.DrawLine(blackpen, new Point(575, 275), new Point(550, 287));

            g.DrawString(osx, font, brush, 580.0F, 275.0F);
            g.DrawString(osy, font, brush, 15.0F, 0.0F);


            for (int i = 0; i < promD.GetLength(1) - 1; i++) {
               g.DrawLine(redpen, new Point(prom[0, i], prom[1, i]), new
Point(prom[0, i + 1], prom[1, i + 1]));
            }
            for (int i = 0; i < promD.GetLength(1) - 1; i++)
            {
               g.DrawLine(bluepen, new Point(prom[2, i], prom[3, i]), new
Point(prom[2, i + 1], prom[3, i + 1]));
            }
            for (int i = 0; i < promD.GetLength(1) - 1; i++)
            {
               g.DrawLine(greenpen, new Point(prom[4, i], prom[5, i]),
new Point(prom[4, i + 1], prom[5, i + 1]));
            }

            g.DrawString(one, font, brush, 95.0F, 10.0F);
            g.DrawLine(redpen, new Point(175, 20), new Point(205, 20));

            g.DrawString(two, font, brush, 245.0F, 10.0F);
            g.DrawLine(bluepen, new Point(325, 20), new Point(355, 20));
```

```
            g.DrawString(three, font, brush, 395.0F, 10.0F);
            g.DrawLine(greenpen, new Point(475, 20), new Point(505, 20));

            DrawDimension(g, blackpen, brush);

            redpen.Dispose();
            blackpen.Dispose();
            bluepen.Dispose();
            greenpen.Dispose();
            font.Dispose();
            brush.Dispose();
        }
    }

    private void DrawSpeed(object sender, EventArgs e) {

        nx = 135;
        ny = 100;

        TimeIntervals();

        double[,] promD = LagrangeSpeed(OZK());
        int[,] prom = new int[promD.GetLength(0) * 2,
promD.GetLength(1)];
        for (int i = 0; i < promD.GetLength(1); i++)
        {
            prom[0, i] = 25 + (int)(time[i] * nx);
            prom[1, i] = 275 - (int)(promD[0, i] * (Math.PI / 180) * ny);

            prom[2, i] = 25 + (int)(time[i] * nx);
            prom[3, i] = 275 - (int)(promD[1, i] * (Math.PI / 180) * ny);

            prom[4, i] = 25 + (int)(time[i] * nx);
            prom[5, i] = 275 - (int)(promD[2, i] * (Math.PI / 180) * ny);
        }

        using (Graphics g = this.CreateGraphics())
        {
```

```
            Pen blackpen = new Pen(Color.Black, 1);
            Pen redpen = new Pen(Color.Red, 2);
            Pen bluepen = new Pen(Color.Blue, 2);
            Pen greenpen = new Pen(Color.Green, 2);
            Font font = new Font("Arial", 14);
            SolidBrush brush = new SolidBrush(Color.Black);

            string osx = "t,c";
            string osy = "Q'(t),рад";
            string one = "Звено 1";
            string two = "Звено 2";
            string three = "Звено 3";

            g.Clear(Color.White);

            g.DrawLine(blackpen, new Point(25, 25), new Point(25, 475));
           g.DrawLine(blackpen, new Point(25, 275), new Point(575, 275));

            g.DrawLine(blackpen, new Point(25, 25), new Point(13, 50));
            g.DrawLine(blackpen, new Point(25, 25), new Point(37, 50));
           g.DrawLine(blackpen, new Point(575, 275), new Point(550, 263));
           g.DrawLine(blackpen, new Point(575, 275), new Point(550, 287));

            g.DrawString(osx, font, brush, 580.0F, 275.0F);
            g.DrawString(osy, font, brush, 15.0F, 0.0F);


            for (int i = 0; i < promD.GetLength(1) - 1; i++)
            {
               g.DrawLine(redpen, new Point(prom[0, i], prom[1, i]), new
Point(prom[0, i + 1], prom[1, i + 1]));
            }
            for (int i = 0; i < promD.GetLength(1) - 1; i++)
            {
               g.DrawLine(bluepen, new Point(prom[2, i], prom[3, i]), new
Point(prom[2, i + 1], prom[3, i + 1]));
            }
            for (int i = 0; i < promD.GetLength(1) - 1; i++)
```

```
            {
                g.DrawLine(greenpen, new Point(prom[4, i], prom[5, i]),
new Point(prom[4, i + 1], prom[5, i + 1]));
            }

            g.DrawString(one, font, brush, 95.0F, 10.0F);
            g.DrawLine(redpen, new Point(175, 20), new Point(205, 20));

            g.DrawString(two, font, brush, 245.0F, 10.0F);
            g.DrawLine(bluepen, new Point(325, 20), new Point(355, 20));

            g.DrawString(three, font, brush, 395.0F, 10.0F);
           g.DrawLine(greenpen, new Point(475, 20), new Point(505, 20));

            DrawDimension(g, blackpen, brush);

            redpen.Dispose();
            blackpen.Dispose();
            bluepen.Dispose();
            greenpen.Dispose();
            font.Dispose();
            brush.Dispose();
        }

    }

    private void DrawAcceleration(object sender, EventArgs e)
    {

        nx = 135;
        ny = 70;

        TimeIntervals();

        double[,] promD = LagrangeAcceleration(OZK());
        int[,] prom = new int[promD.GetLength(0) * 2,
promD.GetLength(1)];
        for (int i = 0; i < promD.GetLength(1); i++)
```

```
{
    prom[0, i] = 25 + (int)(time[i] * nx);
    prom[1, i] = 275 - (int)(promD[0, i] * (Math.PI / 180) * ny);

    prom[2, i] = 25 + (int)(time[i] * nx);
    prom[3, i] = 275 - (int)(promD[1, i] * (Math.PI / 180) * ny);

    prom[4, i] = 25 + (int)(time[i] * nx);
    prom[5, i] = 275 - (int)(promD[2, i] * (Math.PI / 180) * ny);
}

using (Graphics g = this.CreateGraphics())
{

    Pen blackpen = new Pen(Color.Black, 1);
    Pen redpen = new Pen(Color.Red, 2);
    Pen bluepen = new Pen(Color.Blue, 2);
    Pen greenpen = new Pen(Color.Green, 2);
    Font font = new Font("Arial", 14);
    SolidBrush brush = new SolidBrush(Color.Black);

    string osx = "t,c";
    string osy = "Q"(t),рад";
    string one = "Звено 1";
    string two = "Звено 2";
    string three = "Звено 3";

    g.Clear(Color.White);

    g.DrawLine(blackpen, new Point(25, 25), new Point(25, 475));
   g.DrawLine(blackpen, new Point(25, 275), new Point(575, 275));

    g.DrawLine(blackpen, new Point(25, 25), new Point(13, 50));
    g.DrawLine(blackpen, new Point(25, 25), new Point(37, 50));
   g.DrawLine(blackpen, new Point(575, 275), new Point(550, 263));
   g.DrawLine(blackpen, new Point(575, 275), new Point(550, 287));

    g.DrawString(osx, font, brush, 580.0F, 275.0F);
```

```
            g.DrawString(osy, font, brush, 15.0F, 0.0F);


            for (int i = 0; i < promD.GetLength(1) - 1; i++)
            {
                g.DrawLine(redpen, new Point(prom[0, i], prom[1, i]), new
Point(prom[0, i + 1], prom[1, i + 1]));
            }
            for (int i = 0; i < promD.GetLength(1) - 1; i++)
            {
                g.DrawLine(bluepen, new Point(prom[2, i], prom[3, i]), new
Point(prom[2, i + 1], prom[3, i + 1]));
            }
            for (int i = 0; i < promD.GetLength(1) - 1; i++)
            {
                g.DrawLine(greenpen, new Point(prom[4, i], prom[5, i]),
new Point(prom[4, i + 1], prom[5, i + 1]));
            }

            g.DrawString(one, font, brush, 95.0F, 10.0F);
            g.DrawLine(redpen, new Point(175, 20), new Point(205, 20));

            g.DrawString(two, font, brush, 245.0F, 10.0F);
            g.DrawLine(bluepen, new Point(325, 20), new Point(355, 20));

            g.DrawString(three, font, brush, 395.0F, 10.0F);
            g.DrawLine(greenpen, new Point(475, 20), new Point(505, 20));

            DrawDimension(g, blackpen, brush);

            redpen.Dispose();
            blackpen.Dispose();
            bluepen.Dispose();
            greenpen.Dispose();
            font.Dispose();
            brush.Dispose();
        }
    }
```

```
        #endregion

        #region Кнопки

        private void buttonDecart_Click(object sender, EventArgs e)
        {
            DrawManipulator(sender, e);
        }

        private void buttonObobs_Click(object sender, EventArgs e)
        {
            DrawPosition(sender, e);
        }

        private void buttonSkor_Click(object sender, EventArgs e)
        {
            DrawSpeed(sender, e);
        }

        private void buttonUskor_Click(object sender, EventArgs e)
        {
            DrawAcceleration(sender, e);
        }

        #endregion
    }
}
```

5. РЕЗУЛЬТАТЫ ВЫПОЛНЕНИЯ ПРОГРАММЫ

При запуске программы курсовая_работа.exe (программа написана на языке ООП C# в среде Microsoft Visual Studio 2010) появляется диалоговое окно. В представленнем меню мы выбираем раздел с нужными нам сведениями.

Результаты выполнения программы представлены на рис. 5.1–5.5. На рис. 5.1 показан экран программы после планирования траектории с изображенным манипулятором, траекторией движения, промежуточными значениями и исходными данными.

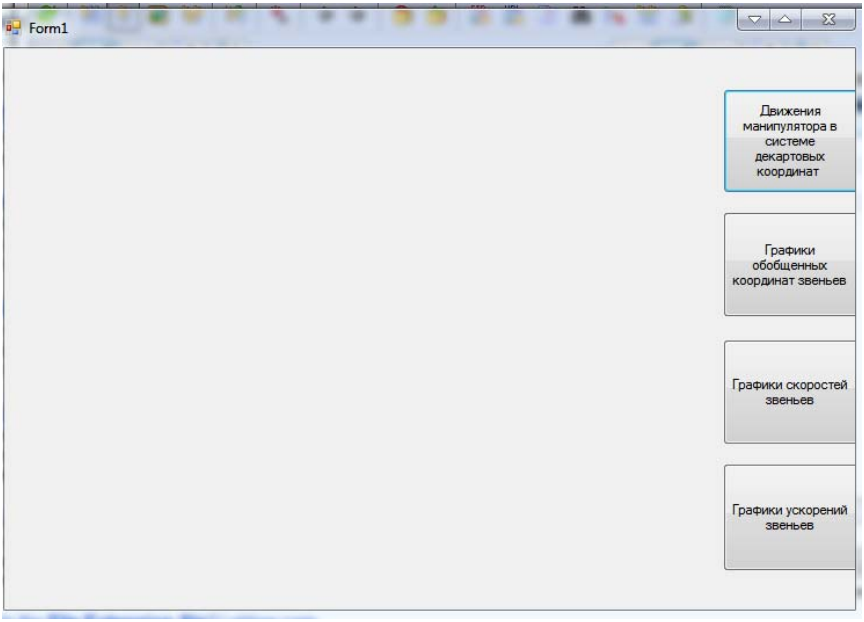

Рис. 5.1. Диалоговое окно

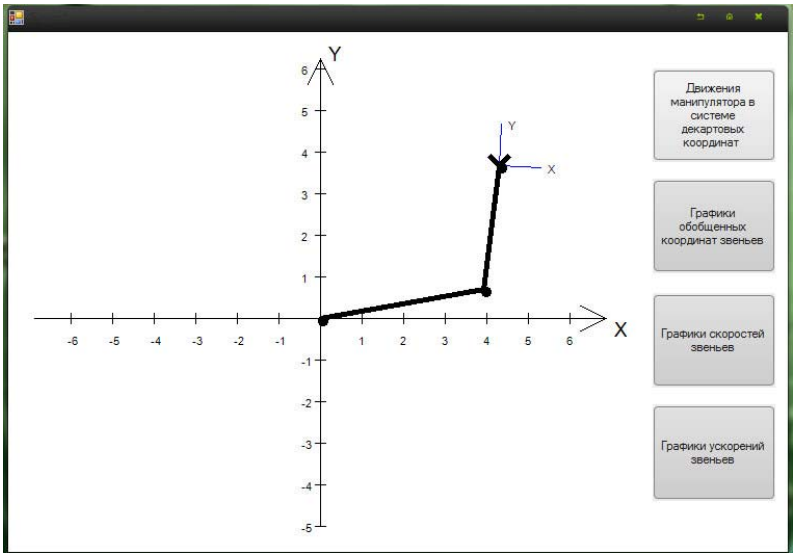

Рис. 5.2. Визуализация движения манипулятора
в системе декартовых координат (начальное положение)

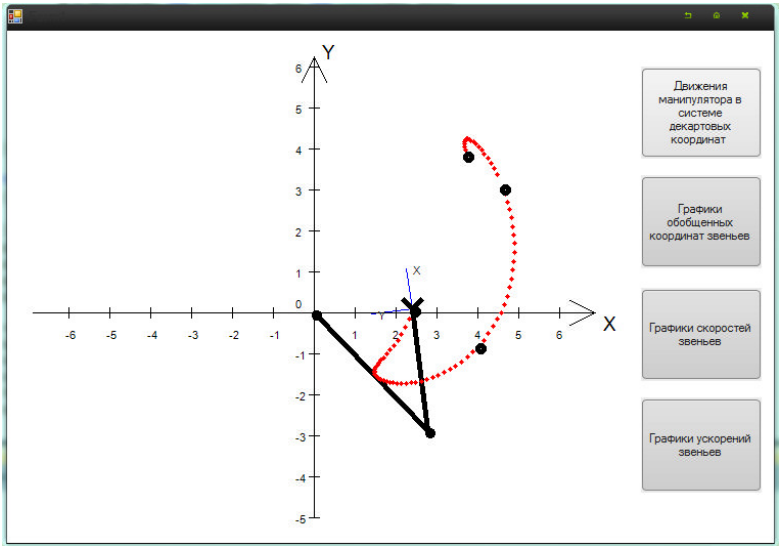

Рис. 5.3. Визуализация движения манипулятора
в системе декартовых координат (конечное положение)

Рис. 5.4. Графики обобщенных координат звеньев

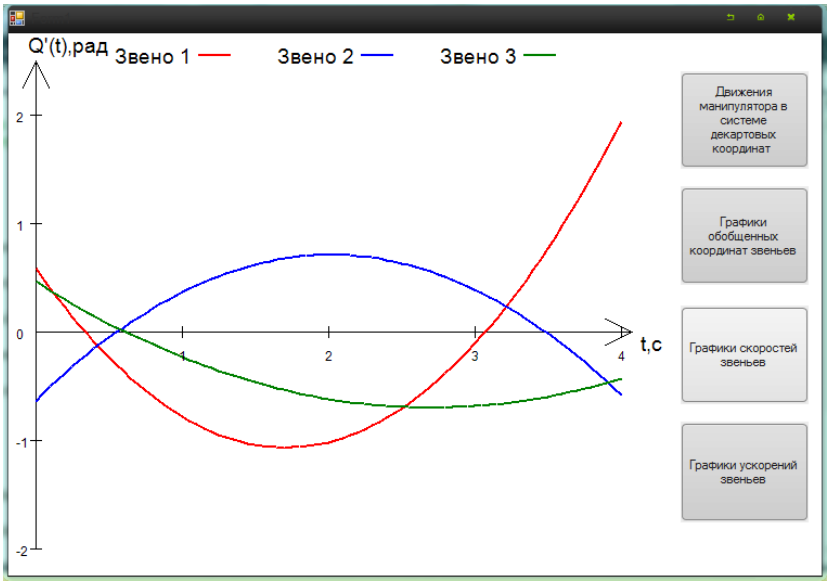

Рис. 5.5. Графики скоростей звеньев

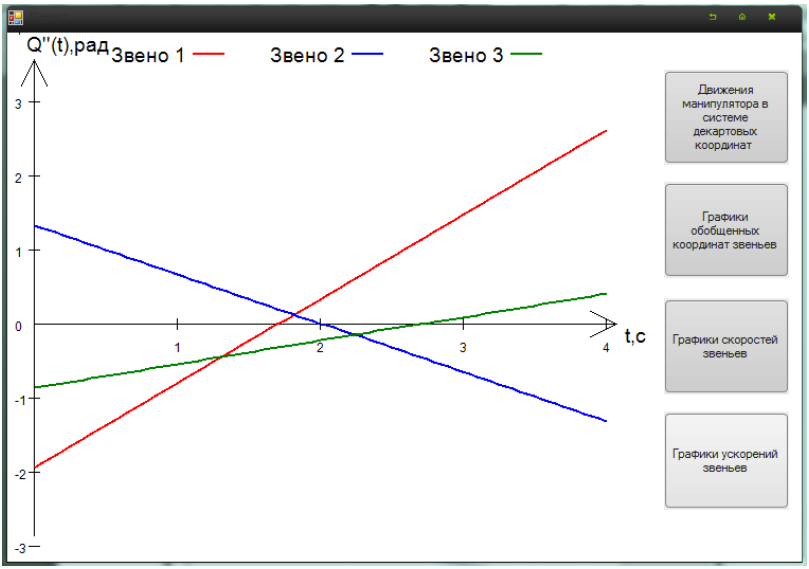

Рис. 5.6. Графики ускорений звеньев

ЛИТЕРАТУРА

1. Фу, К. Робототехника / К. Фу, Р. Гонсалес, К. Ли. – М.: Мир, 1989. – 624 с.

2. Шахинпур, М. Курс робототехники / М. Шахинпур. – М.: Мир, 1990. – 527 с.

3. Механика промышленных роботов / под ред. К. В. Фролова, Е. И. Воробьева. – Кн. 1: Кинематика и динамика. – М.: Высшая школа, 1988. – 304 с.

4. Корн, Г. Справочник по математике / Г. Корн, Т. Корн. – М.: Мир, 1974. – 832 с.

5. Шилдт, Г. C# 4.0: полное руководство: пер. с англ. / Г. Шилдт. – М.: Вильямс, 2011. – 1056 с. – Парал. тит. англ.

6. Околов, А. Р. Математическое обеспечение промышленных роботов: учебно-методический комплекс для студентов специальности 1-53 01 06 «Промышленные роботы и робототехнические комплексы»: в 2 ч. / А. Р. Околов, Е. Р. Новичихина, Г. С. Свидерский. – Минск: БНТУ, 2012. – Ч. 1: Лабораторные работы. – 80 с.

ОГЛАВЛЕНИЕ

Учебное издание

ОКОЛОВ Андрей Ромуальдович
НОВИЧИХИНА Елена Романовна
СВИДЕРСКИЙ Геннадий Сигизмундович и др.

МАТЕМАТИЧЕСКОЕ ОБЕСПЕЧЕНИЕ ПРОМЫШЛЕННЫХ РОБОТОВ

Учебно-методический комплекс
для студентов специальности
1-53 01 06 «Промышленные роботы
и робототехнические комплексы»

В 2 частях

Часть 2

КУРСОВАЯ РАБОТА

Технический редактор *Д. А. Исаев*
Компьютерная верстка *Д. А. Исаева*

Подписано в печать 17.07.2013. Формат 60×84 $^{1}/_{16}$. Бумага офсетная. Ризография.
Усл. печ. л. 3,49. Уч.-изд. л. 2,73. Тираж 100. Заказ 510.

Издатель и полиграфическое исполнение: Белорусский национальный технический университет. ЛИ № 02330/0494349 от 16.03.2009. Пр. Независимости, 65. 220013, г. Минск.