

Лабораторная работа №1

Построение математической модели двигателя NXT

1 Методические рекомендации

До начала работы студент должен установить на компьютере необходимый для работы с набором LEGO Mindstorms NXT драйвер и настроить подключение к интернету. Подробности следует узнать у преподавателя.

2 Теоретические сведения

В данном пособии изучаются основные принципы функционирования неотъемлемой части многих робототехнических устройств — электродвигателя постоянного тока. Преимущественно рассматривается лишь механическая сторона его работы, протекающим в нём электродинамическим процессам будет посвящено следующее занятие.

Математическая модель электродвигателя

Основными частями любого двигателя постоянного тока (рис. 1) являются *статор* (или *индуктор*) и *ротор* (или *якорь*). Статор представляет из себя неподвижную часть двигателя, грубо говоря, его корпус, ротор же — ту составляющую, которая приводится во вращение.



Рис. 1. Пример двигателя постоянного тока.

Чтобы в дальнейшем было возможно теоретическое рассмотрение исследуемого объекта (двигателя), в первую очередь надо составить *математическую модель* его работы. Под ней понимается совокупность уравнений, описывающих изменения характеризующих объект величин и их взаимосвязь друг с другом.

С указанной целью запишем для ротора двигателя второй закон Ньютона в форме, используемой при рассмотрении вращательного движения:

$$M_{\Sigma} = J\dot{\omega}, \quad (1)$$

где $M_\Sigma(t)$ — сумма всех моментов сил, действующих на ротор; J — момент инерции ротора¹; $\omega(t)$ — его угловая скорость вращения. Точкой над буквенным обозначением какой-либо величины здесь и далее будем показывать производную по времени (следовательно $\dot{\omega}$ есть угловое ускорение ротора), а конструкцией, например, $M_\Sigma(t)$ — тот факт, что M_Σ является функцией от времени (для удобства записи формул она может опускаться).

Надо сказать, что составив для упомянутых характеристик двигателя это уравнение, мы уже определили его математическую модель. Действительно, оно устанавливает связь между всеми величинами, описывающими движение ротора, которые потребуются нам в дальнейшем и тем самым, например, позволяет по известной одной из функций $\dot{\omega}(t)$ или $M_\Sigma(t)$ найти другую. Иными словами, благодаря ему появляется возможность прогнозировать поведение ротора двигателя в той или иной ситуации. Несмотря на это, прежде, чем пойти дальше, скажем еще об одном.

В данном случае для определения основного уравнения движения ротора двигателя оказалось удобным использование второго закона Ньютона. Однако существуют и другие способы нахождения уравнений, описывающих движение механических систем. Рассмотрим тот из них, который подразумевает использование так называемого *уравнения Лагранжа второго рода*, записанного с использованием *кинетического потенциала*, и заключается в следующем.

Во-первых, для исследуемой системы записывается *функция Лагранжа*, или *кинетический потенциал*, которая(-ый) представляет из себя разность её кинетической $T(t)$ и потенциальной $U(t)$ энергий:

$$L(t) = T - U. \quad (2)$$

Полагая последнюю, которая в рассматриваемой задаче имеет постоянное значение, равной нулю и раскрывая кинетическую энергию вращения ротора по соответствующей формуле, получаем для функции Лагранжа электродвигателя следующее выражение:

$$L(t) = T - 0 = \frac{J\dot{\theta}^2}{2}, \quad (3)$$

где $\theta(t)$ — угол, на который повернулся ротор двигателя из начального положения.²

Необходимо отметить, что в функцию Лагранжа обязательно будут входить некоторые другие функции от времени, характеризующие состояние системы в каждый момент времени — так называемые *обобщённые координаты*, и производные от них — *обобщённые скорости*. В данном случае в роли обобщённой координаты выступает $\theta(t)$.

Во-вторых, для каждой обобщённой координаты и соответствующей ей обобщенной скорости записывается уже упоминаемое уравнение Лагранжа. Для нашей $\theta(t)$ оно принимает вид

$$\frac{d}{dt} \frac{\partial L}{\partial \dot{\theta}} - \frac{\partial L}{\partial \theta} = M_\Sigma(t), \quad (4)$$

¹При рассмотрении вращательного движения тела этот параметр играет ту же роль, что и масса тела при рассмотрении его поступательного движения.

²Из определения $\theta(t)$ следует, что $\theta(0) = 0$.

где, например, конструкцией $\partial L / \partial \theta$ обозначена частная производная от L по θ ³, а конструкцией d/dt — производная по времени (в данном случае она берется от функции — результата операции $\partial L / \partial \dot{\theta}$). При этом в правой части уравнения Лагранжа будет находиться некоторая функция, имеющая размерность силы, если обобщённая координата суть линейный размер (например обычная декартова координата), и момента силы, если обобщённая координата — некоторый угол. Эта функция называется *обобщенной силой*. На данный момент под ней следует понимать просто сумму всех действующих в системе сил или моментов сил, которые вызывают изменение рассматриваемой обобщенной координаты. В действительности сказанное неверно⁴, однако описание точных правил определения обобщенных сил мы оставим для следующих лабораторных.

После написания уравнения остается только взять необходимые производные, что мы и сделаем.

Так как в функции Лагранжа (3) нет членов включающих θ , второе слагаемое можно отбросить⁵. Тогда уравнение (4) с учётом (3) запишется в виде

$$M_{\Sigma} = \frac{d}{dt} \left(\frac{\partial}{\partial \dot{\theta}} \left(\frac{J\dot{\theta}^2}{2} \right) \right). \quad (5)$$

Выполнив дифференцирование по $\dot{\theta}$, будем иметь

$$M_{\Sigma} = \frac{d}{dt} (J\dot{\theta}), \quad (6)$$

и взяв производную по времени, получим окончательный результат:

$$M_{\Sigma} = J\ddot{\theta}. \quad (7)$$

Если теперь ввести обозначение $\omega = \dot{\theta}$, станет очевидно, что найденное данным методом уравнение совпадает с тем, которое было получено с помощью второго закона Ньютона (1). Таким образом, мы показали, что оба метода эквивалентны. Использованию уравнения Лагранжа ещё будут посвящены следующие занятия, а теперь продолжим дальнейшее изучение работы двигателя.

Пример применения математической модели

С помощью найденной математической модели установим зависимость $\omega(t)$, которая

³В данном случае при дифференцировании L по θ , надо принять все переменные кроме θ постоянными величинами, а θ той переменной, по которой берется производная. Нижеприведенные примеры должны пояснить сказанное:

$$\frac{\partial (3x + 4y)}{\partial x} = 3, \quad \frac{\partial \left(\frac{7y^2}{x} + 2y \right)}{\partial y} = \frac{14y}{x} + 2.$$

⁴Все моменты сил, действующие на якорь, которые мы будем рассматривать в дальнейшем, допускают указанное упрощение, поэтому в правой части уравнения (4) и стоит M_{Σ} без каких-либо изменений.

⁵Производная от постоянной равна нулю.

выполняется при разгоне из состояния покоя ненагруженного ротора двигателя при подаче на него постоянного напряжения.

Как уже было сказано, для того чтобы это действие было возможным, необходимо знать зависимость $M_{\Sigma}(t)$, поэтому сначала определим, как в данном процессе выглядит она.

Величину M_{Σ} можно представить суммой двух слагаемых:

$$M_{\Sigma} = M_{el} + M_{oth}, \quad (8)$$

где M_{el} — момент силы, возникающий в двигателе из-за протекающих в нем электродинамических процессов и раскручивающий его якорь; M_{oth} — сумма всех остальных моментов сил,⁶ действующих на ротор. Поскольку мы рассматриваем ситуацию, когда якорь двигателя ничем не нагружен и полагаем силы трения отсутствующими, то

$$M_{oth} = 0, \quad (9)$$

а значит

$$M_{\Sigma} = M_{el}, \quad (10)$$

то есть ранее поставленная задача сводится к нахождению вида функции $M_{el}(t)$.

Вращающий момент M_{el} несложно определить, если предположить его пропорциональным действующей в цепи двигателя ЭДС (U_{cv})⁷:

$$M_{el} = \alpha_1 U_{cv}, \quad (11)$$

где α_1 — некоторая постоянная. В этом случае можно сказать следующее.

В начальный момент времени, когда мы только подали напряжение на двигатель, а ротор ещё не пришел в движение, момент силы M принимает некоторое значение M_{st} , которое в дальнейшем мы будем называть *пусковым моментом*, равное

$$M_{st} = \alpha_1 U_{ctrl}, \quad (12)$$

где $U_{ctrl} = const$ — ЭДС источника электрической энергии, например аккумулятора, (в самом начале создавать U_{cv} будет только она). В последующие моменты времени из-за вращения ротора в его катушках возникнет ЭДС индукции, направленная против U_{ctrl} и прямо пропорциональная угловой скорости вращения ротора⁸:

$$\mathcal{E}_i(\omega) = \alpha_2 \omega, \quad (13)$$

где α_2 — некоторая постоянная. При этом суммарная ЭДС в цепи будет равняться

$$U_{cv} = U_{ctrl} - \mathcal{E}_i \quad (14)$$

⁶В качестве примера таких можно привести моменты сил трения и моменты, создаваемые весами каких-либо конструктивных элементов, прикрепленных к валу ротора.

⁷На следующем занятии будет показано, что при определенных условиях это предположение полностью оправдывается.

⁸Это равенство также будет доказано на следующем занятии.

и, следовательно, будет уменьшаться. Согласно выражению (11), уменьшаться будет и момент силы, причем, подставив значение для U_{cv} из (11), выражение для \mathcal{E}_i из (13) и значение для U_{ctrl} из (12) в уравнение (14), мы найдем зависимость $M_{el}(\omega)$, которая при этом выполняется:

$$\frac{M_{el}}{\alpha_1} = \frac{M_{st}}{\alpha_1} - \alpha_2\omega.$$

Отсюда окончательно имеем, что

$$M_{el}(\omega) = M_{st} - \alpha_1\alpha_2\omega. \quad (15)$$

Значение неизвестной постоянной (произведения $\alpha_1\alpha_2$), входящей в полученное уравнение, можно определить из условия о том, что при достижении ротором двигателя максимальной скорости вращения ω_{nls} (согласно опытным данным, двигатель не разгоняется бесконечно долго) действующий на него момент силы M_{el} обращается в нуль. Если предположить, что это не так (момент будет отличен от нуля), получается, что на ротор будут продолжать действовать разгоняющие его силы, а значит, он будет ускоряться, что, в свою очередь, неверно. Одним словом, при достижении ротором скорости вращения ω_{nls} выражение (15) принимает вид

$$0 = M_{st} - \alpha_1\alpha_2\omega_{nls},$$

откуда имеем, что

$$\alpha_1\alpha_2 = \frac{M_{st}}{\omega_{nls}}. \quad (16)$$

С учетом этого выражения зависимость (15) примет более удобную для использования форму:

$$M_{el}(\omega) = M_{st} - \frac{M_{st}}{\omega_{nls}}\omega, \quad (17)$$

так как все входящие в выражение (17) величины имеют ясный физический смысл.

Получившееся уравнение и есть искомая зависимость $M_\Sigma(t)$ ⁹. Теперь, для того чтобы окончательно решить поставленную задачу о нахождении зависимости скорости вращения ротора от времени ($\omega(t)$), остается лишь заменить M_Σ в выражении (1) соотношением (17) и решить получившееся дифференциальное уравнение¹⁰:

$$M_{st} - \omega \frac{M_{st}}{\omega_{nls}} = J\dot{\omega} \quad \Rightarrow \quad \frac{M_{st}}{\omega_{nls}}(\omega_{nls} - \omega) = J \frac{d\omega}{dt}. \quad (18)$$

Приведем его решение.

Сначала умножим обе его части на dt , а потом разделим на выражение в скобках и на J :

$$\frac{M_{st}}{J\omega_{nls}}dt = \frac{d\omega}{\omega_{nls} - \omega}. \quad (19)$$

Проинтегрируем обе части выражения, в результате чего получим

$$\frac{M_{st}}{J\omega_{nls}}t = -\ln|\omega_{nls} - \omega| + C. \quad (20)$$

⁹В данном случае мы имеем дело со сложной функцией от времени $M_\Sigma(\omega(t))$.

¹⁰Дифференциальным называется уравнение, связывающее искомую функцию (в данном случае $\omega(t)$), её производные и аргумент некоторым соотношением.

В нашем случае $\omega_{nls} > \omega$, поэтому знак модуля можно просто опустить. Перепишем неизвестную постоянную C в виде $-\ln C_1$, тогда получим

$$\frac{M_{st}}{J\omega_{nls}} t = -\ln(\omega_{nls} - \omega) - \ln C_1. \quad (21)$$

Заменив сумму логарифмов логарифмом произведения, потенцированием избавимся от натурального логарифма:

$$\exp\left(-\frac{M_{st}}{J\omega_{nls}} t\right) = C_1(\omega_{nls} - \omega). \quad (22)$$

Из начальных условий — в момент времени $t = 0$ скорость равна нулю $\omega = 0$ — определим C_1 :

$$\exp(0) = C_1(\omega_{nls} - 0) \Rightarrow C_1 = \frac{1}{\omega_{nls}}. \quad (23)$$

С учетом этого значения получим из (22) окончательное решение:

$$\omega(t) = \omega_{nls} \left(1 - \exp\left(-\frac{M_{st}}{J\omega_{nls}} t\right)\right). \quad (24)$$

Проанализировав это выражение, можно сказать, что характер зависимости $\omega(t)$ определяется значениями таких величин, как максимальная скорость вращения ротора, пусковой момент и момент инерции ротора, а её график имеет вид, представленный на рис. 2.

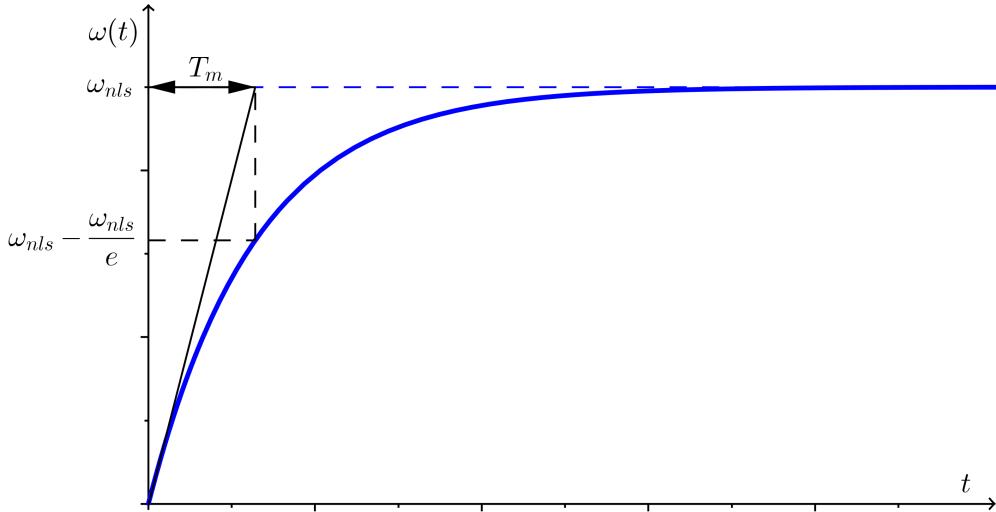


Рис. 2. График зависимости угловой скорости вращения ротора от времени.

Уравнение (24) можно записать, используя еще одну величину, характеризующую двигатель постоянного тока — электромеханическую постоянную времени T_m , равную

$$T_m = \frac{J\omega_{nls}}{M_{st}}. \quad (25)$$

Тогда уравнение (24) примет вид:

$$\omega(t) = \omega_{nls} \left(1 - \exp\left(-\frac{t}{T_m}\right)\right), \quad (26)$$

откуда нетрудно определить физический смысл величины T_m : она есть время, за которое скорость вращения первоначально покоящегося ротора возрастает до

$$\omega(T_m) = \omega_{nls} - \frac{\omega_{nls}}{e}. \quad (27)$$

Таким образом, мы решили ранее поставленную задачу, определив с помощью математической модели закон изменения скорости вращения ротора двигателя от времени, однако прежде, чем закончить данный раздел, получим из найденной зависимости выражения для углового ускорения якоря двигателя (ε) и для его угловой координаты (θ):

$$\varepsilon(t) = \dot{\omega} = \frac{\omega_{nls}}{T_m} \exp\left(-\frac{t}{T_m}\right), \quad (28)$$

$$\theta(t) = \int \omega dt = \omega_{nls} t + \omega_{nls} T_m \exp\left(-\frac{t}{T_m}\right) + C_2, \quad (29)$$

где C_2 — некоторая постоянная, чье значение можно определить из начальных условий ($\theta(0) = 0$):

$$\theta(0) = \omega_{nls} \cdot 0 + \omega_{nls} T_m \exp\left(-\frac{0}{T_m}\right) + C_2 \quad \Rightarrow \quad C_2 = -\omega_{nls} T_m \quad (30)$$

С учетом найденного значения постоянной C_2 окончательно имеем:

$$\varepsilon(t) = \frac{\omega_{nls}}{T_m} \exp\left(-\frac{t}{T_m}\right), \quad (31)$$

$$\theta(t) = \omega_{nls} \left(t - T_m \left(1 - \exp\left(-\frac{t}{T_m}\right) \right) \right), \quad (32)$$

и графики, показанные на рис. 3.

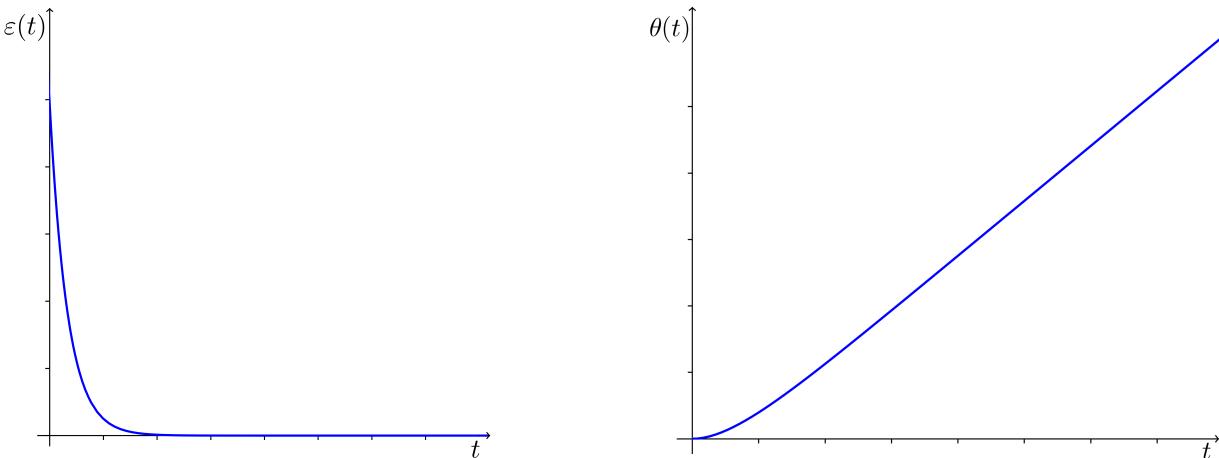


Рис. 3. Графики зависимостей $\varepsilon(t)$ и $\theta(t)$.

Целью практической части данной работы и будет опытная проверка результатов, полученных в этом разделе.

Моделирование работы двигателя

Для того чтобы получать из математической модели количественную информацию о рассматриваемом процессе, необязательно прибегать к решению дифференциальных уравнений. Например, с указанной задачей успешноправляются ряд вычислительных программ, подобных Scilab Xcos, которую мы будем использовать в данном курсе. При этом суть требуемой работы в таком случае оказывается следующей.

Во-первых, на основании составленной математической модели исследуемого процесса в упомянутых программах строится блок-схема, описывающая единичный такт циклически

повторяющихся математических операций, соответствующих изменению характеризующих процесс величин. К слову сказать, созданная в Xcos на основании уравнения (18) для ранее рассмотренного нами разгона ненагруженного двигателя эта схема будет выглядеть так, как показано на рис. 4¹¹. Во-вторых, по полученной схеме производится моделирование процесса и непосредственно сбор всех интересующих исследователя данных.

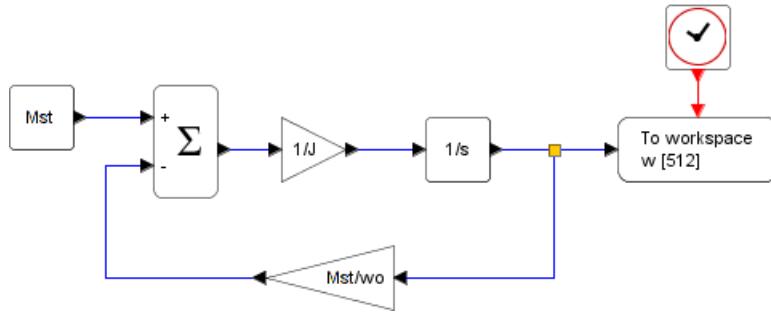


Рис. 4. Схема моделирования исследуемого процесса.

Для раскрытия принципов работы данного подхода скажем следующее.

Каждый блок, составляющий схему имеет в своем составе *входы* и *выходы*, обозначенные треугольничками, направленными соответственно в тело блока и от него. Общение блоков осуществляется с помощью сигналов — определенных числовых значений. Последние, перемещаясь по каналам связи (обозначены синими и красными линиями), приходят на вход блока, в нем изменяются и уже в измененной форме поступают на его выход для дальнейшего путешествия по каналам связи.

Начинающий эту схему блок с буквенным обозначением «Mst» и с одним выходом означает, что на последний подается значение, хранящееся в переменной Mst (очевидно, что ей надо присвоить значение, равное M_{st}). Блоки в форме треугольника умножают входящий сигнал на значение написанного на себе выражения и подают полученный результат на выход. Блок со значком суммы (Σ) вычитает из своего сигнала, идущего на вход, помеченный знаком «+», другой сигнал (идущий на вход, помеченный знаком «-») и подает на свой выход значение этой разности. Блок с надписью «1/s» интегрирует входной сигнал и прибавляет к числовому

¹¹Некоторые несущественные детали схемы такие, как, например, форма некоторых блоков, в зависимости от используемого программного обеспечения и его версий могут различаться.

значению, хранящемуся в его памяти.¹² Эта сумма потом поступает на его выход. Остальные два блока — с изображением часов и с фразой «To workspace» — нужны лишь для того, чтобы сохранить результаты моделирования, отнесенные к заданным промежуткам времени, в используемой среде разработки.

Теперь же покажем, как данная схема расшифровывается (читается). На блок-сумматор (тот, который помечен значком \sum), приходят два сигнала: один поступает из блока с надписью «Mst» и несет значение, равное пусковому моменту, второй — с конца схемы (выхода интегратора) и заключает в себе хранящееся в его памяти число;¹³ умноженное нижним блоком-треугольником на M_{st}/ω_{nls} . Эти сигналы вычтываются друг из друга, и полученная разность приходит на следующий блок, где умножается на значение $1/J$. В таком виде она идет дальше. Таким образом, на блок с надписью «1/s» поступает значение, равное M_{st}/J , которое после интегрирования, согласно (18), даст не что иное, как некоторое значение угловой скорости вращения ротора ω_1 в «первый» момент времени.¹⁴ Этот сигнал, в свою очередь, пойдет в память среды Xcos и, будучи умноженным на M_{st}/ω_{nls} , на один из входов сумматора. С учетом этого имеем, что при втором и последующих обходах сигналами схемы на выходе последнего будет формироваться значение $M_{st} - \omega_i M_{st}/\omega_{nls}$, где ω_i — скорость, вычисленная в предыдущем цикле. Это значение, поделенное на J и после этого проинтегрированное, согласно (18), опять даст значение скорости ω_{i+1} , но уже в другой момент времени — следующий за тем, в течение которого скорость была равна ω_i .

Так повторяясь много раз, данный процесс сформирует два массива данных, в первом из которых будут находиться значения скорости — значения сигналов, пришедших на вход блока с надписью «To workspace» со стороны схемы, а во втором — соответствующие моменты времени, которые представляют из себя значения сигналов, поступивших в этот блок со стороны «часов».

¹²Имеется в виду следующее. Как известно, определенный интеграл дает лишь приращение первообразной $F(x)$ своей функции-«аргумента» $f(x)$, то есть:

$$\int_a^b f(x) dx = F(b) - F(a). \quad (33)$$

При таком условии, для того чтобы иметь возможность вычислять именно значение первообразной в интересующей точке $x = b$, надо знать ее значение в начальной точке $x = a$:

$$F(b) = \int_a^b f(x) dx + F(a). \quad (34)$$

Возможность помещать в память интегратора постоянное числовое значение как раз и рассчитана на то, что в качестве такого будет взято $F(a)$, ведь в этом случае, согласно сказанному, данный блок будет выдавать значение $F(b)$.

¹³Забегая вперед, можно сказать, что в результате моделирования нашей схемы мы хотим получить числовые данные для зависимости $\omega(t)$. Поскольку в начале рассматриваемого процесса скорость ротора равна нулю, то и в память данного блока следует поместить 0.

¹⁴Каждый обход схемы сигналами осуществляется через равные промежутки времени, длительность которых можно задавать вручную. По указанной причине первое прохождение схемы и полученные в результате его числовые результаты будут отнесены к моменту времени, равному $t = \tau$, где τ — упомянутая временная задержка. Следующий обход — к $t = 2\tau$ и т.д. Моменту времени $t = 0$ будет соответствовать сигнал находящийся в конце схемы при ее запуске.

Построенный по их данным график в случае, если схема была построена правильно, совпадет с графиком функции (24) — см. рис. 2.

В заключение сделаем два замечания. Во-первых, те численные значения, с которыми блоками приходится иметь дело, в используемом в рамках данного курса ПО можно задавать и в явном виде, а не через переменные. С учетом этого схема моделирования двигателя может выглядеть и как на рис. 5. Во-вторых, если добавить в конце схемы процесса еще

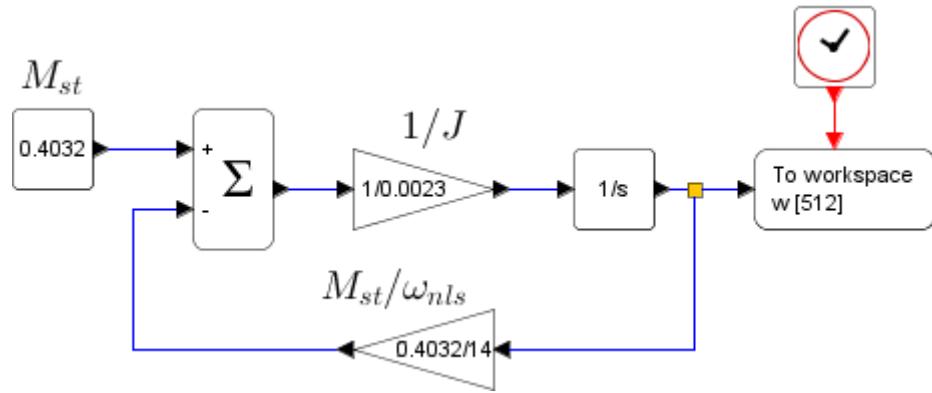


Рис. 5. Схема процесса, возвращающая значения зависимости $\omega(t)$.

один интегратор (рис. 6), то на его выходе будут формироваться значения угловой координаты поворота якоря (θ).

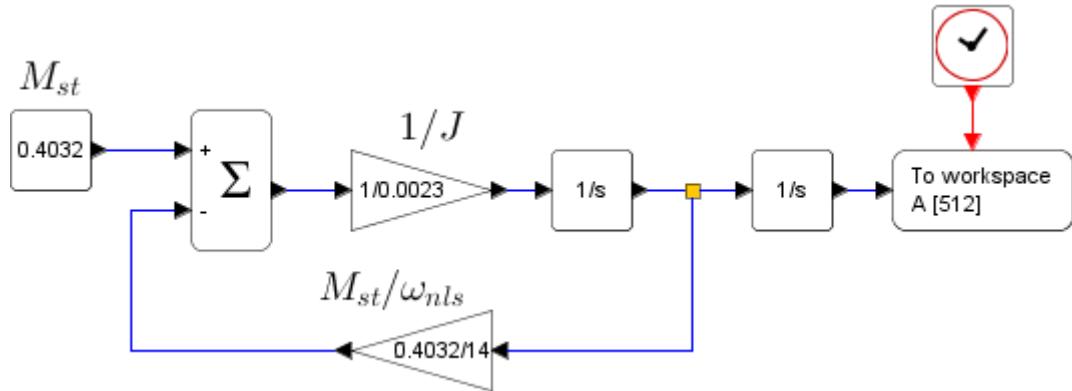


Рис. 6. Схема процесса, возвращающая значения зависимости $\theta(t)$.

3 Цель работы

Познакомиться с оборудованием и программным обеспечением, которые понадобятся при изучении материала данного курса. Экспериментально проверить истинность найденных функций, описывающих работу ненагруженного двигателя постоянного тока, и определить значения входящих в них параметров ω_{nls} и T_m .

4 Порядок выполнения работы

1 Снятие показаний с двигателя NXТ

(при возникновении вопросов попробуйте обратиться к Приложению А).

1.1 Соберите конструкцию, показанную на рис. 7.



Рис. 7. Общий вид экспериментальной конструкции.

1.2 Напишите программу в среде разработки Bricx Command Center, которая подает на двигатель NXТ максимально возможное постоянное напряжение (данному требованию удовлетворяет значение аргумента voltage функции OnFwd(), равное 100). При этом она также должна через одинаковые промежутки времени снимать показания текущего угла поворота ротора и записывать их вместе с соответствующими значениями времени, прошедшего с начала работы программы, в два столбца в текстовый (.txt) файл на NXТ.

Пример содержания файла (Первый столбец — значения угла поворота в градусах, второй столбец — значения времени в миллисекундах):

0	0
0	5
0	10
1	15
2	20
3	25
4	30
6	35
...	...
391	491
395	496
400	501
...	...

1.3 Скомпилируйте и загрузите программу в блок NXT.

1.4 Запустите программу на выполнение.

1.5 Переместите созданный текстовый файл, содержащий экспериментальные данные, из памяти NXT на компьютер.

1.6 Повторите действия, описанные в трех прошлых пунктах, при 9 других значениях аргумента voltage функции OnFwd(), равных элементам следующего множества $\{80, 60, 40, 20, -20, -40, -60, -80, -100\}$.

2 Обработка экспериментальных данных.

Обработку снятых с двигателя показаний следует провести в среде Scilab. Скачать ее последнюю версию можно с сайта www.scilab.org.

2.1 Запустите Scilab.

2.2 В открывшемся командном окне (рис. 8) введите команду `scinotes` — запустится редактор программного кода (рис. 9). Все дальнейшие действия выполняйте в данном редакторе.

2.3 Ведите команду `results=read("file_name",-1,2)`, где `file_name` — полное имя файла, содержащего показания, снятые с двигателя во время выполнения пункта 15, например `C:\Documents\exp_data.txt`. Функция `read` считывает заданный файл и на его основании формирует числовую матрицу указанного двумя последующими числами размера (первое число — количество строк, второе — столбцов). Число -1

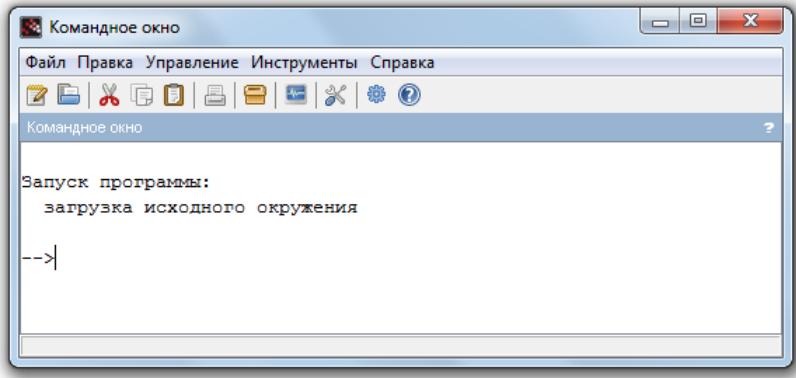


Рис. 8. Командное окно Scilab.

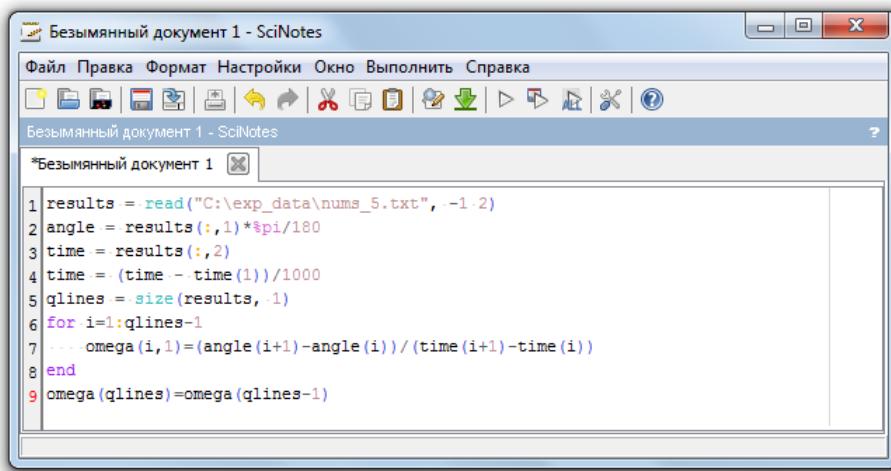


Рис. 9. Текстовый редактор Scinotes.

ставится в том случае, если точное количество структурных единиц (столбцов или строк) неизвестно, однако писать `-1` одновременно вместо двух аргументов нельзя. Результат, возвращаемый этой функцией, присваивается матрице `results` (переменные в Scilab отдельно объявлять не надо).

2.4 Определите количество строк в получившейся матрице командой `qlines=size(results,1);` результат сохранится в переменной `qlines`. К слову сказать, если взамен единицы написать `2`, в переменной `qlines` сохранится количество столбцов.

2.5 Скопируйте первый столбик матрицы `results` в отдельную одностолбцовую матрицу `angle`, используя команду `angle=results(:,1)`.

Обращаясь в Scilab к отдельным элементам матриц, после имени последних в круглых скобках записывают два числа — соответственно номер строки и столбика, на пересечение которых находится интересующий элемент. Данный случай не исключение. Указанное на месте номера строки двоеточие означает, что обращение идет ко всем строкам сразу, поэтому в матрицу `angle` копируется сразу весь первый столбик.

- 2.6 Теперь переведите все значения матрицы `angle` из градусов в радианы: `angle=angle*pi/180`. Несложно догадаться, что конструкцией `%pi` в Scilab обозначается число π .
- 2.7 Второй столбик матрицы `results` также скопируйте в отдельную одностолбцовую матрицу `time` и сразу же переведите из миллисекунд в секунды: `time=results(:,2)/1000`. Одним действием можно было обойтись и в случае матрицы `angle`.
- 2.8 Постройте график зависимости угла поворота ротора от времени командой `plot2d(time, angle, 2)`, последний аргумент которой задает цвет будущего графика.

Если теперь запустить написанный код на выполнение, кликнув, например, на изображение треугольничка на панели инструментов, то в появившемся графическом окне отобразится график, похожий на представленный на рис. 10.

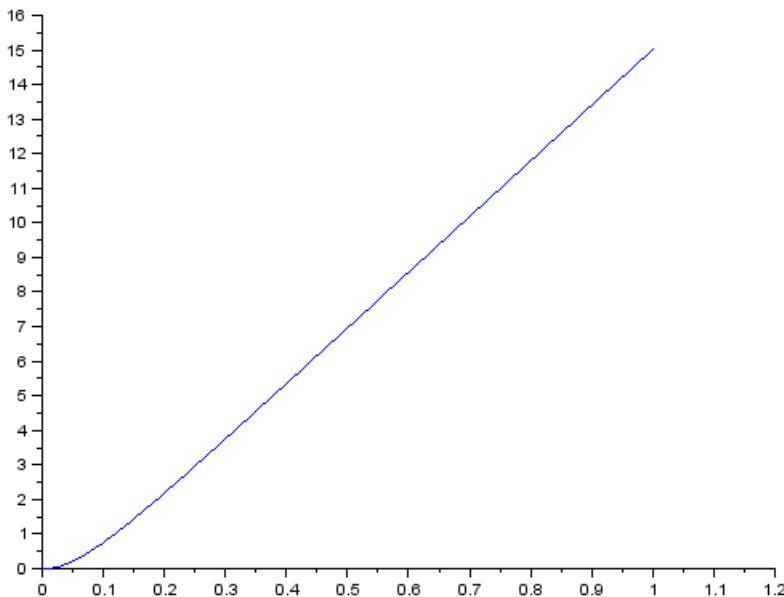


Рис. 10. Первый график.

- 2.9 Теперь надо выполнить *аппроксимацию* полученной зависимости — подобрать некоторую функцию и найти такое ее уравнение, которое наилучшим образом усреднит полученный результат. В качестве последней следует взять непосредственно функцию $\theta(t)$ — выражение (32). Чтобы найти *методом наименьших квадратов* (прочтите о нем в дополнительной литературе) те значения параметров ω_{nls} и T_m , которые обеспечат графику функции (32) наибольшее сходство с графиком, построенным на основании экспериментальных данных, в Scilab надо сделать следующее.

- 2.9.1 Сформируйте матрицу из значений функции (`angle`) и аргумента (`time`), для элементов которой необходимо выполнить аппроксимацию: `aim=[time,angle]`. В получившейся матрице `aim` первый столбец — матрица `time`, второй — матрица `angle`.

При явном определении матриц в Scilab, как в данном примере, составляющие матрицу элементы (числа или другие матрицы) после знака «=» записываются в квадратных [] или фигурных { } скобках. Запятой разделяются элементы одной строки, а точкой с запятой (;) — разные строки.

Так как матрицы time и omega одностолбовые, то и использовался разделитель запятая. Таким образом они составят матрицу размером qlines×2, а не столбик «высотой» 2qlines.

2.9.2 Транспонируйте матрицу aim: `aim=aim'`. Теперь первая ее строка — это матрица time, вторая — матрица angle. Такое представление матрицы aim — обязательное требования для правильной работы функций по аппроксимации.

2.9.3 Задайте вид функции, чьи коэффициенты будут определяться при аппроксимации. Для уравнения зависимости $\theta(t)$ эта операция в Scilab запишется в виде

```
deff('e=func(k,z)', 'e=z(2)-k(1)*(z(1)-k(2)*(1-exp(-z(1)/k(2))))')
```

На самом деле, данным действием задается некоторая функция func, которая вычисляет разность между вторым элементом одностолбцовой (или одностроковой) матрицы z (используются обозначения фигурирующие в определении функции) и значением функции $\theta(t)$. В качестве параметров ω_{nls} и T_m последней берутся значения одностолбцовой (или одностроковой) матрицы k, состоящей из двух элементов, а в качестве значения аргумента (времени) — значение первого элемента матрицы z.

Смысл именно такого определения интересующей функции вытекает исключительно из алгоритмов, которые использует Scilab для выполнения этой операции, но о них в данном пособии говорится не будет.

2.9.4 Задайте одностолбцовую (нельзя одностроковую) матрицу att из двух элементов, где надо разместить те значения параметров ω_{nls} и T_m , которые, как вы думаете, получатся. Это помогает лишь ускорить процесс, поэтому содержание данной матрицы может быть любым, например `att=[15;0.06]`.

2.9.5 Запустите процесс аппроксимации командой `[koeffs,errs] = datafit(func,aim,att)`. В результате ее работы будет создана матрица koeffs, в которую будут сохранены найденные параметры аппроксимирующей функции, и переменная errs, представляющая из себя сумму квадратов разностей экспериментального значения функции (angle) и найденного из уравнения аппроксимирующей функции для каждого из значений матрицы time. Процесс аппроксимации может занять некоторое время.

2.9.6 Введите две новые переменные Wnls и Tm, в которые следует сохранить найденные значения параметров ω_{nls} и T_m аппроксимирующей функции:

```
Wnls = koeffs(1);  
Tm = koeffs(2);
```

- 2.9.7 Для каждого из элементов матрицы time по уравнению аппроксимирующей функции определите соответствующие ее значения и заполните ими новую матрицу model посредством команды `model=Wnls*(time-Tm*(1-exp(-time/Tm)))`.
- 2.10 Постройте полученный аппроксимирующий полином, введя команду `plot2d(time,model,3)`. Теперь весь написанный код следует запустить на выполнение. В результате всех проделанных действий в графическом окне появится еще одна кривая, и общий его вид станет похожим на представленный на рис. 11 пример.¹⁵

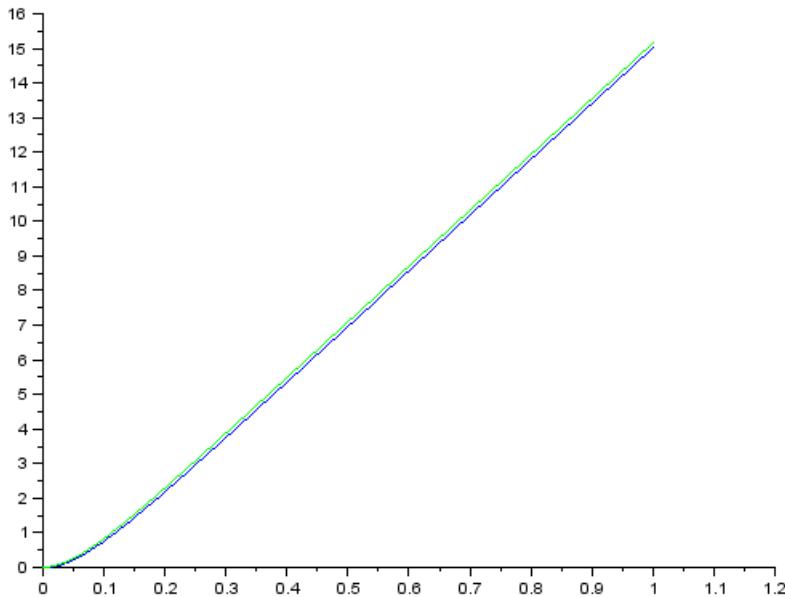


Рис. 11. График с аппроксимирующей кривой.

3 Моделирование схемы в Xcos.

Удостовериться в том, что моделирование представленной на рис. 6 схемы исследуемого процесса дает те же результаты, что и решение дифференциальных уравнений, можно, сравнив график аппроксимирующей функции, найденной в прошлом пункте, с графиком, построенным на основании результатов моделирования схемы. Для создания последней следует использовать пакет прикладных математических программ для моделирования динамических систем Xcos (предустановлен в Scilab). Его внешний вид показан на рис. 12.

3.1 Вызовите рабочие окна Xcos, введя в командное окно Scilab команду `xcos`. В случае, если не появилось окно с палитрами блоков, откройте его, пройдя в главном окне Xcos по вкладке «Вид» и выбрав пункт «Палитры блоков».

3.2 Соберите ту схему нашего двигателя, которая снабжена дополнительным интегратором и, следовательно, выдает значения угла поворота ротора (рис. 6). Значения блоков (надписи на них) выставляются в окне «Ввод значений»,

¹⁵На представленных картинках получающиеся графики для наглядности несколько разнесены относительно друг друга. На самом же деле они будут почти полностью накладываться.

появляющимся при двойном клике на блок или при нажатии сочетания клавиш Ctrl+B с предварительным выделением кликом мыши интересующего блока. Обратите внимание на то, что, если вы хотите использовать в качестве значения блока некоторую переменную, предварительно ее нужно определить, введя в командном окне Scilab конструкцию типа «имя_переменной = ее_значение» (например, $J = 0.0023$).

Значение момента инерции ротора примите равным 0.0023 (оно будет получено на следующем занятии). Пусковой момент следует определить по формуле (25).

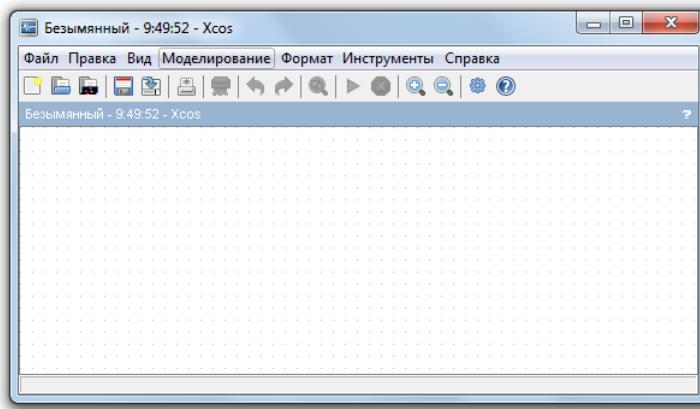


Рис. 12. Главное окно Xcos.

3.3 В окне «Ввод значений» блока с изображением часов полю «Initialization time» присвойте значение, равное нулю (в таком случае учет данных будет производиться с самого начала процесса моделирования). Значение второго поля («Period») есть значение упомянутой в теоретической части пособия величины τ . Несложно догадаться, что оно отразится лишь на плавности будущего графика, поэтому со значением τ можно поэкспериментировать.

3.4 В окне «Ввод значений» блока с надписью «To workplace» есть поле «Size of buffer», устанавливающее размер буфера для записи результатов моделирования. Присвойте ему значение 512. Если в дальнейшем окажется, что этого недостаточно (грубо говоря, будет отсутствовать часть графика), поменяйте установленное значение на большее. Имя переменной, указанное в поле «Scilab variable name», есть общая часть имени массивов результатов моделирования — массива значений скорости (имя.values) и массива соответствующих моментов времени (имя.time). Значение поля «Inherit» оставьте равным нулю.

3.5 Пройдите по вкладке «Моделирование» и выберите пункт «Параметры». В появившемся окне полю «Конечное время интегрирования» присвойте значение, приблизительно равное наибольшему значению времени из графиков, построенных в прошлом пункте («Обработка данных»). Например, на основании графиков, показанных на рис. 10 и 11, это значение можно принять равным 1 с.

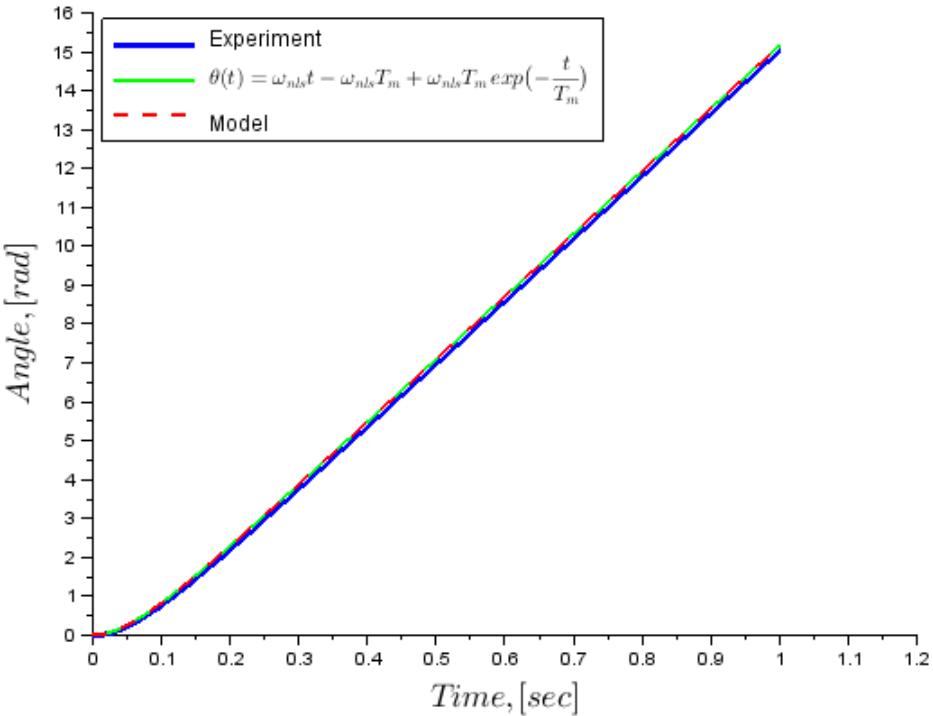


Рис. 13. Итог всех построений.

3.6 Осуществите моделирование системы, нажав на значок с изображением треугольника или выбрав команду «Выполнить» во вкладке «Моделирование».

3.7 Постройте получившийся график, дав в главном окне Scilab команду `plot2d(имя.time, имя.values, 5)` (смысл «имени» см. пункт 34).

Прежде, чем сохранять и экспорттировать график в картинку, пройдите по вкладке «Правка», выберете пункт «Свойства графического окна...» и в появившемся окне отредактируйте свойства графиков, осей и графического окна такие, как цвет, подписи к осям и т.д. Пример конечного результата см. рис. 13. Имеющаяся на нем легенда строится командой `legend()`, в качестве аргументов которой следует указать названия графиков в порядке их построения и «номер» угла, в которой ее следует разместить. Например, показанная на рис. 13 легенда была построена командой `legend('Experiment', '$\theta(t)=\omega_{nls}t-\omega_{nls}T_m+\omega_{nls}T_m\exp(-\frac{t}{T_m})$', 'Model', 2)`.¹⁶

4 Повторите все описанные действия для оставшихся 9 файлов с данными, полученных в п. 16 – получите еще 9 пар значений для величин ω_{lns} и T_m и 9 графиков, подобных показанному на рис. 13.

5 Постройте графики для полученных зависимостей $\omega_{lns}(\text{voltage})$ и $T_m(\text{voltage})$, подобные показанным на рис. 14 и 15.

Любые отступления от приведенного порядка действий в рамках тех же самых программ, дающие необходимый конечный результат, разрешены.

¹⁶Как видно из этой строки, чтобы применять в надписях, встречающихся в среде Scilab, команды языка TeX и макропакета L^AT_EX, их надо ограждать символами \$.

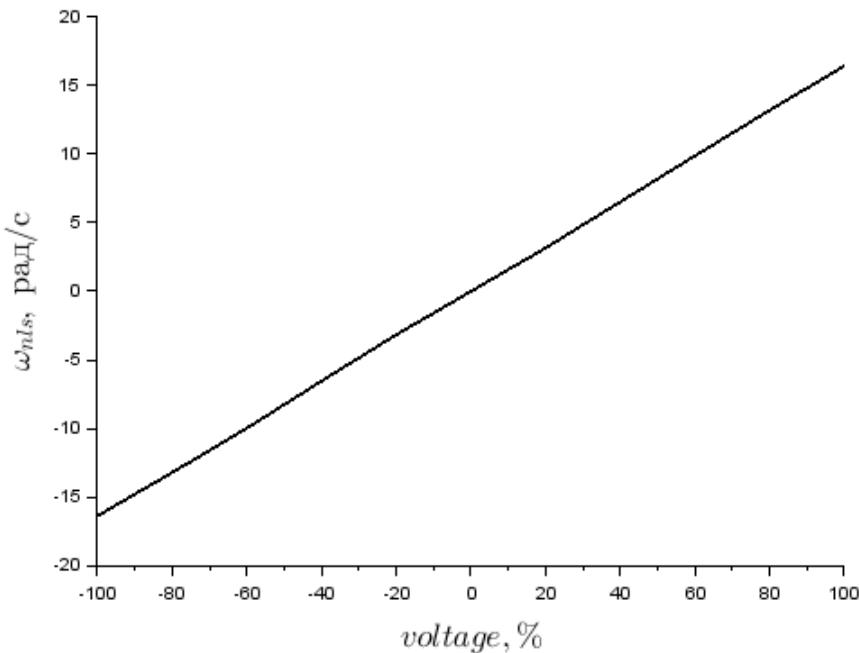


Рис. 14. График зависимости ω_{nls} (voltage).

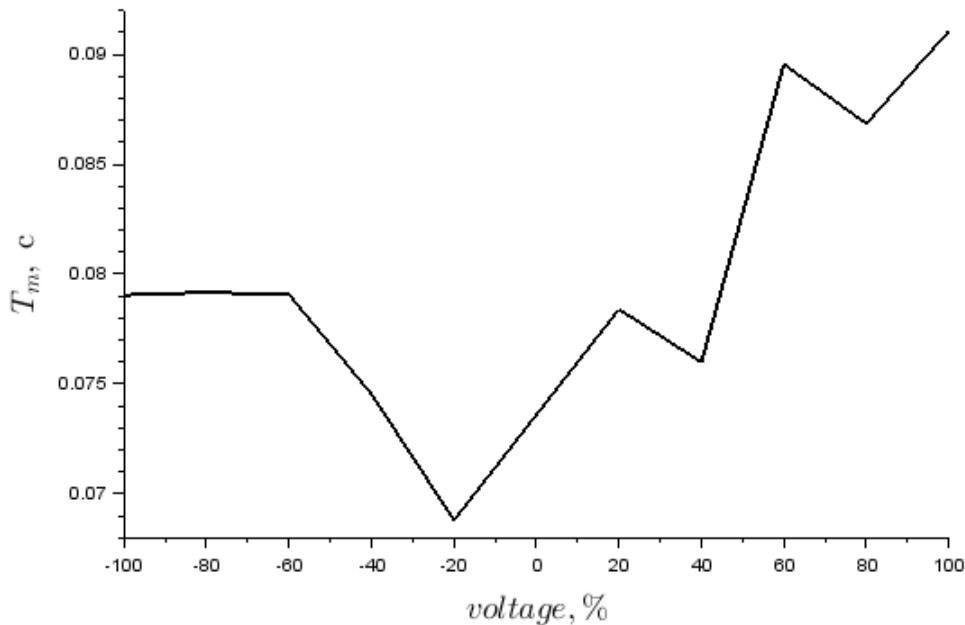


Рис. 15. График зависимости T_m (voltage).

5 Содержание отчета

В отчете обязательно должны присутствовать:

- 1 Графики и результаты расчетов величин ω_{nls} и T_m , предусмотренные пунктами 2 и 3 раздела «Порядок выполнения работы».
- 2 Исходный код основной расчетной программы, подготовленный с помощью редактора scinotes.
- 3 Построенная в Xcos схема моделирования работы двигателя NXT.

4 Исходный код написанной для NXT программы.

5 Выводы о проделанной работе.

6 Приложение А

Описание среды разработки Bricx Command Center

6.1 Общие сведения

Bricx Command Center представляет из себя среду разработки, позволяющую создавать программы для роботов серии LEGO Mindstorms, к которой относится и используемая в данном курсе модель NXT. Она не только дает возможность создавать программный код, но и умеет выполнять ряд других функций, незаменимых при использовании роботов LEGO, таких, как загрузка программ в блоки, работа с памятью последних и др. Внешний вид главного окна BricxCC показан на рис. 16.

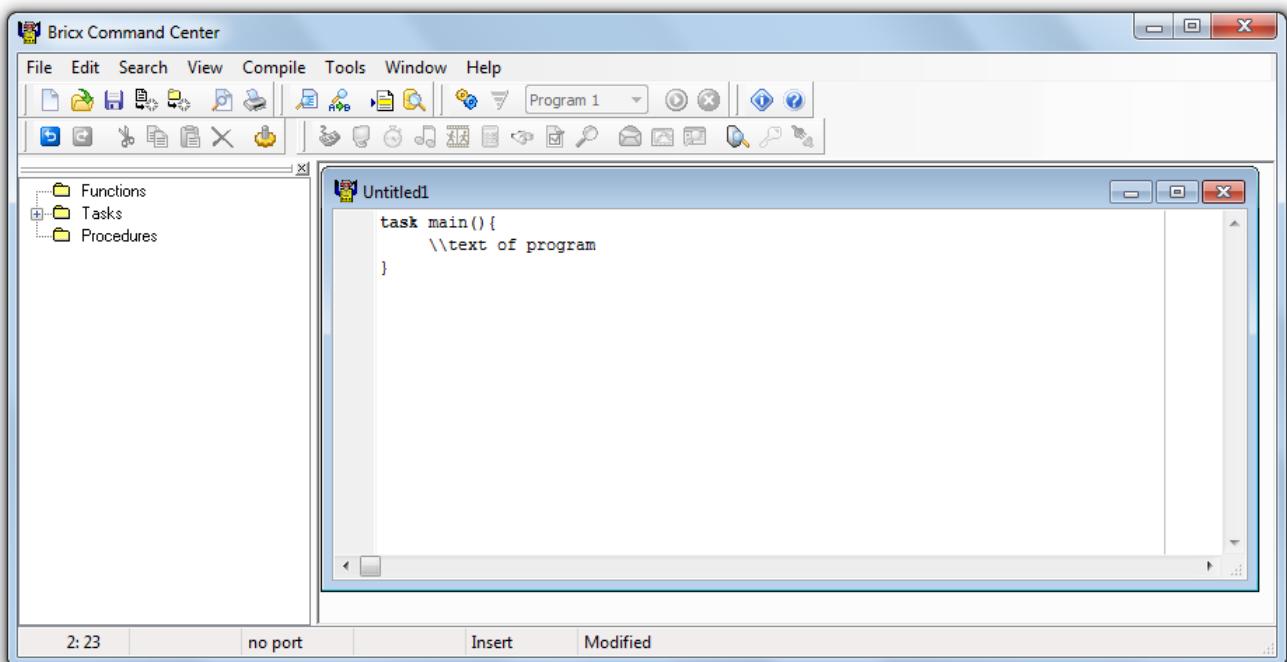


Рис. 16. Внешний вид среды BricxCC.

При выполнении данных лабораторных работ следует использовать язык программирования NXC. Он очень похож на стандартный язык C, поэтому структура программы и запись основных конструктивных составляющих (циклов, операторов, функций и т.д) во многих случаях оказываются либо совсем идентичными, либо отличающимися лишь в некоторых деталях. Например, в NXC основная часть программы в отличие от C должна быть заключена в конструкцию:

```
task main(){
    \\основная часть программы
}
```

6.2 Используемые функции

При написании программы для NXT используйте следующие функции¹⁷:

- `OnFwd(ports, voltage)`

Данная функция включает двигатели, подключенные к портам, указанным на месте `ports`, на движение в одну сторону.

Выходные порты указываются соответствующими заглавными буквами с префиксной конструкцией `OUT_`. Например, при подстановке в эту функцию выражения `OUT_AB`, в движение будут приведены двигатели, включенные в порты А и В. Аргумент `voltage` представляет из себя целое число, лежащее в промежутке от -100 до 100, и означает долю (в процентах) от максимального напряжения аккумулятора, которую необходимо подать на двигатель. Его знак определяет направление вращения двигателя.

- `MotorRotationCount(port)`

Возвращает текущее значение угла поворота ротора с датчика вращения мотора, подключенного к порту `port`.

- `ResetRotationCount(ports)`

Эта функция сбрасывает текущие показания датчиков вращения соответствующих моторов в значение 0.

- `CreateFile("file_name", file_size, file_handle)`

Создает в памяти блока NXT файл и открывает его для записи.

На месте `file_name` записывается имя будущего файла, например `Nums.txt`. На втором — его размер в байтах. На месте `file_handle` — имя переменной-идентификатора файла, ее следует объявить в программе отдельно, указав в качестве ее типа слово `byte`.¹⁸ По последней к созданному файлу можно будет обращаться из других функций.

- `DeleteFile("file_name")`

Функция удаляет из памяти NXT файл, носящий имя `file_name`.

Во избежание всяких проблем полезно вызывать данную функцию перед тем, как использовать прошлую.

¹⁷Классическое, грубо говоря, описание данных функций, указывающее типы аргументов и возвращаемых значений, можно посмотреть в справочной информации к программе, которая вызывается нажатием клавиши F1. Также там можно найти дополнительные примеры использования интересующих функций.

¹⁸Обратите внимание, в справке к языку NXC в качестве типа третьего аргумента данной функции приведена конструкция «`byte &`». Амперсанд, стоящий после имени типа, означает, что соответствующий аргумент *передается в функцию по ссылке*. Это, в свою очередь, говорит о том, что, во-первых, он может быть представлен только переменной, а во-вторых, что вызываемая функция может непосредственно ее изменять. Например, в случае с `CreateFile` так и происходит: в результате ее выполнения, переменная, указанная на месте `handle`, получает некоторое новое значение.

- `WriteLnString(file_handle, the_string, string_size)`

Записывает в файл `file_handle`¹⁹ строку `the_string` и присваивает переменной, указанной на месте `string_size`, значение, равное количеству байтов, занятых строкой `the_string` в файле. Также данная функция переводит указатель на новую строчку.

Здесь надо сказать, что одним из отличий языка C от NXС является наличие в последнем строкового типа данных, который носит имя `string`. Для данной лабораторной работы о переменных такого типа важно знать всего две вещи. Во-первых, надо понимать, что переменные типа `string` — это всего лишь один или несколько символов. Ниже дан пример, как в некоторой программе переменная этого типа (`s`) инициализируется значением `qwerty` (кавычками в программе обозначаются те совокупности символов, которые ей надо расценивать как значения типа `string`):

```
task main(){
    ...
    string s;
    ...
    s = "qwerty";
    ...
}
```

Во-вторых, следует учитывать, что к ним применима *конкатенация* — операция сложения. Пояснение к этому дается следующим примером (в результате его выполнения в переменной `s3` сохранится значение, равное `qwerty`, а в переменной `s2` — значение `tyqwer`):

```
task main(){
    ...
    string s1,s2,s3;
    ...
    s1 = "qwer";
    s2 = "ty";
    s3 = s1 + s2;
    s2 = s2 + s1;
    ...
}
```

- `CloseFile(file_handle)`

Закрывает файл с указанным идентификатором.

Данную функцию надо применить в самом конце программы, когда файл оказывается больше не нужным.

¹⁹В данном случае обращение к файлу осуществляется через его переменную-идентификатор, которая ранее по коду была указана в функции `CreateFile`.

- `NumToStr(number)`

Эта функция возвращает переданное ей числовое значение, переведенное из числового типа в строковый.

- `Wait(milliseconds)`

Осуществляет задержку в выполнении программы на указанное количество миллисекунд.

- `CurrentTick()`

Возвращает показание с внутренних «часов» NXT. Значение этих «часов» увеличивается на единицу через каждую 1/1000 секунды.

- `FirstTick()`

Возвращает то значение с встроенных «часов», которое соответствовало моменту запуска текущей программы.

6.3 Синхронизация блока NXT и ПК

Синхронизировать блок NXT и ПК предложит открытием соответствующего окошка (показано на рис. 17) сама BricxCC при ее запуске. При таком условии остается только выбрать подходящие значения из списков: список «Port» — usb, «Brick Type» — NXT, «Firmware» — узнать у преподавателя.

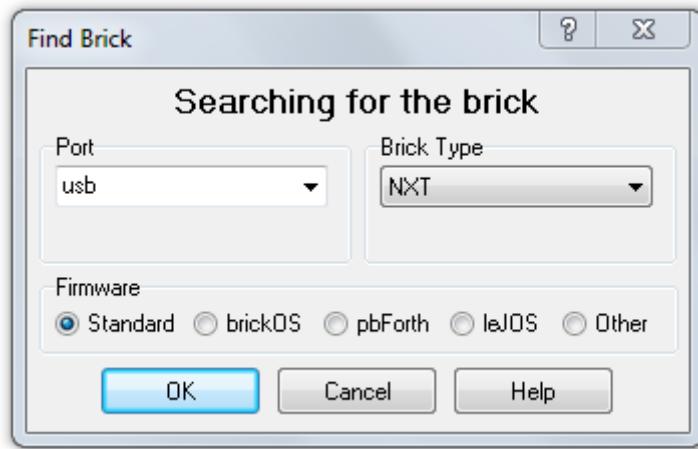


Рис. 17. Окно подключения блоков.

Если по каким-либо причинам при входе в среду подключить NXT к компьютеру не удалось, то это можно сделать пройдя по вкладке «Tools» главного меню и выбрав там пункт «Find brick». В результате этих действий появится уже упоминаемое окно.

6.4 Память NXT

Для просмотра и переноса файлов с памяти программируемого блока NXT используйте файловый менеджер BricxCC, который можно вызвать, пройдя по все той же вкладке «Tools» и выбрав пункт «NXT Explorer». Его внешний вид показан на рис. 18.

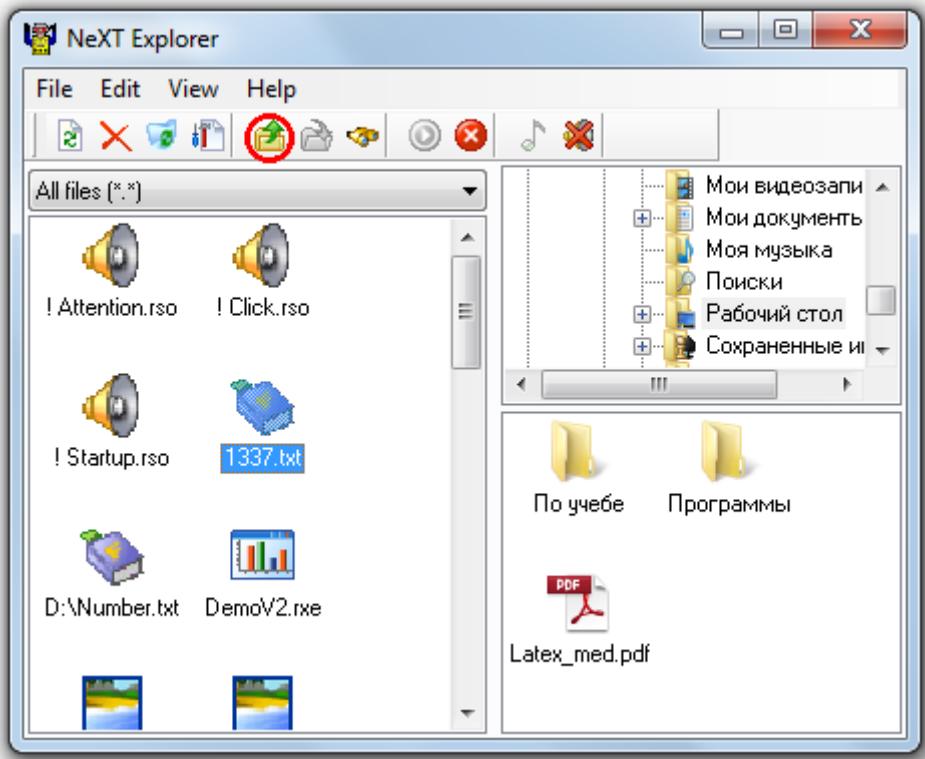


Рис. 18. Общий вид файлового менеджера BricxCC.

Чтобы скопировать необходимый файл на свой ПК, можно, например, кликнуть по нему единожды мышью, а затем нажать на символ, обведенный на рисунке красным кружком.