МИНИСТЕРСТВО ОБРАЗОВАНИЯ РОССИЙСКОЙ ФЕДЕРАЦИИ

МОСКОВСКИЙ ЭНЕРГЕТИЧЕСКИЙ ИНСТИТУТ
(ТЕХНИЧЕСКИЙ УНИВЕРСИТЕТ)

А.В. КОРЕЦКИЙ, Н.В.ОСАДЧЕНКО

КОМПЬЮТЕРНОЕ МОДЕЛИРОВАНИЕ КИНЕМАТИКИ МАНИПУЛЯЦИОННЫХ РОБОТОВ

МЕТОДИЧЕСКОЕ ПОСОБИЕ

ПО КУРСУ

"Вычислительная механика"

для студентов, обучающихся по специальности
"Роботы и робототехнические системы"

Москва Издательство МЭИ 2000

УДК
621.398
К 664
УДК: 621.398.–506.29.001.57(072)

Утверждено учебным управлением МЭИ

Подготовлено на кафедре теоретической механики

Корецкий А.В., **Осадченко Н.В.**

Компьютерное моделирование кинематики манипуляционных роботов. – М.: Издательство МЭИ, 2000. – 48 с.

Рассматриваются вопросы моделирования кинематики манипуляционных роботов с использованием компьютера. Приводится ряд сведений из теории винтов, формализм которой играет в дальнейшем изложении роль основного инструмента при кинематическом анализе манипуляторов. Даются рекомендации по выполнению заданий лабораторного практикума по кинематике манипуляторов. Излагается порядок работы с типовой программой **`kin_31.c`** и программным комплексом **`UM`**.

Для студентов, обучающихся по направлению “Прикладная механика” и специальности “Роботы и робототехнические системы”.

Введение

В предлагаемом методическом пособии рассматриваются задачи кинематического анализа манипуляционных роботов. Исполнительные механизмы манипуляционных роботов – это пространственные механизмы, характеризуемые большим числом степеней свободы и описываемые достаточно сложными уравнениями. Решение таких задач обычно связано с проведением компьютерного моделирования.

В настоящее время компьютерное моделирование систем твёрдых тел с большим числом степеней свободы оформилось в самостоятельный раздел вычислительной механики, переживающий процесс бурного развития и привлекающий внимание многочисленных исследовательских коллективов. За последние двадцать лет опубликовано немало монографий и статей по вопросам автоматизированного формирования уравнений движения многозвенных механических систем и методам их численного исследования, так что характерный для данной области механики теоретический и алгоритмический аппарат достаточно развит. В то же время остаётся актуальной проблема отбора приёмов решения типовых задач, в разумной мере сочетающих универсальность, простоту использования и вычислительную эффективность.

Рассматриваемые в пособии задачи включены в лабораторный практикум по курсу вычислительной механики, читаемому в МЭИ для студентов, специализирующихся в области робототехники. Разбор данных задач предваряется обзором основных сведений из теории винтов, формализм которой играет в последующем изложении роль основного инструмента.

Применение метода винтов позволило разработать достаточно простой и эффективный алгоритм кинематического анализа шестизвенного манипулятора. Данный алгоритм был реализован в виде типовой программы **`kin_31.c`**, написанной на языке Си и рассчитанной на персональные компьютеры класса **PC IBM**. На основе этой типовой программы студенты могут создавать собственные программы для решения задач практикума.

При решении рассматриваемых задач предусматривается также использование программного комплекса *“Универсальный механизм”* (**UM**). Комплекс может применяться как в учебных, так и в научных целях. Он позволяет эффективно решать задачи кинематического и динамического анализа для широкого класса пространственных механизмов и, в частности, робототехнических систем. В пособии изложен порядок работы с этим комплексом при решении задач кинематики роботов-манипуляторов.

Глава 1. Метод винтов и его применение для кинематического анализа манипуляционных роботов

1.1. О моделировании в вычислительной механике

Компьютерное моделирование систем твёрдых тел – перспективное направление в исследовании пространственных механизмов.

Поскольку с ростом сложности проектируемых систем их аналитическое исследование становится всё более затруднительным, а создание опытных образцов обходится всё дороже, то моделирование механических систем на компьютере зачастую оказывается ведущим (а то и единственным) доступным методом исследования.

Процесс моделирования включает в себя два этапа: создание *математической модели* системы и её исследование (с использованием, в частности, разнообразных численных методов). Модель должна одновременно удовлетворять двум требованиям: правильно отражать основные особенности моделируемой системы и быть достаточно простой для эффективного применения методов численного анализа.

Понятие "математическая модель" в вычислительной механике по своей сути двойственно. Для человека-исследователя математической моделью реальной системы служит [4] некоторая *математическая структура* (т.е. знаковая система, представляющая некое множество объектов с заданной системой отношений между ними), причём объекты этой структуры трактуются как идеализированные реальные вещи, а отношения между объектами – как конкретные связи между элементами действительности. Для компьютера математическая модель реальной системы выступает как совокупность *данных*, размещённых в его памяти и содержащих всю информацию об этой системе, существенную для процесса моделирования.

В связи с этим проблема разработки математических моделей для того или иного класса реальных систем включает два аспекта, которые можно назвать абстрактно-математическим и информационным. С одной стороны, итогом построения модели должна быть некоторая знаковая система, а с другой – некоторая структура данных. В целом же в задачах вычислительной механики можно выделить три *уровня моделирования* [5]: инвариантный, координатный и программный. Первые два из них отражают абстрактно-математический аспект моделирования, а третий связан с информационным аспектом.

На инвариантном уровне модель представляется в наиболее компактном и удобном для восприятия виде, не загромождённом вносимой извне дополнительной структурой. В задачах вычислительной механики при этом используют, прежде всего, аппарат линейной алгебры (включая такие понятия, как векторы, линейные и аффинные отображения). Координатный уровень позволяет ввести количественные характеристики моделируемых процессов. На этом уровне векто-

рам соответствуют вектор-столбцы и их компоненты, а отображениям – матрицы. На программном уровне моделирования модель представляется в виде структуры данных; столбцам и матрицам обычно сопоставляются массивы, функциям – процедуры.

Для практики моделирования представляется удобной и полезной возможность раздельно рассматривать понятия, относящиеся к различным уровням моделирования, и попеременно переключаться с одной точки зрения на другую при рассмотрении конкретных соотношений. При этом можно на инвариантном уровне моделирования дать максимально компактную формулировку основных алгоритмов, а специфические тонкости, связанные с выбором систем координат и программной реализацией алгоритмов, трактовать уже как детализацию полученных соотношений (носящую во многом рутинный характер).

1.2. Матрично-операторные методы в кинематике абсолютно твёрдого тела

В настоящем пособии основным аппаратом описания кинематики пространственных механизмов служит метод винтов [**6**], который в последние годы получает всё большее распространение в механике роботов. Это связано как с его достаточно высокой вычислительной эффективностью [**1**,**7**–**9**], так и с тем, что данный формализм возникает как результат применения стандартных подходов современной линейной алгебры и обеспечивает, в определённом смысле, наиболее естественный способ описания движения твёрдого тела.

Заметим, что в кинематике можно выделить три основных раздела: геометрию движения (изучает перемещения точек и тел без учёта конкретной зависимости их от времени), теорию скоростей и теорию ускорений. Займёмся сперва геометрией движения.

К числу фундаментальных понятий механики относится понятие *системы отсчёта*. Под системой отсчёта *(СО)* понимается [**10**] произвольная геометрическая твёрдая среда, относительно которой рассматривается движение точек и тел. В математическом плане геометрическая твёрдая среда есть *евклидово точечное пространство* (т.е. аффинное пространство, для которого пространство свободных векторов евклидово – в нём введена операция скалярного умножения).

С любым абсолютно твёрдым телом (реальным или воображаемым) можно жёстко связать некоторую СО – *связанную систему отсчёта.* В дальнейшем, рассматривая тот или иной механизм, будем предполагать его звенья пронумерованными и обозначать СО, связанную с j-м звеном, через $\mathcal{E}_j$. Предположим также, что зафиксирована некоторая СО, которую назовём *условно неподвижной системой отсчёта* и обозначим $\mathcal{E}$ (при изучении движения механизмов естественно связать $\mathcal{E}$ с основанием: $\mathcal{E} = \mathcal{E}_0$).

Рассмотрим сперва геометрию движения одного абсолютно твёрдого тела и обозначим через $\mathcal{E}^*$ связанную с ним систему отсчёта.

Следуя [**11**, **12**], будем проводить различие между точками самого тела и положениями, которые они занимают в СО $\mathcal{E}$ в текущий момент времени t. Именно, точки геометрической твёрдой среды $\mathcal{E}^*$ будем называть *телесными точками* и помечать звёздочкой.

Таким образом, если $M^* \in \mathcal{E}^*$ – какая-либо телесная точка, то $M \in \mathcal{E}$ – её текущее положение. Аналогично, если $\mathcal{V}^*$ и $\mathcal{V}$ – пространства свободных векторов для точечных пространств $\mathcal{E}^*$ и $\mathcal{E}$, то вектор $\overline{M^*N^*} \in \mathcal{V}^*$ служит примером вектора в связанной СО (*телесного вектора*), причём в условно неподвижной СО ему в текущий момент времени соответствует вектор $\overline{MN} \in \mathcal{V}$.

Компактные и выразительные средства для записи различных соотношений и формул, относящихся к аффинной геометрии, предоставляет *барицентрическое исчисление* [**3**, **13**]. Символика его проста: операцию откладывания вектора $\bar{\mathbf{u}}$ от точки M, т.е. нахождения точки N, для которой $\overline{MN} = \bar{\mathbf{u}}$, обозначают как обычное сложение: $N = M + \bar{\mathbf{u}}$ или $N = \bar{\mathbf{u}} + M$, а обратную ей операцию получения вектора $\bar{\mathbf{u}} = \overline{MN}$ – как вычитание: $\bar{\mathbf{u}} = N - M$.

При этом понятие суммы точек не определено; в общем случае линейная комбинация точек $P_0, \ldots, P_k$ с коэффициентами λ_i в барицентрическом исчислении имеет смысл в следующих двух случаях:

1) при $\sum_i \lambda_i = 1$ (*барицентрическая* линейная комбинация);

2) при $\sum_i \lambda_i = 0$ (*сбалансированная* линейная комбинация).

В первом из этих случаев комбинация трактуется как точка $M = Q + \sum_i \lambda_i (P_i - Q)$, а во втором – как вектор $\bar{\mathbf{u}} = \sum_i \lambda_i (P_i - Q)$ (в обоих случаях результат от выбора точки Q не зависит).

Удобство символики барицентрического исчисления – в облегчении автоматизации различных геометрических выкладок. К тому же в этих выкладках можно применять обычные алгебраические правила преобразования сумм и разностей, не заботясь об интерпретации промежуточных результатов.

Точки $P_0, \ldots, P_k$ в аффинном пространстве E называют *аффинно независимыми*, если ни одна из них не является барицентрической комбинацией остальных. Совокупность всевозможных барицентрических комбинаций конечных подсемейств точек множества $F \subset E$ называется [**3**] *аффинной оболочкой* этого множества и представляет собой линейное многообразие в E.

Заметим, что связанная СО $\mathcal{E}^*$ является не чем иным, как аффинной оболочкой твёрдого тела.

Если пространство E n-мерно, то оно само является аффинной оболочкой произвольного семейства $P_0, \ldots, P_n$ аффинно независимых

своих точек. При этом любая точка $M \in E$ однозначно представима в виде барицентрической комбинации

$$M = \sum_j \lambda_j P_j ;$$

в этом контексте семейство $\{P_j\}_{j=0}^{j=n}$ называется *точечным базисом* в E, а числа λ_j – *барицентрическими координатами* точки M. Введение точечных базисов и барицентрических координат обеспечивает переход на координатный уровень моделирования.

Пусть теперь E и E' – два аффинных пространства. Рассмотрим *аффинное отображение* $\alpha: E \to E'$, т.е. такое отображение, что для любой барицентрической комбинации точек из E справедливо

$$\alpha\left(\sum_j \lambda_j P_j \right) = \sum_j \lambda_j \alpha(P_j) . \qquad (1)$$

Для аффинных отображений понятие барицентрической комбинации также имеет смысл. Так, разностью отображений α и β является отображение $\gamma: E \to V'$, определяемое равенством

$$\forall M \in E \qquad \gamma(M) = \alpha(M) - \beta(M) .$$

Если взять в E и E' точечные базисы $\{P_j\}_{j=0}^{j=n}$ и $\{P_i'\}_{i=0}^{i=m}$ и разложить по ним произвольную точку M и её образ $\alpha(M)$:

$$M = \sum_j \lambda_j P_j , \qquad \alpha(M) = \sum_i \mu_i P_i' ,$$

то в силу (1) получаем

$$\alpha(M) = \sum_{j=0}^{n} \lambda_j \alpha(P_j) = \sum_{j=0}^{n} \lambda_j \sum_{i=0}^{m} \alpha_{ij} P_i' \equiv \sum_{i=0}^{m} \mu_i P_i' ;$$

откуда, обозначая через λ и μ столбцы из соответствующих коэффициентов, имеем

$$\mu = B_\alpha \lambda . \qquad (2)$$

Матрица B_α в (2) – *барицентрическая матрица* отображения α. Её коэффициенты α_{ij} (*барицентрические компоненты* этого отображения) определяются формулами

$$\alpha(P_j) = \sum_{i=0}^{m} \alpha_{ij} P_i' .$$

Всякому аффинному отображению $\alpha: E \to E'$ ставится в соответствие [**12**, **14**] линейный оператор $\overline{\mathbf{A}}: V \to V'$, обозначаемый $\nabla\alpha$ и называющийся *линейным оператором, ассоциированным* с аффинным отображением α. Он однозначно определяется равенством

$$\overline{\mathbf{A}}\,\overline{\mathbf{u}} = \alpha(M + \overline{\mathbf{u}}) - \alpha(M) \qquad \forall M \in E, \ \forall \overline{\mathbf{u}} \in V . \qquad (3)$$

На координатном уровне ассоциированный линейный оператор представляется той же матрицей B_α, что и само отображение α.

Введение точечных базисов лежит в основе формализма барицентрических координат. Этот формализм позволяет интерпретировать на координатном уровне различные формулы с участием точек и аффинных отображений, не привлекая аппарат векторов и ассоциированных линейных операторов. Впрочем, барицентрические координаты – не единственное средство координатного описания в аффинной геометрии.

Характерный для барицентрического исчисления единообразный подход к точкам и векторам позволяет, наряду с точечными, ввести в рассмотрение *смешанные* базисы, состоящие из точек и векторов.

Так, перейдём от точечного базиса $\{P_j\}_{j=0}^{j=n}$ в пространстве E к *точечно-векторному базису*, состоящему из n векторов

$$\bar{\mathbf{e}}_j = P_j - P_n , \qquad j = 0, \ldots, n-1 ,$$

и одной-единственной точки P_n.

Любая точка $M \in E$ и любой вектор $\bar{\mathbf{u}} \in V$ однозначно представимы в виде линейных комбинаций элементов нового базиса. Коэффициенты этих комбинаций называются *однородными координатами* точки M и вектора $\bar{\mathbf{u}}$.

Однородными координатами точки M называют также [3] любые наборы коэффициентов, получающиеся из введённых в предыдущем абзаце умножением на произвольное число, отличное от нуля (впрочем, нам это обобщение не понадобится).

Выясним, как связаны между собой однородные и барицентрические координаты (полученные выводы будут справедливы, если точечно-векторный базис получается из исходного точечного изложенным выше способом).

Заметим, что векторы $\{\bar{\mathbf{e}}_j\}$ образуют базис (обычный) в пространстве свободных векторов V, так что в выражении вектора $\bar{\mathbf{u}}$ через элементы точечно-векторного базиса точка P_n вообще не принимает участия. Поэтому первые n однородных координат вектора $\bar{\mathbf{u}}$ совпадают с его компонентами u_j, а $u_n = 0$.

Пусть теперь вектор $\bar{\mathbf{u}}$ есть радиус-вектор точки M относительно полюса P_n:

$$M = \bar{\mathbf{u}} + P_n = \sum_{j=0}^{j<n} u_j \bar{\mathbf{e}}_j + 1 \cdot P_n ;$$

следовательно, первые n однородных координат точки M совпадают с компонентами её радиуса-вектора, а последняя равна единице.

Пользуясь тем, что $\bar{\mathbf{e}}_j = P_j - P_n$, получаем:

$$M = \sum_{j=0}^{j<n} u_j P_j + \left(1 - \sum_{j=0}^{j<n} u_j\right) P_n , \qquad \bar{\mathbf{u}} = \sum_{j=0}^{j<n} u_j P_j + \left(-\sum_{j=0}^{j<n} u_j\right) P_n .$$

Таким образом, первые n барицентрических координат точки (или вектора) совпадают с соответствующими однородными координатами; значение последней координаты для точки вполне определяется условием барицентричности, а для вектора – условием сбалансированности.

Получим теперь матричное представление аффинного отображения $\alpha: E \to E'$ в однородных координатах.

Введём вместо точечного базиса $\{P'_i\}_{i=0}^{i=m}$ в пространстве E' точечно-векторный (как это делалось выше для пространства E), положив

$$\bar{\mathbf{e}}'_i = P'_i - P'_m, \qquad i = 0, \ldots, m-1.$$

Нетрудно убедиться, что столбец однородных координат точки $\alpha(M) \in E'$ можно вычислить, умножая столбец однородных координат точки $M \in E$ на матрицу, имеющую такое блочное представление:

$$\begin{pmatrix} A & p \\ 0 & 1 \end{pmatrix},$$

где A – матрица оператора $\bar{\mathbf{A}} \equiv \nabla\alpha$ относительно базисов $\{\bar{\mathbf{e}}_j\}$ и $\{\bar{\mathbf{e}}'_i\}$, а p – столбец компонент вектора $\alpha(P_n) - P'_m$.

При этом компоненты матрицы A и столбца p выражаются через барицентрические компоненты отображения α следующим образом:

$$a_{ij} = \alpha_{ij} - \alpha_{in}, \qquad p_i = \alpha_{in};$$

здесь $i = 0, \ldots, m-1$, $j = 0, \ldots, n-1$.

Записанные формулы позволяют также найти первые m строк барицентрической матрицы B_α, если известны A и p. Последнюю же строку матрицы B_α легко получить, пользуясь барицентричностью её столбцов:

$$\alpha_{mj} = 1 - \alpha_{0j} - \ldots - \alpha_{m-1,j}, \qquad j = 0, \ldots, n.$$

Вернёмся к геометрии движения абсолютно твёрдого тела.

Когда говорится о движении тела в какой-либо системе отсчёта (например, $\mathcal{E}$), то это значит, что в каждый момент времени t любой точке M^* тела сопоставлено её текущее положение M (иными словами, для каждого t задано отображение точек тела в пространство $\mathcal{E}$). Это отображение называется [**12**] *конфигурацией тела* (в данной системе отсчёта) и далее обозначается $\boldsymbol{\varkappa}$. Для абсолютно твёрдых тел (в отличие от ситуации в механике сплошных сред) оно продолжается на все пространство $\mathcal{E}^*$ и является аффинным отображением.

Более того, отображение $\boldsymbol{\varkappa}: \mathcal{E}^* \to \mathcal{E}$ сохраняет расстояния и, следовательно, является изометрией. Поэтому ассоциированный с $\boldsymbol{\varkappa}$ линейный оператор $\bar{\boldsymbol{\Gamma}}: \mathcal{V}^* \to \mathcal{V}$ оказывается ортогональным оператором: $\bar{\boldsymbol{\Gamma}}^{-1} = \bar{\boldsymbol{\Gamma}}^{\mathrm{T}}$. Символ $^{\mathrm{T}}$ здесь обозначает транспонирование: *транспонированный оператор* для оператора $\bar{\mathbf{C}}: X \to Y$ – это оператор $\bar{\mathbf{C}}^{\mathrm{T}}: X \to Y$, определяемый [**15**] с помощью скалярного произведения тождеством

$$(\bar{\mathbf{v}}, \bar{\mathbf{C}}^{\mathrm{T}}\bar{\mathbf{w}}) = (\bar{\mathbf{C}}\bar{\mathbf{v}}, \bar{\mathbf{w}}) \qquad \forall\, \bar{\mathbf{v}} \in X, \ \forall\, \bar{\mathbf{w}} \in Y.$$

Оператор $\bar{\boldsymbol{\Gamma}}$ носит название *оператора ориентации* абсолютно твёрдого тела. Для каждого t он устанавливает взаимно однозначное соответствие между телесными и пространственными векторами.

Из определения ассоциированного линейного оператора (3) следует, что текущее положение любой телесной точки B^* можно найти, если известны положение некоторой фиксированной точки (*полюса*) A^* и оператор ориентации: $B = A + \overline{\mathbf{r}}^*$, где $\overline{\mathbf{r}}^* = \overline{A^* B^*} = \text{const}$. Переходя к радиусам-векторам $\overline{\mathbf{r}}_A = A - O$, $\overline{\mathbf{r}}_B = B - O$, где O – полюс в $\mathcal{E}$, получаем *основную формулу геометрии движения* (в векторном представлении):

$$\overline{\mathbf{r}}_B = \overline{\mathbf{r}}_A + \overline{\Gamma}\,\overline{\mathbf{r}}^*. \qquad (4)$$

Для перехода на координатный уровень моделирования достаточно в основной формуле геометрии движения заменить векторы столбцами их компонент, а оператор ориентации – его матрицей Γ.

Разумеется, вместо использования оператора ориентации и радиуса-вектора полюса можно оперировать непосредственно с самим отображением $\varkappa$. При этом равенство $B = \varkappa(B^*)$ в соответствии с (2) переходит на координатном уровне в формулу

$$b = B_{\varkappa} b^*, \qquad (5)$$

где b и b^* – столбцы барицентрических координат точек B и B^* (основная формула геометрии движения в барицентрическом представлении).

Перейдём теперь к теории скоростей.

Определения производных от точки и аффинного отображения в символике барицентрического исчисления выглядят так же, как для скаляров и векторов. В частности, для текущего положения B телесной точки B^* (оно, разумеется, меняется с течением t: $B = B(t)$) производная по времени совпадает с вектором её скорости:

$$\dot{B} \equiv \frac{\mathrm{d}B}{\mathrm{d}t} = \lim_{\Delta t \to 0} \frac{B(t+\Delta t) - B(t)}{\Delta t} = \lim_{\Delta t \to 0} \frac{B(t+\Delta t) - O - (B(t) - O)}{\Delta t} =$$

$$= \lim_{\Delta t \to 0} \frac{\overline{\mathbf{r}}_B(t+\Delta t) - \overline{\mathbf{r}}_B(t)}{\Delta t} = \frac{\overline{\mathbf{r}}_B}{\mathrm{d}t} \equiv \overline{\mathbf{v}}_B .$$

Аналогично определяется производная от конфигурации:

$$\dot{\varkappa} = \lim_{\Delta t \to 0} \frac{\varkappa_{t+\Delta t} - \varkappa_t}{\Delta t} .$$

Отображение $\dot{\varkappa}\colon \mathcal{E}^* \to \mathcal{V}$ сопоставляет каждой телесной точке B^* вектор $\overline{\mathbf{v}}_B$ её скорости в данный момент времени: $\dot{\varkappa}(B^*) = \overline{\mathbf{v}}_B$.

В векторном анализе наряду с обычными производными линейных операторов важную роль играют [**15**] их мультипликативные производные.

Для оператора $\overline{\mathbf{C}}: X \to Y$ (который предполагается зависящим от t как от параметра: $\overline{\mathbf{C}} = \overline{\mathbf{C}}(t)$) *мультипликативной производной* называется оператор

$$\frac{\mathrm{D}\overline{\mathbf{C}}}{\mathrm{d}t} = \dot{\overline{\mathbf{C}}}\,\overline{\mathbf{C}}^{-1}, \quad \det C \neq 0; \qquad \frac{\mathrm{D}\overline{\mathbf{C}}}{\mathrm{d}t}: Y \to Y .$$

Мультипликативную производную можно рассматривать и для аффинных отображений.

Так, отображение $\dfrac{\mathrm{D}\varkappa}{\mathrm{d}t} : \mathscr{E} \to \mathscr{V}$, определяемое как $\dot{\varkappa} \circ \varkappa^{-1}$, задаёт фактически векторное *поле скоростей* абсолютно твёрдого тела.

Перепишем основную формулу геометрии движения (4) в виде

$$\varkappa(B^*) \;=\; \varkappa(A^*) + \overline{\Gamma}\,\overline{\mathbf{r}}^{\,*}$$

и продифференцируем её почленно:

$$\dot{\varkappa}(B^*) \;=\; \dot{\varkappa}(A^*) + \dot{\overline{\Gamma}}\,\overline{\mathbf{r}}^{\,*}.$$

Учитывая, что $\overline{\mathbf{r}}^{\,*} = \overline{\Gamma}^{\,\mathrm{T}}\,\overline{\mathbf{r}}_{AB}$, и переходя от обычных производных к мультипликативным, получим

$$\frac{\mathrm{D}\varkappa}{\mathrm{d}t}(B) \;=\; \frac{\mathrm{D}\varkappa}{\mathrm{d}t}(A) + \frac{\mathrm{D}\overline{\Gamma}}{\mathrm{d}t}\,\overline{\mathbf{r}}_{AB}\,, \qquad \frac{\mathrm{D}\overline{\Gamma}}{\mathrm{d}t} \equiv \dot{\overline{\Gamma}}\,\overline{\Gamma}^{\,\mathrm{T}}. \tag{6}$$

В выражении для мультипликативной производной оператора ориентации вместо обратного оператора записан транспонированный (для ортогональных операторов это – одно и то же).

Мультипликативная производная оператора ориентации $\overline{\Gamma}$ называется [**11**] *оператором угловой скорости* абсолютно твёрдого тела. Этот оператор – антисимметричный. Как и всякому антисимметричному линейному оператору в трёхмерном ориентированном евклидовом пространстве, ему ставится в соответствие некоторый вектор, а именно вектор угловой скорости $\overline{\omega}$. С учётом этого формула (6) оказывается эквивалентной хорошо известной формуле Эйлера для распределения скоростей точек твёрдого тела:

в операторной записи – $\quad \overline{\mathbf{v}}_B \;=\; \overline{\mathbf{v}}_A + \overset{\vee}{\overline{\omega}}\,\overline{\mathbf{r}}_{AB}\,;$

в векторной записи – $\quad \overline{\mathbf{v}}_B \;=\; \overline{\mathbf{v}}_A + [\,\overline{\omega}, \overline{\mathbf{r}}_{AB}\,]\,.$

Здесь и далее квадратные скобки обозначают векторное произведение, а символ $^{\vee}$ – операцию, сопоставляющую данному вектору антисимметричный оператор по правилу

$$\forall\, \overline{\mathbf{u}} \qquad \overset{\vee}{\overline{\mathbf{a}}}\,\overline{\mathbf{u}} \;=\; [\,\overline{\mathbf{a}}, \overline{\mathbf{u}}\,]\,.$$

Последнее слагаемое в формуле Эйлера можно преобразовать:

$$[\,\overline{\omega}, \overline{\mathbf{r}}_{AB}\,] \;=\; -[\,\overline{\mathbf{r}}_{AB}, \overline{\omega}\,] \;=\; [\,\overline{\mathbf{r}}_{BA}, \overline{\omega}\,] \;=\; \overset{\vee}{\overline{\mathbf{r}}}_{BA}\,\overline{\omega}\,;$$

тогда эта формула примет вид

$$\overline{\mathbf{v}}_B \;=\; \overset{\vee}{\overline{\mathbf{r}}}_{BA}\,\overline{\omega} + \overline{\mathbf{v}}_A\,. \tag{7}$$

Итак, поле скоростей абсолютно твёрдого тела однозначно определяется двумя векторами: свободным вектором $\overline{\omega}$ и вектором скорости в точке A (последний можно считать приложенным в этой точке). Скорости во всех других его точках можно найти по формуле (7).

Используя этот результат кинематики твёрдого тела, нетрудно перейти к общему понятию винта.

1.3. Кинематические винты

Начнём с общего определения понятия винта.

Винт (в пространстве $\mathcal{E}$) – объект $\overline{\mathbf{U}}$, который при выборе какой-либо точки $A \in \mathcal{E}$ за полюс представляется совокупностью двух векторов $\bar{\mathbf{u}}$ и $\bar{\mathbf{u}}^\circ \equiv \bar{\mathbf{u}}^\circ_A$, принадлежащих $\mathcal{V}$, причём при смене полюса (переходе, скажем, от точки A к точке B) $\bar{\mathbf{u}}$ не меняется, а $\bar{\mathbf{u}}^\circ$ преобразуется по формуле

$$\bar{\mathbf{u}}^\circ_B = \overset{\vee}{\bar{\mathbf{r}}}_{BA}\, \bar{\mathbf{u}} + \bar{\mathbf{u}}^\circ_A . \qquad (8)$$

При этом векторы $\bar{\mathbf{u}}$ и $\bar{\mathbf{u}}^\circ$ соответственно называются [6] *главным вектором винта* и *моментом винта*, а вместе – его *элементами приведения*.

Эквивалентное определение: **винт** – совокупность вектора $\bar{\mathbf{u}}$ и векторного поля $\bar{\mathbf{u}}^\circ = \bar{\mathbf{u}}^\circ(B)$, удовлетворяющих соотношению (8).

Таким образом, если полюс выбран, то винт $\overline{\mathbf{U}}$ эквивалентен блочному вектору, составленному из элементов приведения:

$$\overline{\mathbf{U}} \sim \overline{\mathbf{U}}_A \equiv \begin{pmatrix} \bar{\mathbf{u}} \\ \bar{\mathbf{u}}^\circ \end{pmatrix}.$$

Нетрудно убедиться, что совокупность всех винтов в $\mathcal{E}$ образует шестимерное векторное пространство, которое мы обозначим $\mathcal{P}$.

Заметим теперь, что формула (8) только обозначениями отличается от (7). Поэтому естественно ввести следующее определение:

Кинематический винт абсолютно твёрдого тела – совокупность вектора угловой скорости $\bar{\boldsymbol{\omega}}$ и поля $\bar{\mathbf{v}}(B)$ скоростей телесных точек. Иначе говоря, это – объект $\overline{\mathbf{U}}$, который при выборе какой-либо точки $A \in \mathcal{E}$ за полюс представляется совокупностью векторов $\bar{\boldsymbol{\omega}}$ и $\bar{\mathbf{v}}_A$.

Строго говоря, в (7) точка A – не “какая-либо”, а текущее положение телесной точки A^* (полюса в $\mathcal{E}^*$; т.е. здесь за полюс в $\mathcal{E}$ взята движущаяся точка). Но это не принципиально: полюс можно брать любым.

Далее в качестве полюса в $\mathcal{E}$ будет использоваться неподвижная точка O. При этом элементами приведения кинематического винта будут $\bar{\boldsymbol{\omega}}$ и $\bar{\mathbf{v}}_O$ – значение векторного поля $\bar{\mathbf{v}}(B)$ в точке O. Это значение, разумеется, связано с $\bar{\mathbf{v}}_A$ соотношением

$$\bar{\mathbf{v}}_O = \overset{\vee}{\bar{\mathbf{r}}}_{OA}\, \bar{\boldsymbol{\omega}} + \bar{\mathbf{v}}_A . \qquad (9)$$

В теории винтов важную роль играют инварианты винта (т.е. такие величины, выражающиеся через элементы приведения, которые не зависят от выбора полюса). Основных (независимых) инвариантов у винта – два.

Вектор $\bar{\mathbf{u}}$ и скаляр $(\bar{\mathbf{u}}^\circ, \bar{\mathbf{u}})$ называются соответственно *первым* и *вторым* инвариантами винта. На основании анализа инвариантов строится полезная классификация винтов.

Винт называется *вырожденным*, если его 1-й инвариант (т.е. главный вектор $\bar{\mathbf{u}}$) равен нулю. Частным случаем вырожденного вин-

та является *нулевой винт*, у которого равны нулю оба элемента приведения.

Для невырожденного винта отношение

$$p = \frac{(\bar{\mathbf{u}}^{\circ}, \bar{\mathbf{u}})}{(\bar{\mathbf{u}}, \bar{\mathbf{u}})}$$

определено и не зависит от выбора полюса. Это – *параметр винта* (он также является инвариантом, но не независимым, поскольку выражается через первый и второй инварианты).

Винт называется *изотропным*, если его 2-й инвариант равен нулю. Изотропными являются все вырожденные винты, а также *винты нулевого параметра* (невырожденные винты, у которых $p=0$).

Среди винтов нулевого параметра выделяют *единичные винты*, у которых главный вектор $\bar{\mathbf{u}}$ является единичным вектором.

Прямая l называется *осью винта*, если его элементы приведения относительно любого полюса B, взятого на данной оси, сонаправлены этой оси. Для невырожденного винта ось определена однозначно.

Два винта называются *соосными*, если их оси совпадают.

В качестве базиса в векторном пространстве $\mathcal{P}$ можно брать произвольные шесть линейно независимых винтов. Но обычно работают с базисами специального вида.

Пусть $O \in \mathcal{E}$, а $\{\bar{\mathbf{e}}_i\}_{i=0}^{i<3}$ – правый ортонормированный базис в $\mathcal{V}$. Набор из шести винтов $\bar{\mathbf{E}}_0$, $\bar{\mathbf{E}}_1$, $\bar{\mathbf{E}}_2$, $\bar{\mathbf{E}}_0^{\circ}$, $\bar{\mathbf{E}}_1^{\circ}$, $\bar{\mathbf{E}}_2^{\circ}$, которые относительно полюса O имеют блочное представление

$$\bar{\mathbf{E}}_i \sim \begin{pmatrix} \bar{\mathbf{e}}_i \\ 0 \end{pmatrix}, \qquad \bar{\mathbf{E}}_i^{\circ} \sim \begin{pmatrix} 0 \\ \bar{\mathbf{e}}_i \end{pmatrix},$$

называется *плюккеровым базисом* в $\mathcal{P}$, соответствующим данным точке и базису. Компоненты винта относительно плюккерова базиса называют *плюккеровыми координатами* этого винта (первые три из них совпадают с компонентами главного вектора, а вторые три – с компонентами момента винта).

Стоит отметить формулу $\bar{\mathbf{E}}_i^{\circ} = \bar{\boldsymbol{\varepsilon}}\,\bar{\mathbf{E}}_i$, где $\bar{\boldsymbol{\varepsilon}}: \mathcal{P} \to \mathcal{P}$ – *оператор Клиффорда*, т.е. линейный оператор на пространстве винтов, при любом выборе полюса в $\mathcal{E}$ имеющий блочное представление

$$\bar{\boldsymbol{\varepsilon}} \sim \begin{pmatrix} 0 & 0 \\ \bar{\mathbf{I}} & 0 \end{pmatrix}$$

(так что вторая тройка винтов плюккерова базиса может быть получена из первой тройки умножением входящих в неё винтов на $\bar{\boldsymbol{\varepsilon}}$).

Заметим, что винты $\bar{\mathbf{E}}_i$ – единичные, а винты $\bar{\mathbf{E}}_i^{\circ}$ – соосные им вырожденные винты.

Можно рассматривать винты не только в условно неподвижной системе отсчёта $\mathcal{E}$, но и в любой другой (например, в $\mathcal{E}^*$). Наличие между пространствами $\mathcal{E}^*$ и $\mathcal{E}$ в текущий момент времени взаимно однозначного соот-

ветствия (реализуемого посредством отображения $\boldsymbol{\varkappa}$) порождает взаимно однозначное соответствие и между пространствами винтов $\mathcal{P}^*$ и $\mathcal{P}$. Это соответствие оказывается линейным оператором.

Линейный оператор $\overline{\mathbf{G}}: \mathcal{P}^* \to \mathcal{P}$, сопоставляющий для текущей конфигурации телесным винтам винты в условно неподвижной системе отсчёта, называется [6,16] *верзором* абсолютно твёрдого тела. Если в $\mathcal{E}^*$ и $\mathcal{E}$ в качестве полюсов взяты точки A^* и O, то для верзора справедливо следующее блочное представление:

$$\overline{\mathbf{G}} \sim \begin{pmatrix} \overline{\mathbf{\Gamma}} & 0 \\ \overline{\mathbf{K}} & \overline{\mathbf{\Gamma}} \end{pmatrix}, \qquad \text{где} \quad \overline{\mathbf{K}} \equiv \overline{\overset{\vee}{\mathbf{r}}}_{OA}\,\overline{\mathbf{\Gamma}}\ . \tag{10}$$

Отметим, что верзор и конфигурация однозначно определяют друг друга. Всего в матрице верзора – 36 элементов, но среди них есть заведомо нулевые и повторяющиеся, и на программном уровне моделирования можно хранить 12 чисел: 9 элементов матрицы Γ и 3 компоненты столбца r_{OA}.

Верзор абсолютно твёрдого тела служит [6] примером важного класса линейных операторов на пространстве винтов – винтовых аффиноров.

Линейный оператор $\overline{\mathbf{A}}: \mathcal{P} \to \mathcal{P}$ называется *винтовым аффинором*, если он коммутирует с оператором $\overline{\boldsymbol{\varepsilon}}$: $\overline{\mathbf{A}}\,\overline{\boldsymbol{\varepsilon}} = \overline{\boldsymbol{\varepsilon}}\,\overline{\mathbf{A}}$. Для того, чтобы оператор был винтовым аффинором, необходимо и достаточно, чтобы относительно какого-либо полюса B он имел представление

$$\overline{\mathbf{A}} \sim \begin{pmatrix} \overline{\mathbf{P}} & 0 \\ \overline{\mathbf{Q}} & \overline{\mathbf{P}} \end{pmatrix};$$

при переходе к новому полюсу O оператор $\overline{\mathbf{P}}$ не меняется, а $\mathbf{Q}$ преобразуется по формуле $\overline{\mathbf{Q}}_O = [\,\overline{\overset{\vee}{\mathbf{r}}}_{OB}, \overline{\mathbf{P}}\,] + \overline{\mathbf{Q}}_B$ (квадратные скобки обозначают коммутатор линейных операторов).

Можно показать, что винтовые аффиноры – это в точности те линейные операторы на пространстве винтов, умножение на которые сохраняет соосность винтов.

Винтовой аффинор называется *блочно-антисимметричным*, если его элементы приведения $\overline{\mathbf{P}}$ и $\overline{\mathbf{Q}}$ являются антисимметричными операторами. Всякому винту $\overline{\mathbf{U}}$ можно взаимно однозначно сопоставить блочно-антисимметричный винтовой аффинор $\overline{\overset{\vee}{\mathbf{U}}}$ по формуле

$$\overline{\mathbf{U}} \sim \begin{pmatrix} \overline{\mathbf{u}} \\ \overline{\mathbf{u}}^{\circ}_{B} \end{pmatrix} \quad \Rightarrow \quad \overline{\overset{\vee}{\mathbf{U}}} \sim \begin{pmatrix} \overline{\overset{\vee}{\mathbf{u}}} & 0 \\ \overline{\overset{\vee}{\mathbf{u}}}^{\circ}_{B} & \overline{\overset{\vee}{\mathbf{u}}} \end{pmatrix}.$$

Примером блочно-антисимметричного винтового аффинора служит *кинематический аффинор* абсолютно твёрдого тела, определяемый как мультипликативная производная верзора этого тела и связанный с кинематическим винтом $\overline{\mathbf{U}}$ соотношением:

$$\frac{\mathrm{D}\overline{\mathbf{G}}}{\mathrm{d}t} = \overset{\vee}{\overline{\mathbf{U}}};$$

иначе говоря, $\overset{\vee}{\overline{\mathbf{U}}} = \dot{\overline{\mathbf{G}}}\,\overline{\mathbf{G}}^{-1}$.

1.4. Координаты Денавита – Хартенберга

Рассмотрим механизм, содержащий n подвижных звеньев, которые образуют простую кинематическую цепь. Общее число m кинематических соединений будет равно n для открытой цепи и $n+1$ для замкнутой.

Будем предполагать, что звенья пронумерованы так: номер *1* имеет звено, соединённое с основанием, номер *2* – звено, соединённое с *1*-м звеном, и т.п. Пронумеруем и кинематические соединения, присвоив номер j соединению звеньев с номерами j и $j+1$ (в случае замкнутой цепи номер $n+1$ получит соединение между n-м звеном и основанием).

Обозначим через $\mathcal{E}_j$ систему отсчёта, связанную с j-м звеном; соответствующие пространства свободных векторов и винтов обозначим через $\mathcal{V}_j$ и $\mathcal{P}_j$.

Пусть $\boldsymbol{\varkappa}_j : \mathcal{E}_j \to \mathcal{E}$ – конфигурация j-го звена в условно неподвижной системе отсчёта $\mathcal{E} = \mathcal{E}_0$ (далее $\boldsymbol{\varkappa}_j$ будет называться *абсолютной* конфигурацией), а $\boldsymbol{\varkappa}_{ij} : \mathcal{E}_j \to \mathcal{E}_i$ – конфигурация этого же звена относительно системы отсчёта $\mathcal{E}_i$ (*относительная* конфигурация).

Аналогично, одинарными и двойными индексами, будем отмечать и все остальные кинематические величины, вводимые для звеньев цепи.

Далее полагаем $i \equiv j-1$. Рассмотрим вычисление относительных верзоров $\overline{\mathbf{G}}_{ij}$. В силу (10) для этого достаточно уметь вычислять оператор ориентации $\overline{\mathbf{\Gamma}}_{ij}$ и радиус-вектор $\overline{\mathbf{r}}_{ij} \equiv O_{ij} - O_i^*$ текущего положения $O_{ij} \equiv \boldsymbol{\varkappa}_{ij}(O_j^*)$ полюса $O_j^* \in \mathcal{E}_j$.

Такие вычисления проводятся на координатном уровне моделирования, и значительную актуальность приобретает вопрос рационального выбора систем координат в пространствах $\mathcal{E}_j$.

Обозначим через $S_j^* \equiv O_j^* x_j^* y_j^* z_j^*$ систему координат в $\mathcal{E}_j$, а через $S_j \equiv O_j x_j y_j z_j$ – её текущую конфигурацию в $\mathcal{E}$. Единичные векторы системы S_j обозначим через $\overline{\mathbf{e}}_{ij}$, $i = 0, \ldots, 2$.

Значительную популярность в робототехнике приобрела [**1**,**2**] процедура введения систем координат, предложенная Ж.Денавитом и Р.Хартенбергом. Этот подход ориентирован на простые цепи с кинематическими соединениями V-го класса (вращательными, поступательными, винтовыми).

Итак, рассмотрим алгоритм Денавита – Хартенберга.

Алгоритм Денавита – Хартенберга состоит из двух частей. В ряде случаев он допускает определённую свободу выбора (неоднозначность). Такие места алгоритма отмечены комментарием "**(!)**".

Часть 1. Выбор осей.

D1. Для $j = 0,\dots,m-1$ направить ось z_j вдоль оси $(j+1)$-го сочленения. Ось z_m направить произвольно **(!)**.

D2. Задать точку O на оси z_0 **(!)**, а ось x_0 направить перпендикулярно оси z_0 **(!)**.

D3. Для $j = 1,\dots,m$ взять точку O_j на оси z_j:
- z_{j-1} и z_j скрещиваются $\Rightarrow$ на пересечении с общей нормалью к z_{j-1} и z_j;
- пересекаются $\Rightarrow$ в точке пересечения;
- параллельны или совпадают $\Rightarrow$ в любом месте оси z_j **(!)**.

D4. Для $j = 1,\dots,m$ направить ось x_j:
- z_{j-1} и z_j скрещиваются или параллельны $\Rightarrow$ вдоль общей нормали к ним;
- пересекаются $\Rightarrow$ перпендикулярно z_{j-1} и z_j;
- совпадают $\Rightarrow$ перпендикулярно z_j **(!)**.

D5. Для $j = 0,\dots,m$ направить ось y_j таким образом, чтобы выполнялось соотношение $\bar{\mathbf{e}}_{1j} = [\,\bar{\mathbf{e}}_{2j}, \bar{\mathbf{e}}_{0j}\,]$.

Часть 2. Определение параметров ($j = 1,\dots,m$).

D6. θ_j (*угол в сочленении*) – угол поворота вокруг z_{j-1} от x_{j-1} до x_j.

D7. d_j (*звенное расстояние*): $d_j = \mathrm{pr}_{z_{j-1}} \overline{O_{j-1}A_j}$, где A_j – точка пересечения осей z_{j-1} и x_j.

D8. α_j (*угол скручивания*) – угол поворота вокруг x_j от z_{j-1} до z_j.

D9. a_j (*длина звена*): $a_j = \mathrm{pr}_{x_j} \overline{A_j O_j}$.

Геометрия механизма набором параметров Денавита – Хартенберга θ_j, d_j, α_j, a_j характеризуется однозначно.

Далее ограничимся случаем, когда механизм образует открытую кинематическую цепь. В рассматриваемой ситуации n обобщённых координат, задающих текущую конфигурацию механизма, можно выбрать так, чтобы координата с номером j описывала конфигурацию j-го звена относительно $(j-1)$-го (в робототехнике такие координаты q_j называют *координатами в сочленении*).

За координату в j-м сочленении принимается:
- в случае поступательной пары – звенное расстояние d_j;

– в случае вращательной или винтовой пары – угол в сочленении θ_j.

Остальные параметры Денавита – Хартенберга оказываются постоянными (исключая случай винтовой пары, когда звенное расстояние изменяется, но его изменение связано с изменением угла в сочленении соотношением

$$d_j = p_j \theta_j, \qquad p_j = \pm \frac{h_j}{2\pi};$$

здесь p_j – параметр винтовой пары, h_j – шаг винта).

Вернмся к задаче вычисления относительных верзоров $\overline{\mathbf{G}}_{ij}$.

Заметим, что переход от S_{j-1} к S_j сводится к двум последовательным винтовым перемещениям:

– повороту на θ_j вокруг оси z_{j-1} и сдвигу на d_j вдоль неё же;

– повороту на α_j вокруг оси x_j и сдвигу на a_j вдоль неё же.

Поэтому

$$\Gamma_{ij} = \Gamma_{\theta_j} \Gamma_{\alpha_j} = \begin{pmatrix} \cos\theta_j & -\sin\theta_j \cos\alpha_j & \sin\theta_j \sin\alpha_j \\ \sin\theta_j & \cos\theta_j \cos\alpha_j & -\cos\theta_j \sin\alpha_j \\ 0 & \sin\alpha_j & \cos\alpha_j \end{pmatrix}; \qquad (11)$$

$$r_{ij} = \begin{pmatrix} 0 \\ 0 \\ d_j \end{pmatrix} + \Gamma_{\alpha_j} \begin{pmatrix} a_j \\ 0 \\ 0 \end{pmatrix} = \begin{pmatrix} a_j \cos\theta_j \\ a_j \sin\theta_j \\ d_j \end{pmatrix}. \qquad (12)$$

Итак, верзоры $\overline{\mathbf{G}}_{ij}$ вычислены.

1.5. Рекуррентные формулы кинематики для манипуляционного робота

Решим *прямую задачу геометрии движения*: найдём конфигурации звеньев механизма по известным координатам в сочленениях.

Абсолютная конфигурация $\varkappa_j$ может быть найдена, если известны конфигурация предыдущего звена $\varkappa_i$ и относительная конфигурация $\varkappa_{ij}$. Получаем рекуррентные формулы:

$$\varkappa_1 = \varkappa_{01}, \qquad \varkappa_j = \varkappa_i \circ \varkappa_{ij}; \qquad j = 2, \ldots, n, \quad i \equiv j. \qquad (13)$$

При верзорном описании движения формулам (13) соответствуют рекуррентные формулы для верзоров:

$$\overline{\mathbf{G}}_1 = \overline{\mathbf{G}}_{01}, \qquad \overline{\mathbf{G}}_j = \overline{\mathbf{G}}_i \overline{\mathbf{G}}_{ij}; \qquad j = 2, \ldots, n, \quad i \equiv j. \qquad (14)$$

Поскольку в § 1.4 задача вычисления относительных верзоров по известным значениям параметров Денавита – Хартенберга была решена, то по

формулам (14) все абсолютные верзоры $\overline{\mathbf{G}}_j$ могут быть найдены.

Перейдём теперь от геометрии движения к теории скоростей.

Возьмём мультипликативные производные от обеих частей формулы $\overline{\mathbf{G}}_j = \overline{\mathbf{G}}_i \overline{\mathbf{G}}_{ij}$:

$$\overset{\vee}{\overline{\mathbf{U}}}_j \equiv \frac{\mathrm{D}\overline{\mathbf{G}}_j}{\mathrm{d}t} = \frac{\mathrm{D}(\overline{\mathbf{G}}_i \overline{\mathbf{G}}_{ij})}{\mathrm{d}t} = \frac{\mathrm{D}\overline{\mathbf{G}}_i}{\mathrm{d}t} + \overline{\mathbf{G}}_i \frac{\mathrm{D}\overline{\mathbf{G}}_{ij}}{\mathrm{d}t} \overline{\mathbf{G}}_i^{-1} = \overset{\vee}{\overline{\mathbf{U}}}_i + \overline{\mathbf{G}}_i \overset{\vee}{\overline{\mathbf{U}}}_{ij} \overline{\mathbf{G}}_i^{-1}.$$

Переходя от кинематических аффиноров к кинематическим винтам, получаем

$$\overline{\mathbf{U}}_j = \overline{\mathbf{U}}_i + \overline{\mathbf{G}}_i \overline{\mathbf{U}}_{ij} . \qquad (15)$$

Последнее слагаемое в (15) – *кинематический винт j-го сочленения*: $\overline{\mathbf{V}}_j \equiv \overline{\mathbf{G}}_i \overline{\mathbf{U}}_{ij} \in \mathcal{P}$ (отметим, что $\overline{\mathbf{U}}_{ij} \in \mathcal{P}_i$).

Рекуррентные формулы для кинематических винтов можно записать в виде

$$\overline{\mathbf{U}}_0 = 0, \quad \overline{\mathbf{U}}_j = \overline{\mathbf{U}}_i + \overline{\mathbf{V}}_j ; \quad j = 1,\dots,n, \quad i \equiv j-1. \qquad (16)$$

Структуру манипулятора зададим [16] набором констант λ_j, μ_j:
– для вращательного сочленения $\lambda_j = 1$, $\mu_j = 0$;
– для поступательного сочленения $\lambda_j = 0$, $\mu_j = 1$;
– для винтового сочленения $\lambda_j = 1$, $\mu_j = p_j$.

Тогда, учитывая выбор осей координат в процедуре Денавита – Хартенберга, можно выразить относительные кинематические винты $\overline{\mathbf{U}}_{ij}$ через обобщённые скорости $\dot{q}_j$ (*скорости в сочленениях*):

$$\overline{\mathbf{U}}_{ij} = \dot{q}_j (\lambda_j \overline{\mathbf{E}}_{2i} + \mu_j \overline{\mathbf{E}}^{\circ}_{2i}) ,$$

где $\overline{\mathbf{E}}_{0i}$, $\overline{\mathbf{E}}_{1i}$, $\overline{\mathbf{E}}_{2i}$, $\overline{\mathbf{E}}^{\circ}_{0i}$, $\overline{\mathbf{E}}^{\circ}_{1i}$, $\overline{\mathbf{E}}^{\circ}_{2i}$ – базисные винты в $\mathcal{P}_i$.

Переходя к кинематическим винтам сочленений, получаем

$$\overline{\mathbf{V}}_j = \dot{q}_j \overline{\mathbf{D}}_j , \qquad (17)$$

где $\overline{\mathbf{D}}_j = \overline{\mathbf{G}}_i (\lambda_j \overline{\mathbf{E}}_{2i} + \mu_j \overline{\mathbf{E}}^{\circ}_{2i})$ – *направляющий винт j-го сочленения*.

Заметим, что вычисление направляющих винтов сочленений не вызывает каких-либо трудностей. Эти винты являются линейными комбинациями произведений верзора i-го тела на *2*-й и *5*-й (при сквозной нумерации от нуля) базисные винты; но столбец компонент произведения линейного оператора на базисный вектор есть соответствующий столбец матрицы этого оператора. Поэтому:

Столбец плюккеровых координат направляющего винта j-го сочленения есть линейная комбинация *2*-го и *5*-го столбцов матрицы верзора $\overline{\mathbf{G}}_i$ с коэффициентами λ_j и μ_j.

С учётом (17) рекуррентные формулы для кинематических винтов принимают вид

$$\overline{\mathbf{U}}_0 = 0, \quad \overline{\mathbf{U}}_j = \overline{\mathbf{U}}_i + \dot{q}_j \overline{\mathbf{D}}_j ; \quad j = 1,\dots,n, \quad i \equiv j-1. \qquad (18)$$

Пусть теперь механизм представляет собой шестизвенный манипулятор: $n=6$ (*6*-е звено – схват).

Из всех задач кинематики манипулятора рассмотрим так называемые задачи о скоростях. Они могут быть сформулированы [**6**,**16**] так:

Прямая задача о скоростях: найти кинематические винты звеньев (в частности, кинематический винт схвата) по заданным скоростям в сочленениях.

Обратная задача о скоростях: найти скорости в сочленениях по кинематическому винту схвата.

Для решения прямой задачи о скоростях можно воспользоваться непосредственно формулами (18). Но рациональнее их преобразовать.

От рекуррентных формул (18) перейдём к явным:

$$\overline{\mathbf{U}}_j = \sum_{k=1}^{j} \dot{q}_k \overline{\mathbf{D}}_k \quad - \text{ для } j\text{-го звена;} \qquad \overline{\mathbf{U}}_6 = \sum_{j=1}^{6} \dot{q}_j \overline{\mathbf{D}}_j \quad - \text{для схвата.}$$

Эти формулы и дают решение прямой задачи о скоростях.

Теперь перейдём к обратной задаче о скоростях.

Перепишем полученную для винта $\overline{\mathbf{U}}_6$ формулу в виде

$$U_6 = A_{\text{K}} \dot{q} \ , \tag{19}$$

где U_6 – столбец плюккеровых координат кинематического винта схвата; $\dot{q}$ – столбец из q_j; A_{K} – матрица, столбцами которой служат столбцы плюккеровых координат винтов $\overline{\mathbf{D}}_j$.

Итак, получена система линейных алгебраических уравнений относительно $\dot{q}_j$.

Поэтому решение обратной задачи о скоростях даётся формулой

$$\dot{q} = U_6 \backslash A_{\text{K}} \tag{20}$$

(обратная дробная черта обозначает [**17**] операцию *левого деления* столбца на матрицу, т.е. операцию вычисления решения системы линейных алгебраических уравнений).

Из формулы (20) видно, что для существования и единственности решения обратной задачи о скоростях для шестизвенного манипулятора необходима и достаточна невырожденность матрицы A_{K}.

Иными словами, столбцы плюккеровых координат направляющих винтов $\overline{\mathbf{D}}_j$ (и сами винты) должны быть линейно независимыми.

Отметим, что конфигурация механизма называется *особенной* [**6**], если направляющие винты сочленений линейно зависимы. Поэтому для существования и единственности решения обратной задачи о скоростях для шестизвенного манипулятора необходимо и достаточно, чтобы его конфигурация была неособенной.

Глава 2. Моделирование кинематики манипуляционных роботов средствами комплекса UM

Рассмотрим вопросы использования программного комплекса "Универсальный механизм" (**UM**) при решении задач кинематического анализа манипуляционных роботов. Изложение будем вести на примере прямой задачи кинематики для исполнительного механизма шестизвенного манипуляционного робота с вращательными и поступательными кинематическими сочленениями.

Отметим, что комплекс **UM** предназначен [**18**,**19**] для компьютерного моделирования многозвенных систем абсолютно твёрдых тел с разнообразными кинематическими соединениями и ориентирован на широкий класс задач статики, кинематики и динамики. Разработан он Д.Ю.Погореловым.

В последние два десятилетия был создан ряд достаточно универсальных программ для моделирования систем твёрдых тел. Но эти программы, как правило, ориентированы на достаточно мощные компьютеры. В отличие от них, комплекс **UM** реализован на персональных компьютерах семейства **PC IBM** с процессорами класса **Intel 80387** и выше, т.е. на наиболее дешёвых и распространённых ЭВМ. Вместе с тем он позволяет за приемлемое время моделировать системы, включающие свыше тысячи тел.

Таких показателей комплекс **UM** достигает, используя:

– анализ структуры графа системы с выбором оптимального перехода к структуре дерева;

– специальное кодирование входящих в уравнения модели символьных выражений, которое снимает проблемы приведения подобных членов и упрощения тригонометрических выражений;

– синтез уравнений модели в виде дифференциально-алгебраических уравнений (без сведения последних к дифференциальной форме);

– метод подсистем, при котором уравнения всей системы формируются на основе уравнений её простых подсистем (что особенно эффективно при наличии в модели кинематически тождественных подсистем).

Эффективному использованию комплекса **UM** в учебном процессе способствуют следующие его особенности: автоматизация наиболее трудоёмких операций по составлению математической модели системы; гибкий режим ввода исходных данных (с их контролем и визуализацией); возможность непосредственного просмотра промежуточных результатов работы комплекса; вывод результатов в числовой и графической форме; использование методов компьютерной графики, обеспечивающих визуализацию движения моделируемой системы.

2.1. Постановка прямой задачи кинематики

Рассмотрим следующий вариант постановки прямой задачи кинематического анализа для шестизвенного манипулятора [**18**].

Задача. Структура и геометрические характеристики манипулятора даны. Для координат q_j в его сочленениях заданы также их законы изменения. Требуется на заданном интервале времени $[0,\ t_{\text{fin}}]$

найти движение звеньев манипулятора и, в частности, определить, как зависят от времени угловая скорость $\overline{\omega}$ схвата и вектор $\overline{\mathbf{v}}_C$ скорости центра схвата C^*.

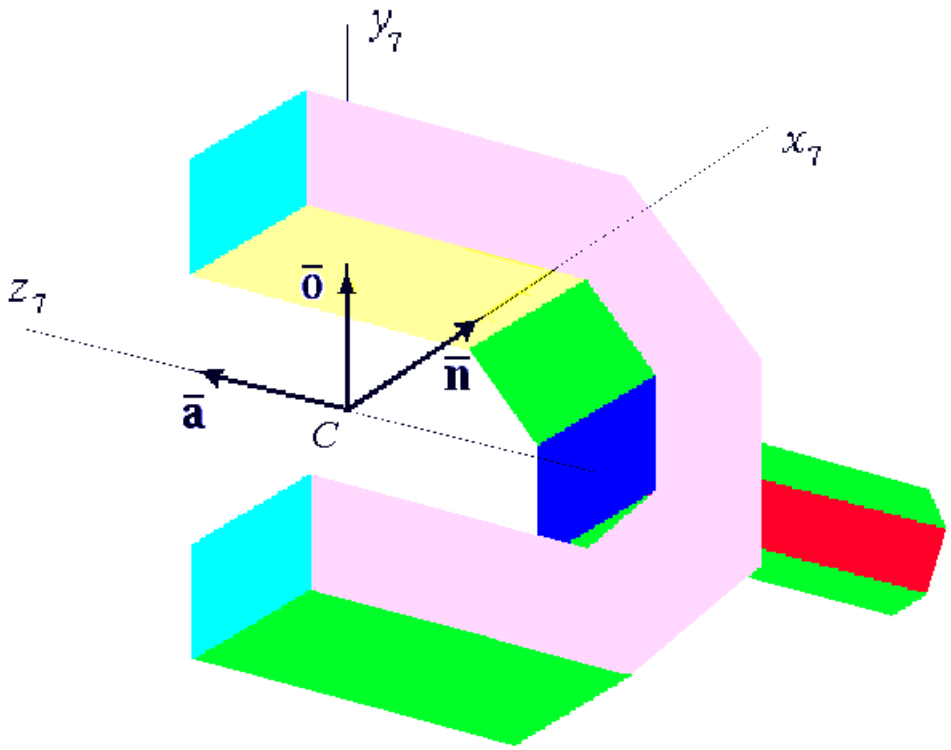
Под “центром схвата” понимается далее фиксированная точка связанной со схватом системы отсчёта (как правило, это – геометрический центр сжатых губок схвата [2]).

Далее предполагается, что координаты в сочленениях изменяются по гармоническому закону: $q_j = a_j \sin \nu_j t + b_j$.

Разные варианты заданий лабораторного практикума отличаются количеством и взаимным расположением вращательных и поступательных сочленений, а также геометрическими характеристиками звеньев робота.

Свяжем с каждым из звеньев систему координат Денавита – Хартенберга, как это делалось в предыдущей главе. При этом рекомендуется сделать эскиз, на котором будет изображена кинематическая схема манипулятора (в какой-либо конфигурации) вместе с осями систем координат. После этого можно определить значения параметров Денавита – Хартенберга и поместить их в таблицу, которая понадобится для формализованного описания геометрии сочленений робота.

Введём три единичных вектора в связанной со схватом СО: $\overline{\mathbf{n}}^*$ – вектор, перпендикулярный плоскости движения губок схвата; $\overline{\mathbf{o}}^*$ – вектор, определяющий направление движения губок при их раскрытии; $\overline{\mathbf{a}}^*$ – вектор, задающий ось симметрии схвата. Отвечающие этим телесным векторам в условно неподвижной СО векторы $\overline{\mathbf{n}}$, $\overline{\mathbf{o}}$, $\overline{\mathbf{a}}$ называются соответственно *вектором нормали*, *вектором ориентации* и *вектором подхода*).



В робототехнике принято [**2**], чтобы началом связанной со схватом системы координат служил его центр, а единичные векторы координатных осей соответствовали векторам $\bar{\mathbf{n}}$, $\bar{\mathbf{o}}$, $\bar{\mathbf{a}}$ (см. рисунок).

С помощью алгоритма Денавита – Хартенберга, однако, можно добиться лишь совмещения оси z_6 с осью симметрии схвата. Точки O_6 и C, вообще говоря, совпадать не будут.

Чтобы преодолеть возникшее затруднение, оставаясь в рамках формализма Денавита – Хартенберга, введём, наряду с системой отсчёта $\mathcal{E}_6$, ещё один экземпляр связанной со схватом системы отсчёта: $\mathcal{E}_7$. В этой СО будем пользоваться системой координат S_7^* с осями, удовлетворяющими сформулированному выше требованию; как обычно, через S_7 будет обозначаться текущая конфигурация S_7^* в $\mathcal{E}$.

При этом переход от S_6 к S_7 сведётся к винтовому перемещению вдоль оси z_6 (с постоянными параметрами θ_7 и d_7). Параметры же α_7 и a_7 будут равны нулю.

Для того, чтобы решить рассматриваемую задачу с использованием (в качестве инструментальной системы) программного комплекса **UM**, достаточно сообщить комплексу, что число подвижных звеньев механизма равно шести, и описать средствами этого комплекса характер и параметры кинематических соединений (а также задать зависимость обобщённых координат q_j от времени).

Конечно, чтобы увидеть движение механизма на экране, надо ещё описать звенья как геометрические тела (это обсуждается в следующем параграфе).

Формальное описание вращательных и поступательных сочленений средствами комплекса **UM** основано на понятии "*нормального шарнира*". В терминологии, принятой в комплексе **UM**, под нормальным шарниром понимается [**19**] кинематическое соединение, математическая модель которого сводится к указанию последовательности *элементарных преобразований* (*ЭП*), определяющих переход от системы координат S_{j-1} к системе S_j.

Каждое из элементарных преобразований есть либо сдвиг, либо поворот относительно одной из координатных осей и характеризуется вектором $\bar{\mathbf{e}} = \text{const}$ и параметром s. Различаются такие типы ЭП: **tc**, **tt**, **tv**, **rc**, **rt**, **rv**; здесь первая буква обозначает сдвиг (**t**) или поворот (**r**), а вторая показывает, что параметр s является константой (**c**), известной функцией времени (**t**), или переменной величиной (**v**), т.е. одной из локальных координат в шарнире.

При использовании формализма Денавита – Хартенберга каждое из рассматриваемых кинематических соединений моделируется как нормальный шарнир, определяемый цепочкой из четырёх ЭП, параметры которых совпадают с параметрами Денавита – Хартенберга. Задать значения этих параметров можно либо в диалоговом режиме, либо в текстовом файле **input.dat**.

Например, если *1*-е сочленение манипулятора является поступательным, то при просмотре описания нормального шарнира в файле **input.dat** увидим:

```
with joint1; type=comm; bd1=0; bd2=1;
with et; type=rc; ez=1; s0=<θ1>;
with et; type=tt; ez=1;
     s(t)=q1(t);
with et; type=rc; ex=1; s0=<α1>;
with et; type=tc; ex=<a1>;
```

(вместо взятых в угловые скобки обозначений параметров θ_1, α_1, a_1 нужно поставить их численные значения либо имена соответствующих символических параметров). Смысл записи вполне очевиден.

Наличие ЭП типа **tt** или **rt** подразумевает, что пользователь должен подготовить файл **functn.pas**, содержащий написанные на языке Турбо Паскаль тексты подпрограмм (эти подпрограммы в данном примере носят имена **q1**, **...**, **q6**), которые отвечают за вычисление функций времени $q_j(t)$, $\dot{q}_j(t)$ и $\ddot{q}_j(t)$.

Приведем текст подпрограммы **q1** (остальные устроены аналогично):

```
function q1(t : real_; drv1 : integer) : real_;
begin
case drv1 of
  0: begin
       q1:=a1*sin(n1*t)+b1;
     end;
  1: begin
       q1:=a1*n1*cos(n1*t);
     end;
  2: begin
       q1:=-a1*n1*n1*sin(n1*t);
     end;
 end;
end;
```

После того, как характер и параметры кинематических соединений будут описаны, а зависимость обобщённых координат от времени задана, Вы можете запустить задачу на решение. Комплекс **UM** осуществит автоматический вывод уравнений движения робота и обеспечит компьютерную визуализацию его движения.

Решая прямую задачу кинематики при различных значениях параметров, характеризующих изменение координат в сочленениях, Вы получите достаточно полное впечатление о кинематических возможностях своих манипуляторов. После этого можно перейти к решению более сложных задач.

2.2. Программный комплекс UM (“Универсальный механизм”)

Рассмотрим типичные действия при работе с программным комплексом **UM**.

Для решения прямой задачи кинематики следует, войдя в среду комплекса **UM** и задав имя своей задачи, при помощи графического препроцессора комплекса последовательно: 1) ввести числа тел и шарниров (оба числа в данной задаче равны шести); 2) связать с каждым звеном графический объект, который строится из стандартных графических элементов (параллелепипеды, призмы, цилиндры, конусы и т.п.) с указанием размеров и цветов каждого; 3) ввести описания шарниров вместе с номерами соединяемых ими тел; 4) задать первоначальные значения символических параметров. Результатами работы графического препроцессора является ряд файлов, в т.ч. текстовые файлы **inpgraph.dat** (информация о графических объектах), **input.dat** (основные параметры модели) и **constfil.pas** (таблица символических параметров с их начальными значениями).

Содержимое данных файлов часто бывает удобно корректировать непосредственно (с помощью любого текстового редактора).

Завершив графический ввод, надлежит вызвать в среде комплекса **UM** его компоненты, отвечающие за синтез уравнений движения и их оптимизацию. Результатом их работы будет набор программных файлов на языке Турбо Паскаль. После обработки этих файлов компилятором Вы вправе перейти к численному интегрированию уравнений модели. На этом этапе можно: выбирать метод интегрирования и его параметры; изменять значения символических параметров; открывать анимационные окна, служащие для визуализации процесса движения; открывать графические окна, в которых комплекс будет строить графики различных величин; записывать результаты интегрирования в выходные файлы для их последующего анализа.

Рассмотрим теперь последовательно один сеанс работы с комплексом “Универсальный механизм”.

Для запуска комплекса надо войти в директорию **UM** и набрать его имя – **um** – в командной строке.

Если комплекс не запускается, следует проверить, чтобы в конфигурационном файле **autoexec.bat** были правильно указаны пути для комплекса **UM** и компилятора с Паскаля, например: **C:\UM;C:\BP\BIN;**.

Предположим, что комплекс **UM** запустился успешно. Тогда на экране монитора на общем синем фоне появятся две строки белого цвета: одна вверху и одна внизу. В верхней строке будет написано:

Control Preprocessor Equation Postprocessor Edit

В нижней строке красным цветом будут выделены обозначения функциональных клавиш (здесь они даны подчёркиванием), а рядом будет указано название того или иного управляющего действия:

F3 Input F4 Name of Task F10 Menu Alt-X Quit JULA

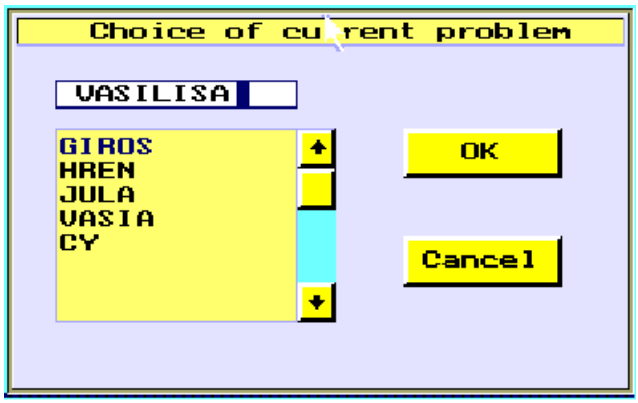
Последнее имя (**JULA**) в этом списке – имя задачи, которая в данный момент является активной.

Если Вы работаете с комплексом впервые, то имя задачи надлежит дать сразу (изменить его в дальнейшем достаточно сложно). Выясним, как это можно сделать.

Во-первых, надо воспользоваться функциональной клавишей **F10**. В результате окно пункта меню **Control** будет подсвечено синим цветом и станет активным. Станут активными и клавиши управления курсором ←, ↑, ↓, →. Последующее нажатие клавиши ↓ приведёт к появлению подменю, в котором активным окажется пункт **Name of Task**.

Заметим, что все перечисленные операции можно было бы реализовать и с помощью мыши.

Нажмите клавишу **Enter**. В командной строке, которая будет при этом высвечена в виде небольшой белой полоски, можно набрать имя задачи (ниже – в окне прокрутки – будут видны имена других задач, с которыми комплекс работал ранее). Если, как показано на рисунке, набрать в качестве имени своей задачи **VASILISA**, то тут же последует первое замечание: это имя содержит более семи символов (что при работе с комплексом **UM** недопустимо).



Итак, имя задачи не должно содержать более семи символов. Кроме того, оно не должно совпадать со стандартными обозначениями функций или операций; например, имя **AND** (как сокращение от имени **ANDREY**) недопустимо, так как оно совпадает с обозначением операции конъюнкции в языке Турбо Паскаль.

В ответ на полученное замечание нажмите клавишу **Esc** и наберите новое имя, содержащее уже меньшее число символов. Допустим, Вы набрали **VASIA**. Комплекс **UM** тут же отреагирует предупреждением: **New Task!** Поскольку задача – действительно новая, то нажмите в ответ на это клавишу **Enter**. В правом нижнем углу экрана тогда появится выбранное имя задачи (в нашем примере – **VASIA**).

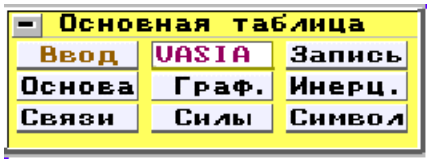
Если возникла необходимость, можно тут же прервать работу с программным комплексом (для чего достаточно ввести команду **Alt-X**). При этом Вы можете быть уверены, что введённое Вами имя задачи комплекс **UM** запомнил (при повторном запуске комплекса оно, являясь теперь именем активного задания, будет находиться на своём месте – в правом нижнем углу экрана).

Но, скорее всего, Ваша цель – решение конкретной задачи. Поэтому разумнее всего сразу же, засучив рукава, приступить к творению

нию своего неповторимого детища. Для этого Вам следует выйти на очередной пункт основного меню комплекса: **Preprocessor**.

В подменю этого пункта – два подпункта, но в рассматриваемой здесь версии **UM** задействован только один: **Object input**.

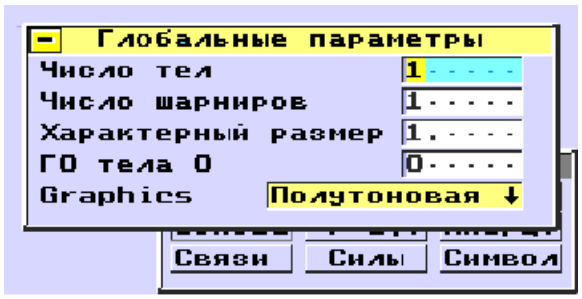
Нажав клавишу **Enter**, начните работу с программой графического ввода. Слева на экране будет видно большое синее окно (рабочий стол для творения), в центре которого изображена прямоугольная система координат *xyz*. Справа в нижнем углу – основное меню программы ввода с названием **Основная таблица**. При выходе на это меню подсвеченным (активным) является пункт **Ввод**.



Нажимать **Enter** не торопитесь (при обращении к пункту **Ввод** программа читает информацию из файлов **input.dat** и **inpgraph.dat**, но Вы ведь ещё ничего не успели сотворить), иначе получите новое замечание. Оно появится в красной полосе внизу экрана и будет содержать код ошибки с поясняющей надписью: **не найден файл inpgraph.dat**.

Выйти из создавшегося положения удастся не сразу. Придётся нажать клавишу **Esc** и затем правильно отвечать на вопросы комплекса **UM**. Поэтому начинать работу с пункта **Ввод** будете при втором и последующих обращениях к программе графического ввода, а в первый раз следует обратиться к другому пункту меню.

Выберем в меню программы ввода пункт **Основа**. Нажмем клавишу **Enter**. На экране появится подменю **Глобальные параметры** с пятью полями. Для шестизвенного манипулятора следует в качестве значений параметров **Число тел** и **Число шарниров** (в обозначениях главы 1 это – *n* и *m*) ввести число *6*. Число *1* в поле **Характерный размер** без особой нужды менять не следует (если же его заменить на меньшее число, то изображение Вашего объекта станет увеличенным, а замена единицы на большее число приведёт к уменьшению изображения).



А вот стоящее в поле **ГО тела 0** число *0* следует обязательно исправить на *1*! "ГО" означает: "графический объект", т.е. геометрическое тело, сопоставляемое данному звену. Нумерация графических объектов в комплексе **UM** ведётся от единицы, так что если Вы хотите увидеть "тело *0*" (т.е. основание) на этапе визуализации движения, следует создать графический объект и для него.

В принципе нумерацию звеньев механизма и графических объектов можно вести независимо (в том случае, когда два тела имеют одинаковую форму, размер и раскраску, им вполне разумно сопоставить один и тот же графический объект). Но в обычных ситуациях рекомендуется пользоваться простым правилом: "номер ГО – это номер звена плюс *1*".

В поле **Graphics** указан тип графики: **Полутоновая**. Если

выйти на подменю, соответствующее этому пункту, то Вам будут предоставлены следующие возможности: **Каркасная**, **Полутоновая**, **Многоцветная**, **Контурная**. Мы рекомендуем при первом обращении к программе графического ввода выбрать **Многоцветная** (после того, как изображения всех графических объектов будут созданы, тип графики можно будет изменить).

Отметим, что при работе с меню программы графического ввода в большинстве случаев можно воспользоваться контекстной помощью, для чего достаточно нажать клавишу **F1**.

Закончив работу с пунктом **Основа**, вернёмся (нажатием клавиши **Esc**) к меню **Основная таблица**, в котором выберем пункт **Граф.** Нажмём **Enter**; появится меню **Графические объекты**.

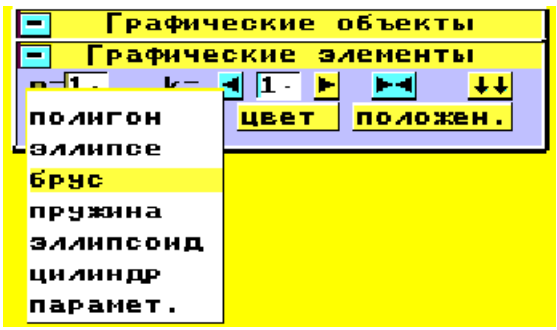
В этом меню Вы увидите поле **n=** (в окне при этом поле указывается общее число графических объектов, а сперва стоит 1), поле **i=** (номер текущего ГО, с которым Вы работаете), а также кнопку с двумя треугольниками и кнопку с двумя стрелками.

В окне при поле **i=** при первом входе тоже стоит число 1. Для увеличения или уменьшения номера текущего ГО служат две маленькие кнопки по обе стороны от окошка с этим номером (аналог скроллинга). Если Вы ошибочно указали номер несуществующего ГО, нажмите кнопку с двумя треугольниками (“забой”).

Кнопка с двумя стрелками означает переход на расширение текущего меню. Нажмём её; появится подменю **Графические элементы**.

Теперь предлагается приступить к вводу данных об элементах, образующих данный графический объект (их общее число показано в поле **n=**). В окне при поле **k=** указан номер текущего графического элемента. Кнопки с двумя треугольниками и с двумя стрелками имеют то же назначение, что и в предыдущем меню. Ниже располагаются три дополнительных окна.

При выходе на первое из них появляется окно прокрутки, в котором указаны типы стандартных графических элементов (заметим, что **полигон** – это плоский многоугольник, а **парамет.** – это поверхность, задаваемая параметрическим способом). Один из пунктов этого перечня надлежит выбрать.



Предположим, что свой выбор Вы остановили на пункте **цилиндр**. Отметим сразу же, что это название достаточно условно, и в действительности Вы сможете создать не только цилиндр, но также конус (обычный или усечённый); а если учесть, что объекты с криволинейными поверхностями аппроксимируются многогранниками, то речь идёт соответственно о призме или о пирамиде (обычной или усечённой).

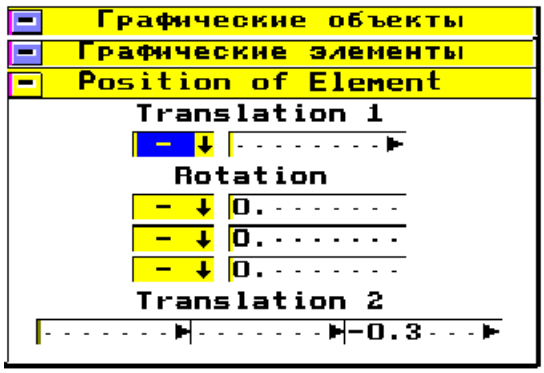
Перейдём теперь на кнопку расширения меню. Вы увидите перечень числовых параметров, которые определяют форму и размеры Вашего “цилиндра”. Так, **r1** и **r2** представляют собой соответственно радиусы нижнего и верхнего оснований; если они одинаковы, как это показано на рисунке, то получается цилиндр, если не равны – конус (усечённый, если оба радиуса отличны от нуля, и обычный, если одно из этих числе равно нулю). Высота тела будет равна **h**.

Заметим, что в основаниях могут лежать не окружности, а дуги окружностей (так что можно получить, например, полцилиндра). В такой ситуации в поле **Arc** указываются (в градусах) начальное и конечное значения угла, задающего дугу.

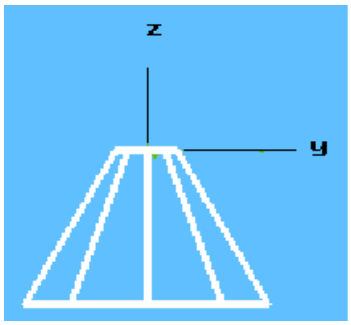
При изменении угла от 0° до 0° окружность считается полной (т.е. угол фактически изменяется от 0° до 360°).

Наконец, параметры **nc** и **nl** задают соответственно число линий на боковой поверхности (т.е. число боковых граней аппроксимирующей призмы или пирамиды) и число линий, параллельных основаниям (так, при **nl**, равном 2, рисуются только сами основания, а при **nl**, равном 3, – ещё и промежуточная окружность).

После выхода из расширения меню на экране появится каркасное изображение “цилиндра”. Если его положение в системе координат, связанной с данным графическим объектом, Вас не устраивает, то обратитесь к пункту **положен.** в меню **Графические элементы**.



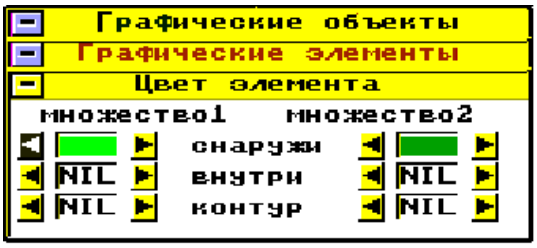
При обращении к этому пункту на экране возникнет очередное подменю, с помощью которого можно задать набор последовательных преобразований, которым будет подвергнуто тело (среди них – сдвиг на заданное значение вдоль какой-либо координатной оси, три поворота вокруг координатных осей и ещё один сдвиг на постоянный вектор.



Например, если **r1** задаётся большим, чем **r2**, а **nc** принимается равным 8, то получается восьмигранная усечённая пирамида, нижнее основание которой будет располагаться в плоскости xy. Если её положение Вас не устраивает (Вы хотели бы совместить с этой плоскостью верхнее основание), то достаточно просто сдвинуть пирамиду вдоль её оси симметрии (ось z) на высоту h. Для этого можно в последнем окне пункта **Translation 2** набрать то же самое число, которое Вы задавали в качестве **h** при вводе числовых параметров графического элемента

(причём со знаком минус: пирамиду надо опустить, а не приподнять).

Следующее, что необходимо сделать – это задать цвета для Вашей пирамиды. Для этого обратимся к пункту **цвет** в меню **Графические элементы**. Комплекс **UM** предлагает Вам задать два набора цветов, которые при визуализации граней элемента будут чередоваться. Для элемента типа **цилиндр** достаточно задать лишь цвета в строке **снаружи**. Поэтому в каждом из двух окон в данной строке необходимо, используя для прокрутки кнопки с треугольниками, установить требуемый цвет.



Рекомендуется выбирать близкие оттенки цвета (более светлый для первого набора и более тёмный – для второго). Такой подбор цветов позволит получить в дальнейшем более реалистичное изображение.

После выбора цветов работа с данным графическим элементом заканчивается, и следует перейти к следующему элементу данного графического объекта (если он есть). Завершив ввод данных для всех элементов данного ГО, возвращайтесь к меню **Графические объекты**; теперь можно приняться за следующий объект…

Вернувшись в конце концов к основному меню программы графического ввода, перейдём к пункту **Инерц.** Поскольку решаемая задача чисто кинематическая, а комплекс рассчитан на решение динамических задач, то нам достаточно было бы войти в меню этого пункта и сразу же выйти, нажав **Esc** (тогда все инерционные параметры окажутся – по умолчанию – нулевыми). Однако в действительности одну вещь в этом пункте сделать всё-таки нужно, а именно: для каждого подвижного звена механизма указать номер соответствующего ему ГО.

Так, в соответствии с уже упоминавшимся правилом, для звена с номером *1* (см. рисунок) следует задать номер ГО, равный *2*.

Перейдём теперь к пункту **Связи** основного меню. После нажатия клавиши **Enter** на экране появится меню **Ввод шарниров**. Назначение кнопок в его верхней строке Вам уже знакомо, но теперь в поле **j=** находится номер текущего шарнира (т.е. кинематического соединения).

В первую очередь рекомендуется указать, какие тела соединяет данный шарнир (в соответствии с принятой в главе 1 нумерацией звеньев и кинематических соединений первый шарнир осуществляет связь между *0*-м и *1*-м телами, так что в левом нижнем окошке первоначально стоявшую там цифру *0* заменять не следует, а в следующем окне надо заменить *0* на *1*, что и показано на рисунке).

Информация в полях средней строки (тип шарнира **Нормальный**, признак шарнира **Внутренний**) нас вполне устраивает (кстати, все шарниры, рассматриваемые в данном пособии, являются в терминологии комплекса **UM** "внутренними"), так что менять здесь ничего не надо. Перейдём на расширение данного меню.

На экране появляется подменю **Нормальный шарнир**. Поле **n=** указывает номер текущего шарнира, а поле **i=** – номер текущего ЭП (напоминаем, что все шарниры в наших задачах определяются цепочками из четырёх ЭП).

Предположим, что первый шарнир, описание которого мы сейчас задаем, моделирует поступательную кинематическую пару. Тогда при **i=1** мы выберем ЭП типа **rc** (тип задаётся в окне прокрутки справа), а в качестве вектора $\bar{\mathbf{e}}$ для него зададим единичный вектор оси z (это означает, что два первых поля с компонентами вектора $\bar{\mathbf{e}}$ можно оставить пустыми, что комплекс будет трактовать как нулевые значения, а в третье поле введём число *1*). Обратившись к пункту **s0,v0**, наберём в поле **s0** значение θ_1 (в градусах).

При **i=2** типом ЭП будет уже **tt** (с тем же вектором $\bar{\mathbf{e}}$). Поскольку соответствующий параметр Денавита – Хартенберга представляет собой заданную функцию времени $q_1(t)$, то вместо пункта **s0,v0** обратимся к пункту **s(t),f** и в соответствующем поле этого пункта наберём **q1(t)** (напоминаем, что функции **q1** отвечает подпрограмма на Паскале, текст которой приводился в параграфе 1).

При **i=3** и **i=4** в качестве типа ЭП задаются **rc** и **tc**. При этом поворот на угол α_1 задаётся так же, как и поворот на θ_1 (только в качестве вектора $\bar{\mathbf{e}}$ выступает единичный вектор оси x), а вот для задания смещения вдоль оси x на a_1 следует непосредственно задать вектор смещения (т.е. в первом из полей с компонентами вектора $\bar{\mathbf{e}}$ ввести значение a_1, а остальные поля оставить пустыми).

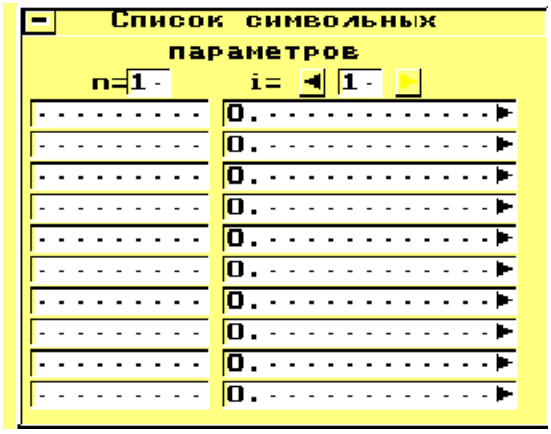
Данные для остальных шарниров вводятся аналогично.

Обратите внимание: при вводе информации о шарнирах Вы фактически дали комплексу **UM** обязательство разработать подпрограммы **q1**, **...**, **q6**. В них будут использоваться символические параметры **a1**, **b1**, **n1**, **a2**, **...**, **n6**. Поэтому нужно сообщить комплексу о существовании этих параметров и присвоить им некоторые значения (которые потом можно будет оперативно менять).

Это делается в пункте **Символ** основного меню программы ввода.

При выходе на этот пункт и нажатии клавиши **Enter** на экране появляется меню **Список символьных параметров**. Список параметров может быть достаточно длинным и состоит из набора таблиц, каждая из которых включает по десять параметров (последняя таблица может быть заполнена частично). В поле **n=** программа указывает общее число таблиц, а в поле **i=** – номер текущей таблицы. В левой части каждой строки таблицы набирается имя параметра, а в правой – его численное значение. Для перехода к новой таблице дос-

таточно воспользоваться кнопкой с треугольником справа от номера.



После завершения ввода символьных параметров работу с программой графического ввода можно считать завершённой. Теперь обратимся к пункту **Запись** основного меню программы ввода, чтобы сохранить результаты своей деятельности. Заметим, что при правильном вводе данных должно появиться сообщение **Ошибки ввода не найдены**. Если вместо него Вы получили сообщение о том, что какие-то данные введены не полностью, следует продолжить работу с программой ввода.

Выход из программы ввода производится нажатием клавиши **Esc** и утвердительным ответом на вопрос: **Хотите ли Вы покинуть программу ввода?**

После выхода из программы ввода мы вновь окажемся в основном меню комплекса **UM**. Перейдём к пункту **Equation**. Здесь сначала выполняется вывод уравнений движения, затем – их оптимизация. На этапе оптимизации комплекс может задать Вам вопрос:

Файл functn.pas уже существует. Переписать? (y/n)

Если Вы впервые работаете с данной задачей, то следует ответить **y**. Тогда комплекс создаст файл с именем **functn.pas**, в котором разместит описания Ваших подпрограмм (в нашем примере они носят имена **q1**, **...**, **q6**), но в этих программах операторы присваивания для **q1** и т.п. будут отсутствовать (чуть позже Вы их добавите).

Важное замечание: при повторной работе с задачей на указанный вопрос следует отвечать **n** (иначе все Ваши труды по корректировке файла **functn.pas** пропадут!).

После завершения оптимизации (если Вы работаете с данной задачей впервые) следует выйти на пункт **Edit** основного меню. Здесь Вы можете корректировать (с помощью текстового редактора интегрированной среды компилятора **Borland Pascal**) различные текстовые файлы, относящиеся к Вашей задаче. В данном случае речь идёт о корректировке файла **functn.pas**.

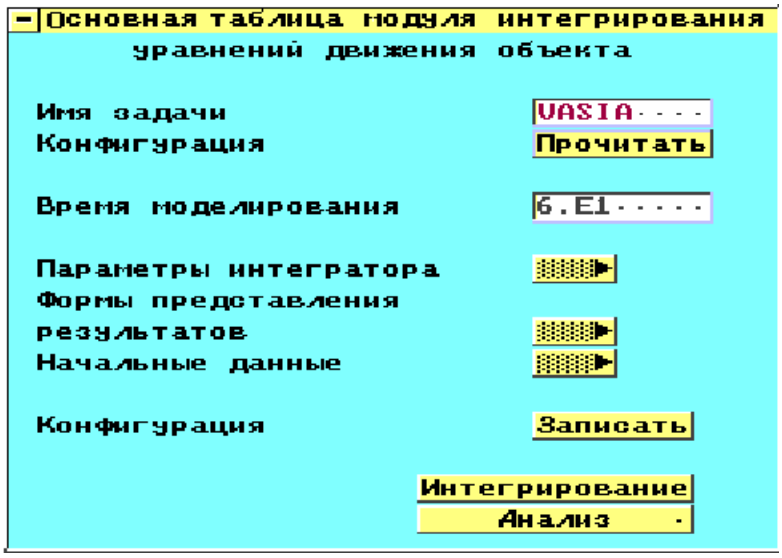
Для этого в меню пункта **Edit** выберем пункт **Equations**, а в нем – подпункт **Functions**. Корректировка осуществляется обычным образом. При этом требуется быть внимательными и не ошибиться при написании первой и второй производных от функций времени $q_j(t)$ (обнаружить сделанные здесь ошибки будет весьма трудно!).

Выйти из текстового редактора можно по команде **Alt-X**.

Теперь выйдем на пункт **Postprocessor** основного меню, в котором выберем подпункт **Integration**, а в нём – подпункт **With compiling**. При этом запустится компилятор с Паскаля. Его рабо-

та продлится несколько секунд, после чего на экране возникнет заставка с надписью **Интегратор**. Нажмите **Enter**, и в ответ появится основная таблица модуля интегрирования.

При первом обращении к модулю интегрирования следует в поле **Время моделирования** задать значение t_{fin} (например, как на рисунке, 60 с), а затем перейти к пункту **Форма представления результатов** и нажать **Enter**. В меню, которое появится на экране, активным (выделенным жёлтым цветом) будет подпункт **Режим модификации окон**. Теперь Ваши действия таковы: надо, нажав **Enter**, выйти в подменю этого подпункта (в левой половине экрана появится пустое окно синего цвета) и сразу же (нажав два раза **Esc**) вернуться к основной таблице.

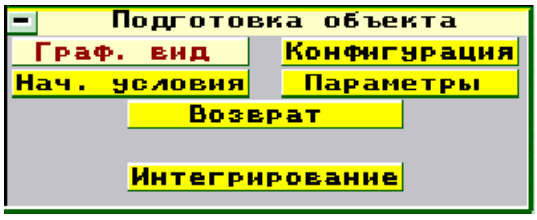


Смысл этих действий: Вы создаёте анимационное окно, в котором комплекс в дальнейшем будет показывать движение механизма.

Теперь нажмите на кнопку **Записать**. При этом текущая конфигурация модуля интегрирования вместе с произведённой только что настройкой будет сохранена на жёстком диске.

При последующих обращениях к модулю интегрирования этой задачи следует нажать на кнопку **Прочитать** (файл с конфигурацией будет при этом прочитан с диска), а повторно задавать значение t_{fin} и открывать анимационное окно уже не потребуется.

Перейдём теперь к пункту **Интегрирование**. На экране при этом появятся окно анимации с изображением манипулятора и таблица **Подготовка объекта**. При необходимости можно обратиться к подпункту **Графический вид**, что позволяет изменить масштаб или ракурс изображения (последнее означает, что воображаемая камера, через которую Вы смотрите на манипулятор, разворачивается вокруг какой-либо оси или сдвигается).



Шаг, задающий вращение вокруг координатной оси, указывается в градусах (значение, используемое по умолчанию, равно 10°). Поэтому, если потребовалось повернуть камеру в положительном направлении вращения (т.е. против хода часовой стрелки) на 30° вокруг оси x,

то достаточно трижды нажать на кнопку с надписью **x**. Если же требуется повернуть её на такое же количество градусов в отрицательном направлении, то лучше войти в поле **Шаг** и набрать в нём –30°. Шаг, задающий сдвиг, по умолчанию равен 0,1 м (впрочем, выбор единиц измерения длины, времени и массы при компьютерном моделировании носит во многом условный характер).

В подпункте **Параметры** можно изменить значения символьных параметров (тех самых, которые впервые задавались в программе графического ввода и могли быть затем изменены корректировкой файла **constfil.pas** с помощью текстового редактора). Перекомпиляция модуля интегрирования при этом не требуется.

После того, как всё это сделано, наступает решающий момент: обратившись к подпункту **Интегрирование** в таблице **Подготовка объекта**, Вы приступаете к численному интегрированию уравнений движения робота, созданного усилиями Вашего интеллекта. При этом на экране можно будет наблюдать его движение.

Применительно к рассматриваемой сейчас задаче это значит, что звенья робота будут совершать колебательные движения с теми амплитудами и частотами, которые были указаны в качестве значений символьных параметров.

После окончания процесса интегрирования его можно повторить, для чего вовсе не обязательно повторять процесс компиляции. Достаточно лишь нажать клавишу **Esc** и вернуться к основной таблице модуля интегрирования (при желании теперь можно увеличить время моделирования, изменить значения параметров интегратора, значения некоторых символьных параметров и т.д.).

В ходе интегрирования можно управлять камерой. Так, для того, чтобы приблизить (или отодвинуть) изображение, достаточно нажать на клавишу **+** (или **–**) в правой части клавиатуры. С помощью клавиш ←, ↑, ↓, → можно двигать изображение в плоскости экрана. Для поворота в положительном направлении вокруг координатных осей используются клавиши с буквами **X**, **Y**, **Z**, а для поворота в противоположную сторону – те же клавиши, но при удерживаемой в нажатом положении клавише **Shift**.

Нажатие клавиши **W** приостанавливает процесс интегрирования, а если нажать её ещё раз, то в верхней части окна анимации появится строка с клавишами управления изображением. Первая из них служит для закрытия окна анимации. Следующие три (с круговыми стрелками) – для выполнения поворотов вокруг соответствующих осей. Последующие две (со взаимно противоположными стрелками) – для сдвига

изображения. Две следующие клавиши позволяют приблизить изображение или отодвинуть его. Клавиша с буквой **R** (**Restore**) служит для восстановления начального ракурса изображения. Наконец, последняя клавиша позволяет вывести в другом окне графики координат и скоростей.

Приостановить процесс интегрирования (сделать паузу) можно нажатием какой-либо клавиши (например, пробела). Для продолжения интегрирования следует нажать клавишу **Enter**. Если потребовалось прервать процесс досрочно, то достаточно нажать клавишу **Q** (**Quit**).

Мы рекомендуем поэкспериментировать с Вашей компьютерной моделью, меняя значения символьных параметров, и получить тем самым достаточно полное представление о кинематических возможностях Вашего манипулятора. Это поможет Вам решить более сложную задачу, к рассмотрению которой мы сейчас и переходим.

Глава 3. Решение обратной задачи кинематического анализа шестизвенного манипулятора

Рассмотрим один из вариантов постановки задачи об управлении (на кинематическом уровне) движением шестизвенного манипулятора. Выполнение её предусматривает многократное решение обратной задачи о скоростях.

3.1. Постановка обратной задачи кинематики

Задача о роботе-бармене. Манипулятор, конфигурация которого при $t = 0$ известна, начинает движение из состояния покоя. Требуется за заданное время t_{fin} подвести схват к переносимому объекту, захватить его и переставить на новое место. Радиусы-векторы $\overline{\mathbf{r}}_A$ и $\overline{\mathbf{r}}_B$ начального (точка A) и конечного (точка B) положений центра объекта, а также максимальная высота H подъёма объекта даны (при этом $\overline{\mathbf{r}}_A = \overline{\mathbf{r}}_B$). Необходимо найти, как в процессе движения изменяются координаты в сочленениях q_j и скорости в сочленениях $\dot{q}_j$.

Предполагаем, что ось Oz направлена вверх. Пусть $t_{\text{fin}} = 40$ с.

Обсудим общий порядок решения задачи.

Сначала выясним, где в начальный момент находится центр схвата и какова при этом ориентация схвата (для этого достаточно решить при $t = 0$ по рекуррентным формулам (14) прямую задачу геометрии движения и найти верзор схвата, а тем самым – радиус-вектор $\overline{\mathbf{r}}_C$ текущего положения центра схвата и оператор ориентации $\overline{\mathbf{\Gamma}}_7$).

Теперь надо запрограммировать движение схвата: обеспечить, чтобы в некоторый момент времени точка C совпала с A, а вектор подхода $\bar{\mathbf{a}}$ оказался направленным вертикально вниз, после чего (не меняя уже значения вектора $\bar{\mathbf{a}}$ – чтобы не расплескать содержимое!) изменять $\bar{\mathbf{r}}_C$ так, чтобы в конце концов точка C совпала с B (причём в процессе этого перемещения координата z_C должна сначала увеличиться на H, а затем на столько же уменьшиться).

Если программный закон движения схвата известен, то задача вычисления координат в сочленениях как функций времени t как раз и представляет собой обратную задачу кинематического анализа манипулятора. Решение её будет обсуждаться далее.

Сейчас же рассмотрим программирование движения схвата.

Далее мы воспользуемся понятием *элементарной транспортной операции* [16]. Такую операцию задают по отдельности для каждой координаты. Суть её заключается в плавном изменении значения этой координаты от одного заданного значения до другого, причём в начальный и конечный моменты времени первая и вторая производные данной координаты должны равняться нулю.

В общем случае элементарная транспортная операция будет состоять из шести этапов. Моменты времени, отвечающие смене этапов, обозначим $\tau_0, \ldots, \tau_6$.

(Строго говоря, это справедливо для первой по счёту элементарной транспортной операции; для второй границами этапов будут $\tau_6, \ldots, \tau_{12}$, и т.д.).

Например, пусть η – одна из функций $v_{Cx}(t)$, $v_{Cy}(t)$, $v_{Cz}(t)$, $\omega_x(t)$, $\omega_y(t)$, $\omega_z(t)$.

Тогда примем, что: на этапе I $\eta = 0$; на этапе II η ускоренно возрастает; на этапе III η тоже возрастает, но замедленно; на этапе IV $\eta = \mathrm{const}$; на этапе V η ускоренно убывает; на этапе VI η убывает, но замедленно.

Пусть $T_N \equiv \tau_1 - \tau_0$ – длительность этапа I, $T_M \equiv \tau_6 - \tau_1$ – длительность пяти остальных, а $T_C \equiv \tau_4 - \tau_3$ – длительность этапа IV. Потребуем для определённости, чтобы длины этапов II, III, V и VI были одинаковыми и равными $T_* = (T_M - T_C)/4$.

Требуем, далее, чтобы функция $\eta(t)$ была кусочно-квадратичной и принадлежала классу C^1 (так что координаты будут функциями класса C^2). Пусть i – номер этапа элементарной транспортной операции ($i = 0, \ldots, 5$).

Введём вспомогательную переменную $s = \dfrac{t - \tau_i}{\tau_{i+1} - \tau_i} \in [0, 1]$.

Обозначим $S = 1 - s$. В этом случае для $\eta(t)$ получаем следующие выражения на различных этапах данной элементарной транспортной операции: 0 при $i = 0$; as^2 при $i = 1$; $2a - aS^2$ при $i = 2$; $2a$ при $i = 3$; $2a - as^2$ при $i = 4$; aS^2 при $i = 5$.

Выбор значения a рассмотрим на примере. Пусть $\eta = V_{Cx}$; положим

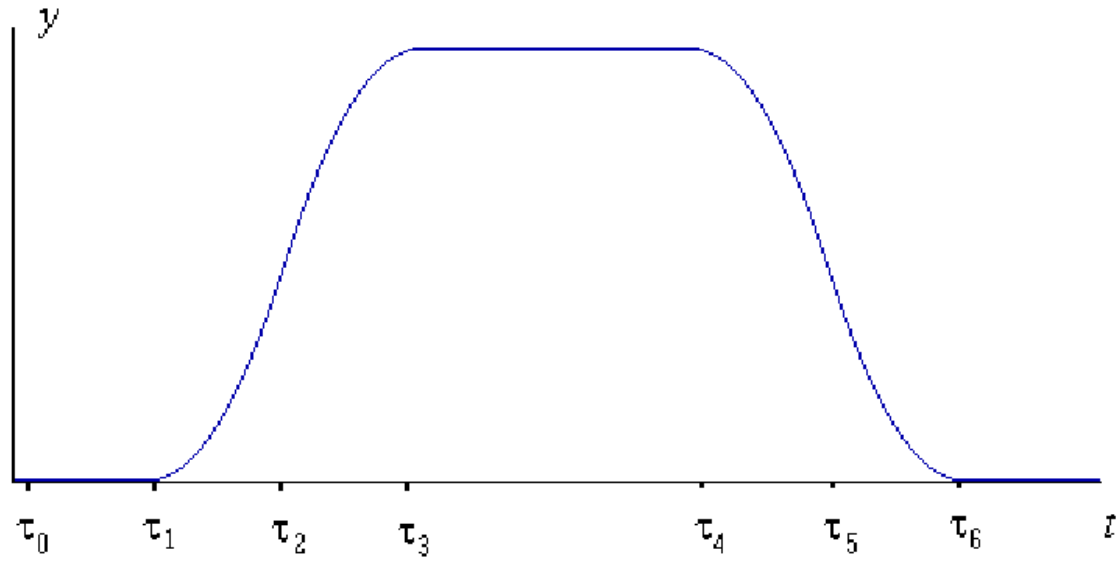
$$\xi = \int_0^t \eta \, dt \equiv x_C \ .$$

Изменение x_C за время моделирования реализуется с помощью двух элементарных транспортных операций, для которых соответственно

$$\Delta\xi = \int_{\tau_0}^{\tau_6} \eta \, dt \equiv x_A - x_C(0) \quad \text{и} \quad \Delta\xi = \int_{\tau_6}^{\tau_{12}} \eta \, dt \equiv x_B - x_A \ ,$$

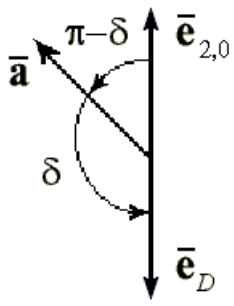
откуда, вычисляя интегралы, находим: $a = \dfrac{\Delta\xi}{T_M + T_C}$.

Тем самым функция $\eta(t)$ полностью определена. Её график выглядит следующим образом:



Для случая $\eta = V_{Cy}$ всё делается аналогично. Для более сложного случая $\eta = V_{Cz}$ (когда имеется три элементарные транспортные операции), константы $\Delta\xi$ соответственно будут равны $z_A - z_C(0)$, H и $-H$.

Что касается случая $\eta = \omega_x$, то требуется, чтобы к моменту захвата объекта вектор подхода $\bar{\mathbf{a}}$ был направлен вниз, оставаясь таким и далее – до конца движения (при $t = 0$ этот вектор произволен). Пусть $\bar{\mathbf{e}}_D$ – орт, направленный вниз ($\bar{\mathbf{e}}_D = -\bar{\mathbf{e}}_{2,0}$), $\bar{\mathbf{a}}^\circ$ – начальное значение вектора подхода. Обозначим через δ текущий угол между ортами $\bar{\mathbf{a}}$ и $\bar{\mathbf{e}}_D$ (он равен $\arccos(-a_z)$), а через δ° – начальное значение этого угла. Вычислим единичный вектор $\bar{\mathbf{e}} \uparrow\uparrow [\bar{\mathbf{a}}^\circ, \bar{\mathbf{e}}_D]$ и потребуем, чтобы на отрезке времени, заканчивающемся захватом объекта, выполнялся поворот на угол δ° вокруг орта $\bar{\mathbf{e}}$, а потом (на стадии переноса объекта из точки A в точку B) ориентация схвата не менялась.



Всё это можно реализовать с помощью двух элементарных транспортных операций, для которых $\Delta\xi$ будут равны соответственно $\delta^{\circ}e_x$ и 0. Аналогичные формулы верны и при $\eta = \omega_y$ или $\eta = \omega_z$.

Итак, программное движение схвата построено.

Отметим, что функции, описывающие зависимость координат от времени, в действительности представляют собой кубические сплайны.

Рассмотрим теперь решение задачи о роботе-бармене с помощью типовой программы **kin_31.c** и программного комплекса **UM**.

3.2. Работа с типовой программой kin_31.c

Обсудим порядок выполнения задания практикума путем непосредственного программирования расчётных формул на языке Си.

Воспользуемся тем же подходом, что и в [**20**]: готовую программу, рассчитанную на решение конкретного варианта задания (условно – 31-го, откуда и название типовой программы), студенту предлагается приспособить для решения своего варианта, т.е. он свою собственную программу разрабатывает на основе типовой.

Для этого, прежде всего, скопируем в файл с новым именем содержимое файла **kin_31.c**.

Далее, используя любой текстовый редактор (например, интегрированную среду системы программирования Турбо Си), исправим в строках комментария, образующих в совокупности заглавие этого файла, название файла, дату его создания и фамилию разработчика.

После этого приступаем к корректировке описаний исходных данных, которые размещены в тексте программной единицы **main**.

Здесь задаётся следующая информация: 1) описание структуры и геометрии манипулятора и его начальной конфигурации, представленное массивами **Lam[6]** и **Miu[6]** структурных констант λ_j и μ_j и массивом **DH[7][4]** параметров Денавита – Хартенберга; 2) начальное и конечное положения центра объекта, т.е. заданные в массивах **rA[3]** и **rB[3]** координаты точек A и B; 3) максимальная высота H подъёма объекта при переносе его из точки A в точку B; 4) номер варианта – целая константа **var**; 5) массивы **durX[3*2]** и **durZ[3*3]** (содержимое этих двух массивов менять не следует).

Массив **durX[3*2]** содержит две тройки констант T_N, T_M, T_C, задающих длительности этапов элементарных транспортных операций для функций $v_{Cx}(t)$, $v_{Cy}(t)$, $\omega_x(t)$, $\omega_y(t)$, $\omega_z(t)$.

В массиве **durZ[3*3]** находятся аналогичные константы для функции $v_{Cz}(t)$.

После внесения необходимых исправлений следует выйти из текстового редактора и с помощью командного файла **pgm.bat** скомпи-

лировать программу. Теперь можно начинать моделирование.

Обсудим схему решения обратной задачи кинематического манипулятора, реализованную в программе **kin_31.c**.

Если закон движения схвата известен, то можно найти зависимость от времени угловой скорости $\overline{\omega}_7$ схвата и линейной скорости $\overline{\mathbf{v}}_C$ его центра.

Тем самым будет найдено и блочное представление кинематического винта схвата относительно полюса C:

$$\overline{\mathbf{U}}_7 \sim \overline{\mathbf{U}}_C \equiv \begin{pmatrix} \overline{\omega}_7 \\ \overline{\mathbf{v}}_C \end{pmatrix}.$$

После этого нетрудно получить и аналогичное представление относительно полюса O:

$$\overline{\mathbf{U}}_7 \sim \overline{\mathbf{U}}_O \equiv \begin{pmatrix} \overline{\omega}_7 \\ \overline{\mathbf{v}}_O \end{pmatrix}, \qquad \overline{\mathbf{v}}_O \equiv \overline{\mathbf{r}}_C \times \overline{\omega} + \overline{\mathbf{v}}_C . \tag{21}$$

Система линейных уравнений (19) в нашем случае имеет вид

$$U_7 = A_{\mathrm{K}} \dot{q} , \tag{22}$$

а столбец U_7 совпадает со столбцом компонент блочного вектора $\overline{\mathbf{U}}_O$.

Таким образом, столбец $\dot{q}$ скоростей в сочленениях можно вычислить, решив систему линейных уравнений (22) одним из численных методов; программа **kin_31.c** использует для этой цели *метод LU-разложения* [**21**].

С этой целью программа обращается к процедурам **LUdek** и **LUsol** из библиотеки **tmx.lib** (последняя разработана на кафедре теоретической механики МЭИ и содержит объектные модули, полученные в результате компиляции написанных на языке Си процедур, которые позволяют решать различные задачи вычислительной математики и выполняют некоторые сервисные функции).

Отметим, что процедура **LUdek** предназначена для разложения невырожденной матрицы A в произведение двух матриц: нижней унитреугольной L и верхней треугольной U; процедура **LUsol** – для вычисления решения системы линейных уравнений $Ax = b$ при условии, что такое разложение получено.

Для нахождения координат в сочленениях численно решается задача Коши для дифференциальных уравнений

$$\frac{\mathrm{d}q}{\mathrm{d}t} = \dot{q} \tag{23}$$

на отрезке времени $[0, t_{\mathrm{fin}}]$.

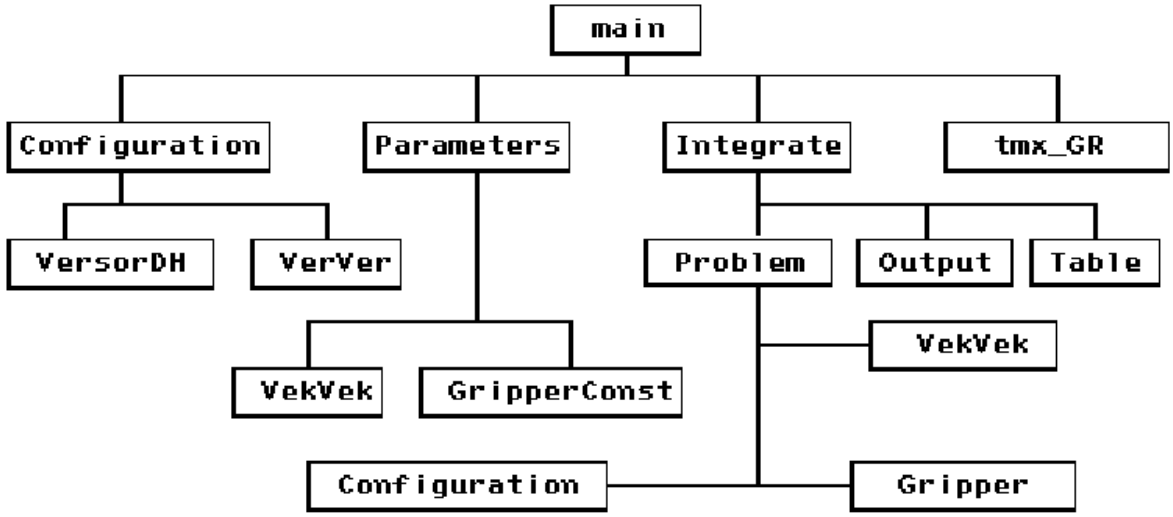
Заметим, что для удобства последующего анализа программа **kin_31.c** осуществляет вывод результатов в числовой и графической форме. В частности, после окончания интегрирования уравнений (23) на экран видеомонитора выводятся графики, показывающие изменение с течением времени компонент векторов $\overline{\mathbf{r}}_C$ и $\overline{\mathbf{a}}$ и величин q_j и $\dot{q}_j$.

Построение графиков обеспечивается вызовом сервисной проце-

дуры **tmx_GR** из библиотеки **tmx.lib**.

Рассмотрим структуру и состав типовой программы **kin_31.c**.

Прежде всего, приведём структурную схему этой программы:



В соответствии с приведённой схемой в состав типовой программы **kin_31.c** входят следующие программные единицы:

1) программная единица **main**, в которой задаются исходные данные и последовательно вызываются подпрограммы **Initialize**, **Parameters**, **Integrate**, а также (несколько раз) сервисная процедура **tmx_GR**;

2) подпрограмма **Initialize** (на схеме не показана), которая выводит на экран таблицу начальных значений параметров Денавита – Хартенберга, а затем осуществляет в массиве **DH** (который содержит значения этих параметров) переход от градусной меры углов к радианной и вызывает подпрограмму **Configuration** с целью расчёта начальной конфигурации манипулятора;

3) подпрограмма **Parameters**, которая вычисляет параметры начальной ориентации схвата, а после этого вызывает подпрограмму **GripperConst** и выводит вычисленные значения на экран;

4) подпрограмма **Integrate**, обеспечивающая численное решение задачи Коши для системы дифференциальных уравнений (23);

5) подпрограмма **Configuration**, отвечающая за решение прямой задачи о положениях манипулятора;

6) подпрограмма **GripperConst**, которая вычисляет константы, определяющие кусочно-полиномиальное представление используемых сплайнов (т.е. моменты времени τ_i и соответствующие значения a);

7) подпрограмма **Problem**, вызываемая из **Integrate** для нахождения правых частей системы (23) путём решения обратной задачи о скоростях;

8) подпрограмма **Gripper**, вычисляющая в текущий момент времени t программные значения компонент блочного вектора $\overline{\mathbf{U}}_C$;

9) подпрограмма **VersorDH**, вызываемая из **Configuration** с целью расчёта относительных верзоров $\overline{\mathbf{G}}_{ij}$ по формулам (10)–(12);

10) подпрограмма **VerVer**, обеспечивающая вычисление произведения двух верзоров;

11) подпрограмма **VekVek**, которая соответственно выполняет операцию векторного умножения;

12) подпрограмма **PrMat** (на схеме не показана), вызываемая из **Parameters** и обеспечивающая вывод на экран матриц и столбцов с соответствующими заголовками;

13) подпрограмма **Output**, вызываемая из **Integrate** и обеспечивающая сохранение в рабочих массивах очередной порции результатов интегрирования, а также вывод части полученных результатов на экран (речь идёт о компонентах векторов $\overline{\mathbf{r}}_C$ и $\overline{\mathbf{a}}$);

14) подпрограмма **Tables**, вызываемая на заключительном этапе работы **Integrate** и выводящая на экран последовательно таблицы значений переменных q_j и $\dot{q}_j$.

Остановимся немного подробнее на отдельных подпрограммах.

Подпрограмма **Parameters** вычисляет (и выводит на экран) векторы $\overline{\mathbf{r}}_C(0)$ и $\overline{\mathbf{a}}^\circ$, угол δ° и орт $\overline{\mathbf{e}}$ оси поворота схвата. Вычисляются также угол β (о нём несколько слов скажем ниже), и – с помощью подпрограммы **GripperConst** – моменты времени τ_i.

По поводу угла β: напомним, что на отрезке времени, заканчивающемся захватом объекта, изменение ориентации схвата сводится к повороту вокруг неподвижного орта $\overline{\mathbf{e}}$; на стадии же переноса объекта ориентация не меняется, а вектор $\overline{\mathbf{a}} \equiv \overline{\mathbf{e}}_{2,7}$ совпадает по направлению с вектором $\overline{\mathbf{e}}_D \equiv -\overline{\mathbf{e}}_{2,0}$.

Это означает, что в любой момент времени поворот вокруг орта $\overline{\mathbf{e}}$ на угол $\pi-\delta$ совмещает орт $\overline{\mathbf{e}}_{2,0}$ с ортом $\overline{\mathbf{e}}_{2,7}$. Для совмещения орта $\overline{\mathbf{e}}_{0,0}$ с ортом $\overline{\mathbf{n}} \equiv \overline{\mathbf{e}}_{0,7}$ достаточно выполнить поворот на некоторый постоянный угол β (тогда и орт $\overline{\mathbf{e}}_{1,0}$ совместится с ортом $\overline{\mathbf{o}} \equiv \overline{\mathbf{e}}_{1,7}$).

Представим матрицу Γ_7 оператора ориентации схвата в виде произведения $\Gamma_7 = \Gamma_{\pi-\delta}\Gamma_\beta$ матриц поворота на углы $\pi-\delta$ и β вокруг векторов $\overline{\mathbf{e}}$ и $\overline{\mathbf{e}}_{2,0}$. Поскольку при $t=0$ матрица Γ_7 и угол δ известны, причём

$$\Gamma_7 = \begin{pmatrix} n_x & o_x & a_x \\ n_y & o_y & a_y \\ n_z & o_z & a_z \end{pmatrix}, \qquad \Gamma_\beta = \begin{pmatrix} \cos\beta & -\sin\beta & 0 \\ \sin\beta & \cos\beta & 0 \\ 0 & 0 & 1 \end{pmatrix},$$

$$\Gamma_{\pi-\delta} = \cos(\pi-\delta)\,I + (1-\cos(\pi-\delta))\,e\otimes e + \sin\delta\,\breve{e}$$

(знак $\otimes$ обозначает операцию тензорного умножения), то, обозначая через c_{ij} компоненты матрицы $\Gamma_{\pi-\delta}$ и вычисляя два элемента левого столбца матрицы $\Gamma_\beta \equiv \Gamma_{\pi-\delta}^{\mathrm{T}}\Gamma_7$, имеем:

$$\cos\beta = c_{00}n_x + c_{10}n_y + c_{20}n_z, \qquad \sin\beta = c_{01}n_x + c_{11}n_y + c_{21}n_z.$$

Получив по этим формулам $\cos\beta$ и $\sin\beta$, можно найти сам угол β как $\beta = \mathrm{atan2}(\sin\beta, \cos\beta)$.

Численное решение задачи Коши в подпрограмме **Integrate** осуществляется обращением к модулю численного интегрирования дифференциальных уравнений **DPI** из библиотеки **tmx.lib**. В этом модуле реализован *метод Дормана – Принса* 5-го порядка точности [**22**], относящийся к семейству методов Рунге – Кутты.

Это – явный 7-стадийный метод, требующий шесть вычислений правых частей системы дифференциальных уравнений на шаге интегрирования и допускающий эффективное управление длиной шага в соответствии с заданным допуском ε_A для локальной погрешности.

С помощью непрерывного расширения [**22**] метода Дормана – Принса в модуле **DPI** реализована возможность вычисления значения решения в точках, не совпадающих с узлами сетки, что позволяет, в частности, обеспечить при необходимости приведение результатов интегрирования к равномерной сетке. Взаимодействие модуля с вызывающей программой организовано по принципу инверсного управления, так что он работает в пошаговом режиме (одно обращение к модулю на одно вычисление правых частей).

Подпрограмма **Integrate** обеспечивает инициализацию модуля **DPI** и его вызов в пошаговом режиме с запоминанием получаемых результатов, приведённых к равномерной сетке, в рабочих массивах. Часть этих результатов (по мере их получения) выводится на экран. По окончании решения задачи Коши подпрограмма **Integrate** осуществляет терминацию интегратора и обращается к подпрограмме **Tables** для вывода на экран остальных результатов интегрирования.

Вычисление правых частей системы (23), вынесенное в подпрограмму **Problem**, организовано следующим образом:

– подпрограмма перевычисляет значения переменных параметров в массиве **DH** в соответствии с текущими значениями координат в сочленениях;

– осуществляется вызов подпрограммы **Configuration** для расчёта текущей конфигурации манипулятора;

– вычисляется матрица A_{K} и выполняется её LU-разложение;

– находятся компоненты блочных векторов $\overline{\mathbf{U}}_C$ и $\overline{\mathbf{U}}_O$;

– наконец, путём решения системы (22) определяется столбец $\dot{q}$.

Подпрограмма **Output**, помимо вывода на экран текущих координат точки C и компонент вектора $\overline{\mathbf{a}}$, выводит также число **ns** шагов интегрирования, выполненных к данному моменту времени t, и количество **nf** сделанных при этом вычислений правых частей системы (23) (эта информация бывает полезна при анализе процесса моделирования).

Заметим теперь, что при программировании движения схвата не учитывалась возможность выхода точки C в её программном движении за пределы рабочего пространства робота. Кроме того, не в любой точке этого рабочего пространства схват может иметь задан-

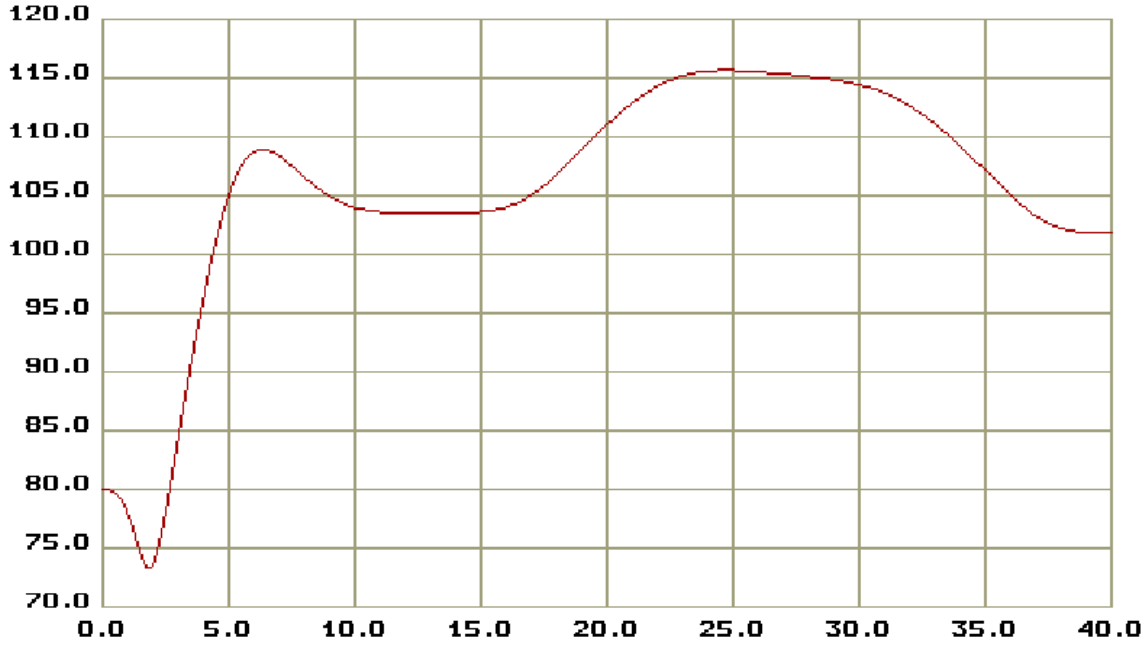
ную ориентацию. Задача анализа рабочего пространства шестизвенного манипулятора весьма сложна [**2**], и её решение при решении задачи о роботе-бармене не предусматривается, но с возможностью того, что описываемая ситуация встретится в процессе моделирования, Вам следует считаться.

Что же при этом происходит? Можно показать, что с приближением к границе рабочего пространства матрица A_K становится плохо обусловленной. На границе же рабочего пространства она оказывается вырожденной, а конфигурация манипулятора в этом случае является особенной.

Как это проявляется в ходе моделирования? На практике программа **kin_31.c** либо выводит сообщение о том, что матрица оказалась вырожденной (этот факт обнаруживается процедурой **LUdek**), либо (что бывает чаще), резко уменьшает, начиная с некоторого момента времени t, шаг интегрирования, практически не продвигаясь вперёд – до тех пор, пока подпрограмма **Integrate** не прервёт моделирование (это происходит в случае, когда значение **nf** превысит заранее заданный предел, равный 20000).

В этом случае следует несколько изменить начальные данные – координаты точек A и B и высоту H. Если же вырождение наступило при $t = 0$, то надо менять уже содержимое массива **DH**, т.е. начальную конфигурацию манипулятора или его конструкцию.

В заключение приведём в качестве примера график функции $q_2(t)$, полученный для 31-го варианта задания.



3.3. Моделирование обратной задачи кинематики в комплексе UM

Решение обратной задачи кинематического анализа манипулятора с помощью комплекса **UM** облегчается тем, что Вы уже решали пря-

мую задачу кинематики для данного манипулятора и, следовательно, уже проделали трудоёмкую работу по вводу геометрических и кинематических характеристик манипулятора и созданию графических объектов. Воспользоваться результатами предыдущей работы можно так.

Прежде всего, нужно войти в среду комплекса **UM** и задать имя новой задачи, а после этого перейти к работе с программой графического ввода.

Войдя в эту программу, увидите в меню **Основная таблица** справа от пункта **Ввод** поле, в котором записано имя Вашей новой задачи. Наберите в этом поле имя старой задачи, выйдите на пункт **Ввод** и нажмите **Enter**. После этого вновь наберите в поле имя новой задачи и обратитесь к пункту **Запись**. Всё! Ваша новая задача теперь представляет собой почти точную копию старой (не будет скопирован файл **functn.pas**, но он Вам и не нужен: этот файл в обратной задаче кинематики будет совсем другим).

Теперь можно в программе графического ввода сделать необходимую корректировку. В частности, надо ввести изменения в описания шарниров.

Прежде всего, координаты в сочленениях более не являются известными функциями времени, а подлежат определению. Поэтому при описании сочленений манипулятора элементарные преобразования типа **tt** или **rt** следует заменить на ЭП типа **tv** или **rv**.

Далее, следует задать закон движения схвата. Как мы видели в предыдущем пункте, для этого достаточно знать неподвижный орт $\bar{\mathbf{e}}$, постоянный угол β и функции времени $\delta(t)$, $r_{Cx}(t)$, $r_{Cy}(t)$, $r_{Cz}(t)$.

Чтобы сделать всю эту информацию доступной для комплекса **UM**, введём 7-й (воображаемый) шарнир, соединяющий основание со схватом и определяемый цепочкой из пяти элементарных преобразований. Первые три имеют тип **tt** и определяют сдвиги вдоль координатных осей на r_{Cx}, r_{Cy} и $r_{Cz}(t)$, переводя точку O в точку C.

Четвёртое (типа **rt**) определяет поворот вокруг орта $\bar{\mathbf{e}}$ на угол $\pi - \delta$, а пятое (типа **rc**) – поворот вокруг оси z на угол β.

Таким образом, описание 7-го шарнира должно иметь следующий вид (таким его можно увидеть в файле **input.dat**):

```
with joint7; type=comm; bd1=0; bd2=6;
  with et; type=tt; ex=1;
           s(t)=rCx(t);
  with et; type=tt; ey=1;
           s(t)=rCy(t);
  with et; type=tt; ez=1;
           s(t)=rCz(t);
  with et; type=rt; ex=<e_x>; ey=<e_y>; ez=<e_z>;
           s(t)=3.1415926-Del(t);
  with et; type=rc ; ez=1.; s0=<β>;
```

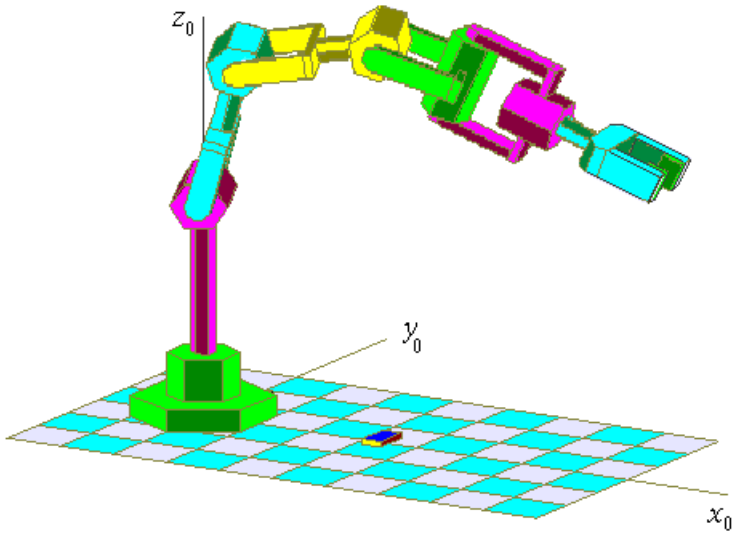
Вместо взятых в угловые скобки значений параметров следует подставить полученные программой **kin_31.c** числовые значения.

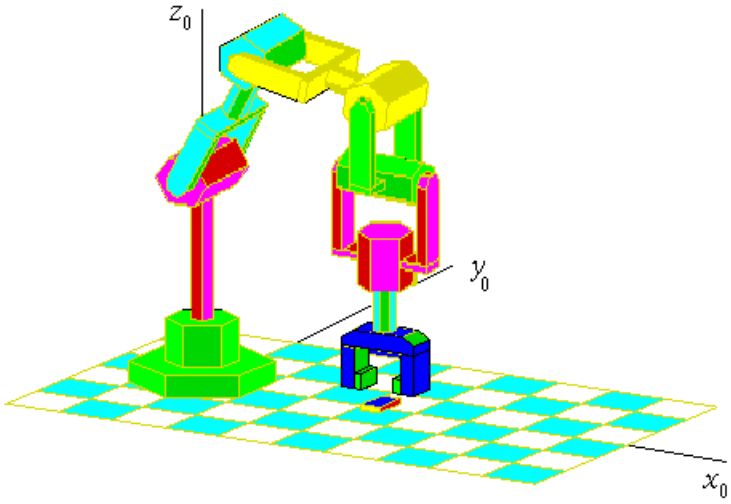
Разумеется, подпрограммы с именами **rCx**, **rCy**, **rCz**, **Del** должны существовать. В данной задаче можете воспользоваться стандартным файлом **functn.pas**, содержащим их тексты. Предусматривается, что с помощью программы графического ввода (пункт **Символ** основного меню) или сразу в файле **constfil.pas** Вы определяете следующие символические параметры: **gCx**, **gCy**, **gCz** – начальные координаты точки C; **gAx**, **gAy**, **gAz**, **gBx**, **gBy**, **gBz** – координаты точек A и B; **H** и **delta** – значения H и δ°.

Не забудьте, что при первом обращении к модулю интегрирования Вам нужно в поле **Время моделирования** задать значение t_{fin}, равное 40 с, и сохранить конфигурацию.

Приведём пару изображений, полученных в анимационном окне модуля интегрирования при решении задачи о роботе-бармене.

На первом из них манипулятор показан в начальной конфигурации, на втором – непосредственно перед захватом объекта (спичечного коробка).





ЛИТЕРАТУРА

Рекомендуемая

1. **Воробьёв Е.И., Попов С.А., Шевелёва Г.И.** Механика промышленных роботов. Кн. 1: Кинематика и динамика. М.: Высш. шк., 1988. 304 с.
2. **Шахинпур М.** Курс робототехники. М.: Мир, 1990. 527 с.
3. **Кострикин А.И., Манин Ю.И.** Линейная алгебра и геометрия. М.: Наука, 1986. 304 с.

Использованная

4. **Яглом И.М.** Математические структуры и математическое моделирование. М.: Сов. Радио, 1980. 144 с.
5. **Осадченко Н.В.** Метод винтов в вычислительной механике // Проблемы механики управляемых систем, машин и механизмов: Межвузовск. тематич. сб. № 77. М.: Моск. энерг. ин-т, 1985. С.61–68.
6. **Диментберг Ф.М.** Теория винтов и её приложения. М.: Наука, 1978. 328 с.
7. **Аксельрод Б.В.** Описание динамики манипуляторов с применением теории винтов // Изв. АН СССР. Мех. твёрд. тела. 1985. № 2. С.79–84.
8. **Liu Yanzhu.** Screw-Matrix Method in Dynamics of Multibody Systems // Acta Mechanica Sinica. 1988. Vol.4. № 2. P.165–174.
9. **Селиг Дж.М.** Применение теории винтов в динамике роботов // Прикл. математика и механика. 1991. Т.55. Вып.2. С.201–211.
10. **Айзерман М.А.** Классическая механика. М.: Наука, 1974. 368 с.
11. **Арнольд В.И.** Математические методы классической механики. М.: Наука, 1989. 472 с.
12. **Трусделл К.** Первоначальный курс рациональной механики сплошных сред. М.: Мир, 1975. 592 с.
13. **Болтянский В.Г.** Оптимальное управление дискретными системами. М.: Наука, 1973. 448 с.
14. **Бурбаки Н.** Алгебра. Алгебраические структуры. Линейная и полилинейная алгебра. М.: Физматгиз, 1962. 516 с.

15. **Гантмахер Ф.Р.** Теория матриц. М.: Наука, 1988. 552 с.

16. **Корецкий А.В., Осадченко Н.В.** Метод винтов и решение на ЭВМ задач кинематического анализа манипуляционных роботов // Тезисы докладов международной конференции “Информационные средства и технологии”. Т. 2. М.: Изд-во “Станкин”, 1996. С.48–53.

17. **Булгаков Б.В.** Колебания. М.: Гостехиздат, 1954. 892 с.

18. **Корецкий А.В., Осадченко Н.В., Погорелов Д.Ю.** Практика моделирования робототехнических систем средствами программного комплекса “Универсальный механизм” // Тезисы докладов международной конференции “Информационные средства и технологии”. Т. 2. М.: Изд-во “Станкин”, 1996. С.54–59.

19. **Погорелов Д.Ю.** Введение в моделирование динамики систем тел. Брянск: Брянский гос. техн. ун-т, 1997. 156 с.

20. **Новожилов И.В., Зацепин М.Ф.** Типовые расчёты по теоретической механике на базе ЭВМ. М.: Высш. шк., 1986. 136 с.

21. **Воеводин В.В., Кузнецов Ю.А.** Матрицы и вычисления. М.: Наука, 1984. 320 с.

22. **Хайрер Э., Нерсётт С., Ваннер Г.** Решение обыкновенных дифференциальных уравнений. Нежёсткие задачи. М.: Мир, 1990. 512 с.

Оглавление

Учебное издание

Корецкий Александр Владимирович
Осадченко Николай Владимирович

КОМПЬЮТЕРНОЕ МОДЕЛИРОВАНИЕ КИНЕМАТИКИ МАНИПУЛЯЦИОННЫХ РОБОТОВ

Методическое пособие
по курсу
“Вычислительная механика”
для студентов, обучающихся по специальности
“Роботы и робототехнические системы”

Редактор издательства О.М. Горина
ЛР № 020528 от 05.06.97

Темплан издания МЭИ 2000 (I), метод.
Подписано к печати 30.01.2000 г.
Формат 60 × 84 ⁄ 16
Физ. печ. л. 3,0
Тираж 300 Изд. № 13 Заказ 277

Издательство МЭИ, 111250, Москва, Красноказарменная, д. 14
Типография ЦНИИ “Электроника”, 117415, Москва,
просп. Вернадского, д. 39