Московский государственный университет им. М. В. Ломоносова
Механико-математический факультет
Кафедра прикладной механики и управления

Курсовая работа

Анализ экспериментальных данных ходьбы экзоскелетона нижних конечностей

Выполнила студентка 3 курса
Липко Анфиса Игоревна
Научный руководитель:
в. н. с., к. ф-м. н. Буданов Владимир Михайлович

Москва
2022 год

Содержание

Введение

В данной курсовой работе будут рассмотрены алгоритмы управления приводами в коленном суставе экзоскелета с использованием информации, получаемой с датчиков. Рассматриваются проблемы организации движения экзоскелета как в режиме плоской регулярной ходьбы по ровной горизонтальной поверхности, так и при ходьбе с препятствиями (уступы, лестницы, нерегулярная поверхность). Экзоскелет снабжен двумя двигателями в коленных шарнирах, и фиксируется на человеке при помощи лямок, соединяющих его и аппарат в некоторых точках тела. При эксперименте будут фиксироваться данные с силовых датчиков в стопе и в нижнем шарнире (голени), угол сгиба колена, а также момент, создаваемый двигателем в колене. Также рассмотрено построение алгоритмов управления по показаниям силового датчика и датчика угла. Одной из задач является исследование возможностей оценки эффективности экзоскелета по показаниям всех датчиков.

Результаты обработки экспериментальных данных предполагается использовать для улучшения алгоритмов управления. При обработке данных будут использованы библиотеки с реализацией элементов искусственного интеллекта (scipy, numpy, pandas, scikit learn) для среды программирования python.

1. Постановка задачи

Рассматривается система: человек + экзоскелет нижних конечностей с двигателями в коленных суставах (Рис.1). Для системы требуется оценивать эффективность работы экзоскелета при разных алгоритмах управления, в частности изменения энергии человека. Перед данной курсовой работой поставлено две задачи. Первая: освоение динамической модели двуногой ходьбы, в частности, определение моментов, возникающих в суставах. Вторая: предварительный анализ данных, получаемых с датчиков экзоскелета в разных режимах движения и построение классификатора, различающего эти режимы.

Рис.1. Человек в исследуемом экзоскелете (а) и отдельный экзоскелет (б)

2. Модель и схема экзоскелета

2.1. Математическое описание модели ходьбы

Система «человек-экзоскелет» состоит из пяти весомых инерционных элементов: двух ног из двух звеньев и корпуса-балансира. Каждая из ног имеет бедро длины $2a$ и голень длины $2b$. В точке О подвеса ног к корпусу расположен также центр масс платформы («таз»), которая в данной схеме моделируется материальной точкой массы m_0.

Положение таза в пространстве задается двумя декартовыми координатами x, y, а положение ног и корпуса — угловыми координатами α_1, β_1, α_2, β_2, ψ. Система имеет 7 степеней свободы. В качестве обобщенных лагранжевых координат возьмем x, y, α_1, β_1, α_2, β_2, ψ.

Обозначим моменты сил управления в каждом колене аппарата за u_1, u_2, и моменты сил управления между бедрами и корпусом за q_1, q_2.

Цель создания экзоскелета — разгрузка человека при ходьбе и беге, в частности, уменьшение работы человека по созданию моментов в коленных суставах и возможность переносить тяжелые грузы на рычагах экзоскелета. В механической модели был реализован двигатель для создания дополнительных моментов в коленных суставах, для корпуса-балансира двигатель не реализовывался. Необходимый балансирующий момент при движении создаётся непосредственно человеком.

Также в фиксированной точке стопы приложена сила R_i, $i=1,2$, представляющая сумму всех сил, действующих на опорную точку (реакция опоры, сила трения-сцепления). Назовем R_i силами реакции.

Динамическая схема системы «человек-экзоскелет» изображена на рис.2. Подобная модель также была составлена в источниках [1], [2], [3]. Ниже приведены обозначения, подразумевающие следующие величины для системы человек-экзоскелет (массы суммируются, а длины звеньев экзоскелета регулируются и равны длинам звеньев человека).

Обозначения (введены в соответствии с работами Белецкого [1],[2])

m_0 — масса таза

M — масса корпуса-балансира

h — высота корпуса-балансира

r — расстояние от O до центра масс корпуса

J — момент инерции корпуса относительно оси z в т. O

$m_б$ — масса бедра

a — расстояние от O до центра масс бедра

$J_б$ — момент инерции бедра относительно оси z в т. O

$m_Г$ — масса голени

b — расстояние от колена до центра масс голени

$J_Г$ — момент инерции голени относительно оси z в т. колена

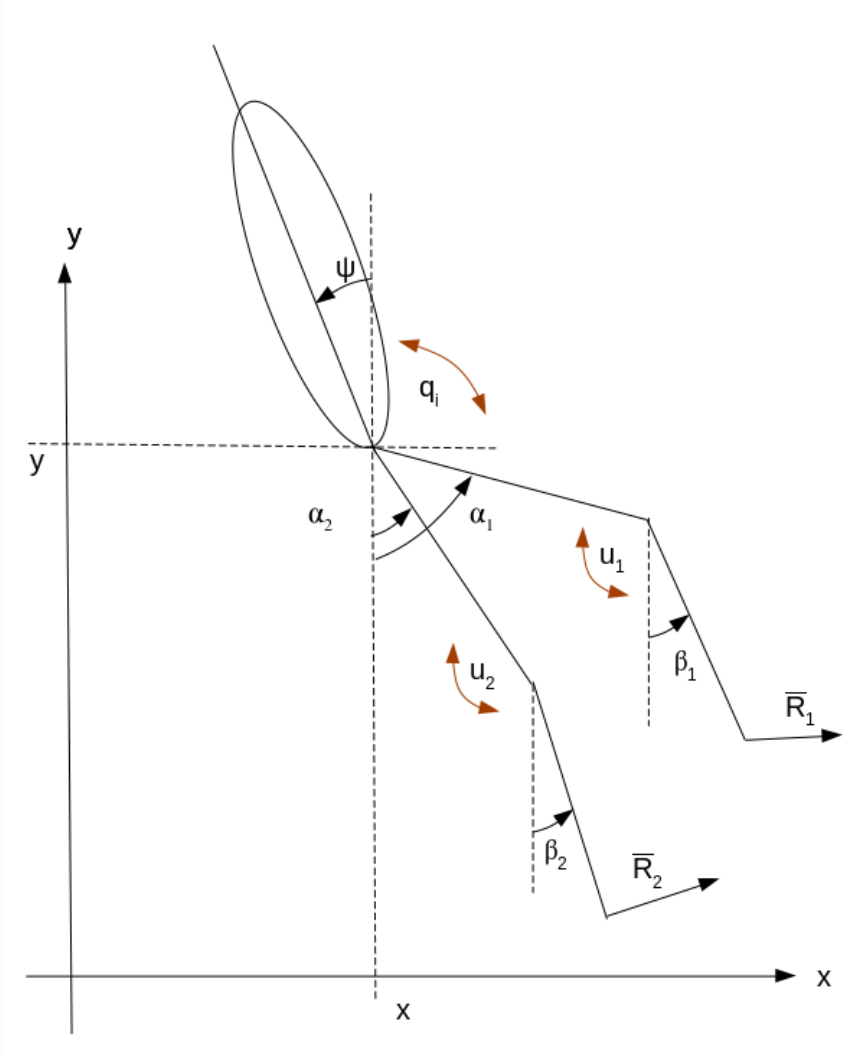


Рис.2. Динамическая схема аппарата

Добавим еще некоторые обозначения моментов сил и инерции:

$\underline{M} = m_0+2m_б+2m_Г+M$

$J_б^* = J_б+4m_Гa^2$

$K_б= (m_б+2m_Г)a$

$K_Г = m_Гb$

$J_{бГ} = 2m_Гab$

$K_T = M_T> 0$

В качестве уравнений движения рассмотрим уравнения Лагранжа 2 рода. Для вычисления кинетической энергии T системы используем формулу А.И. Лурье, согласно которой для каждого элемента массы M кинетическая энергия:

(1) $T = \frac{1}{2}\left[Mv_0^2 + 2M(\overrightarrow{v_0} * \vec{\omega})\overrightarrow{r_c'} + \vec{\omega}\mathring{\theta}\vec{\omega}\right]$

где:

$\overrightarrow{v_0}$ — скорость полюса O элемента

$\overrightarrow{r_c'}$ — радиус-вектор OC центра инерции C элемента в системе осей, имеющих начало в полюсе O

$\mathring{\theta}$ — тензор инерции элемента в точке O

Используя формулу (1), вычисляем кинетическую энергию рассматриваемой системы:

(2) $T = \frac{1}{2}[M(\dot{x}^2 + \dot{y}^2)] + \frac{1}{2}\left[J\dot{\psi}^2 - 2K_{\text{Т}}(\dot{x}cos\psi + \dot{y}sin\psi)\dot{\psi}\right] + \cdots$

$$\ldots + \frac{1}{2}\sum_{i=1}^{2}\Big[J_б\ \dot{\alpha_i}^2 + J_Г\dot{\beta_i}^2 + 2K_б\dot{\alpha_i}(\dot{x}cos\alpha_i + \dot{y}sin\alpha_i) + 2K_Г\dot{\beta_i}(\dot{x}cos\beta_i + \dot{y}sin\beta_i) + 2J_{бГ}\alpha_i\beta_i cos(\alpha_i - \beta_i)\Big]$$

Силовая функция (отрицательная потенциальная энергия) имеет вид:

(3) $U = -g[My + K_{\text{Т}}cos\psi - \sum_{i=1}^{2}[K_б cos\alpha_i + K_Г cos\beta_i]]$

Обозначим через Q обобщенные силы, q — обобщенные координаты, $\dot{q}$ — обобщенные скорости, и составим уравнения Лагранжа 2 рода:

(4) $\frac{d}{dt}\left(\frac{\partial T}{\partial \dot{q}}\right) - \frac{\partial T}{\partial q} = \frac{\partial U}{\partial q} + Q$

Имеем для 7 обобщенных координат 7 дифференциальных уравнений. Напомним, что положение платформы (O) в пространстве задается двумя

декартовыми координатами x, y, положение ног — угловыми координатами α_1, β_1, α_2, β_2, а положение корпуса — угловой координатой ψ.

К решению уравнений (4) применяют полуобратный метод — частично задаются координаты как функции времени, иными словами, задается траектория движения. После чего координаты дифференцируются и подставляются в уравнения Лагранжа 2 рода для поиска моментов.

Отметим, что моменты в этой модели, как и массовые характеристики, являются суммами моментов, созданных совместно человеком и двигателем экзоскелета.

2.2. Модель одноопорной ходьбы

Ходьба — периодическое движение с разными фазами опоры на каждую из ног по очереди. Движение человека можно условно разбить на две фазы [4]. В каждом случае модель имеет разное количество степеней свободы, поэтому каждая из фаз будет характеризоваться своим набором уравнений.

Первая фаза — двухопорная. Характеризуется наличием точек контакта с поверхностью в обеих ногах, передняя и задняя ноги опираются на поверхность. Фаза характеризуется 3 степенями свободы (одна пара углов сгиба для одной ноги однозначно задает положение второй ноги и положение таза, а еще одна степень свободы отвечает за наклон туловища).

Вторая фаза — одноопорная. Одна нога находится в фазе переноса, другая — в фазе опоры. У системы — 5 степеней свободы.

В исследуемом нами движении преобладает именно одноопорная ходьба. В работе ограничимся рассмотрением только второй фазы, поскольку в этом случае удается привести уравнения к достаточно простому и симметричному виду. Кроме того, эта фаза занимает около 60% от всего цикла ходьбы, и зачастую цикл моделируют как последовательность периодов опоры на одну ногу с мгновенной сменой ног.

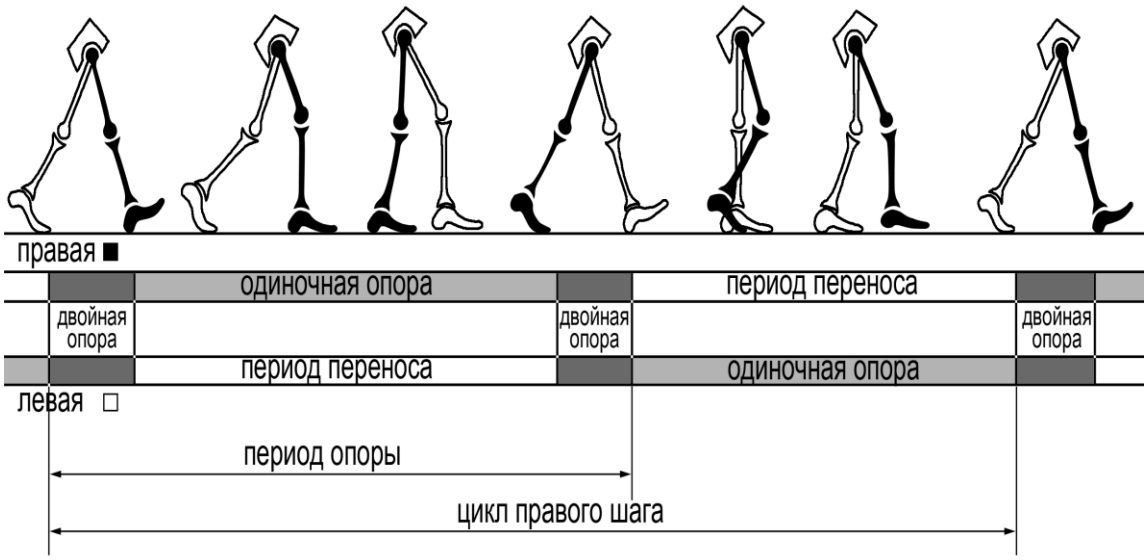


Рис.3. Цикл ходьбы человека [4], [5]

Следовательно, мы уменьшили число степеней свободы до 5. Если заданы α_i, β_i, то по опорной ноге однозначно определяется положение таза x, y.

Если мы обозначим через 1 индекс переносимой ноги, и через 2 индекс опорной ноги, то $R_{1x}=0$, $R_{1y}=0$ — условие отсутствия силы реакции у переносимой ноги.

Перейдем к записи уравнений Лагранжа. Искомыми являются 7 функций: u_i, q_i, ψ, R_{Sx}, R_{Sy}, i={1,2}. С помощью 7 уравнений их определить можно, следовательно, задача определена.

Упростим выражение кинетической энергии T_i из теоремы Кёнига:

(5) $T_i = \frac{m_i v_{ci}^2}{2} + \frac{J_i \dot{\varphi_i}^2}{2} = \frac{m_i v_{ci}^2}{2} + \frac{m_i l_i^2 \dot{\varphi_i}^2}{24}$

где m_i и l_i — масса и момент инерции i-го стержня, $v_{ci}^2 = \left(\dot{x_{ci}}^2 + \dot{y_{ci}}^2\right)$, x_{ci} и y_{ci} — координаты центра масс i-го стержня. Этот центр условно помещается на расстоянии половины длины стержня от точки шарнирного соединения.

Обозначим индексы: т — туловище, б1 — бедро первой ноги, г1 — голень первой ноги, б2 — бедро второй ноги, г2 — голень второй ноги.

Потенциальная энергия перенесена в правую часть уравнений вместе с обобщенными силами. Таким образом, полученные уравнения в векторно-матричной форме будут включать только вторые производные и первые производные в квадрате [2, с.32] и будут выглядеть следующим образом:

(6) $A(\phi)\ddot{\phi} + B(\phi)\dot{\phi}_2 = Q(\phi)$

где $\phi=(\varphi_1, \dots, \varphi_5)^T=(\psi, \alpha_1, \alpha_2, \beta_1, \beta_2)^T$, $\dot{\phi}_2 = \left(\dot{\varphi_1}^2, \dots, \dot{\varphi_5}^2\right)^T$, $\dim(A) = 5\times5$.

Матрицы A и B были найдены в рамках нашей модели с помощью выводов Формальского А.М. [6] и представлены ниже.

Матрица A:

$J_т + 0.25\ m_т\ l_т^2$	0	$0.5m_т\ l_т\ l_{б2} \cos(\psi-\alpha_2)$	0	$0.5m_т\ l_т\ l_{г2} \cos(\psi-\beta_2)$
0	$J_{б1} + l_{б1}^2\ (0.25m_{б1} + m_{г1})$	$(m_{г1} + 0.5m_{б1})\ l_{б1}\ l_{б2} \cos(\alpha_1-\alpha_2)$	$(0.5m_{г1})l_{г1}\ l_{б1} \cos(\alpha_1-\beta_1)$	$(m_{г1} + 0.5m_{б1})\ l_{б1}\ l_{г2} \cos(\alpha_1-\beta_2)$
$0.5m_т\ l_т\ l_{б2} \cos(\alpha_2 - \psi)$	$(m_{г1} + 0.5m_{б1})\ l_{б1}\ l_{б2} \cos(\alpha_2 - \alpha_1)$	$J_{б2} + l_{б2}^2\ (m_{г1} + m_{б1} + m_т + 0.25m_{б2})$	$(0.5m_{г1})l_{г1}\ l_{б2} \cos(\alpha_2-\beta_1)$	$(m_{г1} + m_{б1} + m_т + 0.25m_{б2})\ l_{г2}\ l_{б2} \cos(\alpha_2 - \beta_2)$
0	$(0.5m_{г1})l_{г1}\ l_{б1} \cos(\beta_1 - \alpha_1)$	$(0.5m_{г1})l_{г1}\ l_{б2} \cos(\beta_1-\alpha_2)$	$J_{г1} + l_{г1}^2\ (0.25m_{г1})$	$(0.5m_{г1})l_{г1}\ l_{г2} \cos(\beta_1-\beta_2)$
$0.5m_т\ l_т\ l_{г2} \cos(\beta_2 - \psi)$	$(m_{г1} + 0.5m_{б1})\ l_{б1}\ l_{г2} \cos(\beta_2 - \alpha_1)$	$(m_{г1} + m_{б1} + m_т + 0.25m_{б2})\ l_{г2}\ l_{б2} \cos(\beta_2 - \alpha_2)$	$(0.5m_{г1})l_{г1}\ l_{г2} \cos(\beta_2-\beta_1)$	$J_{г2} + l_{г2}^2(m_{б2} + m_{г1} + m_т + 0.25m_{г2})$

Матрица B:

0	0	$0.5m_т\ l_т\ l_{б2} \sin(\psi-\alpha_2)$	0	$0.5m_т\ l_т\ l_{г2} \sin(\psi-\beta_2)$
0	0	$(m_{г1} + 0.5m_{б1})\ l_{б1}\ l_{б2} \sin(\alpha_1-\alpha_2)$	$(0.5m_{г1})l_{г1}\ l_{б1} \sin(\alpha_1-\beta_1)$	$(m_{г1} + 0.5m_{б1})\ l_{б1}\ l_{г2} \sin(\alpha_1-\beta_2)$
$0.5m_т\ l_т\ l_{б2} \sin(\alpha_2 - \psi)$	$(m_{г1} + 0.5m_{б1})\ l_{б1}\ l_{б2} \sin(\alpha_2 - \alpha_1)$	0	$(0.5m_{г1})l_{г1}\ l_{б2} \sin(\alpha_2-\beta_1)$	$(m_{г1} + m_{б1} + m_т + 0.25m_{б2})\ l_{г2}\ l_{б2} \sin(\alpha_2 - \beta_2)$
0	$(0.5m_{г1})l_{г1}\ l_{б1} \sin(\beta_1 - \alpha_1)$	$(0.5m_{г1})l_{г1}\ l_{б2} \sin(\beta_1-\alpha_2)$	0	$(0.5m_{г1})l_{г1}\ l_{г2} \sin(\beta_1-\beta_2)$
$0.5m_т\ l_т\ l_{г2} \sin(\beta_2 - \psi)$	$(m_{г1} + 0.5m_{б1})\ l_{б1}\ l_{г2} \sin(\beta_2 - \alpha_1)$	$(m_{г1} + m_{б1} + m_т + 0.25m_{б2})\ l_{г2}\ l_{б2} \sin(\beta_2 - \alpha_2)$	$(0.5m_{г1})l_{г1}\ l_{г2} \sin(\beta_2-\beta_1)$	0

Как и было отмечено, матрицы действительно симметричны, в записи отличаются только знаки косинусов (косинус — четная функция).

Хотим вычислить моменты, действующие в системе (u_i, q_i).

Выражение для элементарной работы всех сил:

(7) $\delta W = (q_1 + q_2)\delta\psi + (q_1 - u_1)\delta\alpha_1 + (q_2 - u_2)\delta\alpha_2 + (u_1)\delta\beta_1 + (u_2)\delta\beta_2$

Так как действие происходит в поле силы тяжести, то можно написать выражение для потенциальной энергии:

(8) $П = \sum_{i=1}^{5} П_i$, где $П_i = m_i g y_{ci}$

Предполагая, что таз движется на постоянной высоте, можно записать выражение для потенциальной энергии каждого звена относительно таза.

$$\Pi_{\psi} = Mghcos\psi$$

$$\Pi_{\alpha_1} = -m_{Б}gacos\alpha_1$$

$$\Pi_{\beta_1} = -m_{Г}g(bcos\beta_1 + 2acos\alpha_1)$$

$$\Pi_{\alpha_2} = -m_{Б}gacos\alpha_2$$

$$\Pi_{\beta_2} = -m_{Г}g(bcos\beta_2 + 2acos\alpha_2)$$

Обобщенные силы тогда будут выглядеть следующим образом:

$$Q_T = q_1 + q_2 - \frac{\partial \Pi}{\partial \psi} = q_1 + q_2 + Mghsin\psi$$

$$Q_{б1} = q_1 - u_1 - \frac{\partial \Pi}{\partial \alpha_1} = q_1 - u_1 - m_{Б}gasin\alpha_1 - 2m_{Г}gasin\alpha_1$$

$$Q_{б2} = q_2 - u_2 - \frac{\partial \Pi}{\partial \alpha_2} = q_2 - u_2 - m_{Б}gasin\alpha_2 - 2m_{Г}gasin\alpha_2$$

$$Q_{Г1} = q_1 - \frac{\partial \Pi}{\partial \beta_1} = q_1 - m_{Г}gbsin\beta_1$$

$$Q_{Г2} = q_2 - \frac{\partial \Pi}{\partial \beta_2} = q_2 - m_{Г}gbsin\beta_2$$

Выпишем уравнения для каждой координаты в явном виде.

(9.1)

$(J_т + 0.25\ m_т\ l_т^2 + 0.5m_т\ l_т\ l_{б2}\cos(\alpha_2 — \psi) + 0.5m_т\ l_т\ l_{г2}\cos(\beta_2 — \psi))\ \ddot{\psi} +$

$+ (0.5m_т\ l_т\ l_{б2}\sin(\alpha_2 — \psi) + 0.5m_т\ l_т\ l_{г2}\sin(\beta_2 - \psi))\ \dot{\psi}^2 = q_1 + q_2 - \frac{\partial \Pi}{\partial \psi}$

(9.2)

$(J_{б1} + l_{б1}^2\ (0.25m_{б1} + m_{г1}) + (m_{г1} + 0.5m_{б1})\ l_{б1}\ l_{б2}\cos(\alpha_2 - \alpha_1) + (0.5m_{г1})l_{г1}\ l_{б1}\cos(\beta_1 - \alpha_1) +$

$+ (m_{г1} + 0.5m_{б1})\ l_{б1}\ l_{г2}\cos(\beta_2 - \alpha_1))\ \ddot{\alpha}_1 +$

$+ ((m_{г1} + 0.5m_{б1})\ l_{б1}\ l_{б2}\sin(\alpha_2 - \alpha_1) + (0.5m_{г1})l_{г1}\ l_{б1}\sin(\beta_1 - \alpha_1) +$

$+ (m_{г1} + 0.5m_{б1})\ l_{б1}\ l_{г2}\sin(\beta_2 - \alpha_1))\ \dot{\alpha}_1^2 = q_1 - u_1 - \frac{\partial \Pi}{\partial \alpha_1}$

(9.3)

$(0.5m_т\ l_т\ l_{б2}\cos(\psi-\alpha_2) + (m_{г1} + 0.5m_{б1})\ l_{б1}\ l_{б2}\cos(\alpha_1 — \alpha_2) + J_{б2} + l_{б2}^2\ (m_{г1} +$

$+ m_{б1} + m_т + 0.25m_{б2}) + (0.5m_{г1})l_{г1}\ l_{б2}\cos(\beta_1 — \alpha_2) + (m_{г1} + m_{б1} + m_т +$

$+ 0.25m_{б2})\ l_{г2}\ l_{б2}\cos(\beta_2 - \alpha_2))\ \ddot{\alpha}_2 +$

$+ (0.5m_т\ l_т\ l_{б2}\sin(\psi-\alpha_2) + (m_{г1} + 0.5m_{б1})\ l_{б1}\ l_{б2}\sin(\alpha_1 — \alpha_2) + (0.5m_{г1})l_{г1}\ l_{б2}\sin(\beta_1 — \alpha_2) +$

$+ (m_{г1} + m_{б1} + m_т + 0.25m_{б2})\ l_{г2}\ l_{б2}\sin(\beta_2 - \alpha_2))\ \dot{\alpha}_2^2 = q_2 - u_2 - \frac{\partial \Pi}{\partial \alpha_2}$

(9.4)

$((0.5m_{г1})l_{г1}\ l_{б1}\cos(\alpha_1 — \beta_1) + (0.5m_{г1})l_{г1}\ l_{б2}\cos(\alpha_2 — \beta_1) + J_{г1} + l_{г1}^2\ (0.25m_{г1}) +$

$+ (0.5m_{г1})l_{г1}\, l_{г2} \cos(\beta_2 - \beta_1))\ddot{\beta}_1 +$

$+ ((0.5m_{г1})l_{г1}\, l_{б1} \sin(\alpha_1 - \beta_1) + (0.5m_{г1})l_{г1}\, l_{б2} \sin(\alpha_2 - \beta_1) +$

$+ (0.5m_{г1})l_{г1}\, l_{г2} \sin(\beta_2 - \beta_1))\dot{\beta}_1^2 = q_1 - \frac{\partial \Pi}{\partial \beta_1}$

(9.5)

$(0.5m_т\, l_т\, l_{г2} \cos(\psi - \beta_2) + (m_{г1} + 0.5m_{б1})\, l_{б1}\, l_{г2} \cos(\alpha_1 - \beta_2) + (m_{г1} + m_{б1} + m_т +$

$+ 0.25m_{б2})\, l_{г2}\, l_{б2} \cos(\alpha_2 - \beta_2) + (0.5m_{г1})l_{г1}\, l_{г2} \cos(\beta_1 - \beta_2) + J_{г2} + l_{г2}^2(m_{б2} + m_{г1} + m_т + 0.25m_{г2}))\ddot{\beta}_2$

$+$

$+ (0.5m_т\, l_т\, l_{г2} \sin(\psi - \beta_2) + (m_{г1} + 0.5m_{б1})\, l_{б1}\, l_{г2} \sin(\alpha_1 - \beta_2) + (m_{г1} + m_{б1} + m_т +$

$+ 0.25m_{б2})\, l_{г2}\, l_{б2} \sin(\alpha_2 - \beta_2) + (0.5m_{г1})l_{г1}\, l_{г2} \sin(\beta_1 - \beta_2))\dot{\beta}_2^2 = q_2 - \frac{\partial \Pi}{\partial \beta_2}$

Отсюда, задавая законы движения каждой из точек, можно получить выражения для моментов. Законы движения могут быть заданы как аналитически, так и получены из экспериментальных данных. Но так как у нас в экзоскелете всего по одному датчику угла в каждой ноге, который замеряет угол $\Omega_i = 180 - \alpha_i + \beta_i$, то придется применять первый метод.

Положим, индекс 1 у той ноги, которая в начале цикла опорная, а затем — переносимая. Комфортабельность ходьбы, согласно В. В. Белецкому [7], объединяет два условия, предъявляемых к движению: во-первых, точка подвеса ног должна двигаться на постоянной высоте от поверхности; во-вторых, движение ее осуществляется с постоянной скоростью. Аналитический вид этих условий:

(10)

$$y - y_1^0 = h$$

$$x - x_1^0 = V(t - t_1) - s$$

где x,y — координаты точки подвеса ног, x_1^0, y_1^0 — координаты точки опоры, h — высота точки подвеса ног над поверхностью, V — скорость движения человека, t_1 — момент начала одноопорной фазы, s — опорный сдвиг (расстояние от точки опоры до проекции точки подвеса ног на горизонтальную плоскость в момент начала одноопорной фазы).

Для остальных обобщенных координат зададим траектории согласно выводам Формальского А.М. (численное моделирование одноопорной фазы, [6]) и предположениям Гурфинкеля В.С. о динамике равновесия вертикальной ходьбы [10]. Здесь параметры системы были следующие:

(11)

$\underline{M}$ = 75 кг (масса всего аппарата с человеком)

r = 0.386 м (расстояние от О до центра масс корпуса)

J = 11.3 кг*м^2 (момент инерции корпуса относительно оси z в т. О)

$m_б$ = 8.6 кг (масса бедра)

a = 0.18 м (расстояние от О до центра масс бедра)

$J_б$ = 0.535 кг*м^2 (момент инерции бедра относительно оси z в т. О

$m_Г$ = 4.6 кг (масса голени)

b = 0.324 м (расстояние от колена до центра масс голени)

$J_Г$ = 1.02 кг*м^2 (момент инерции голени относительно оси z в т. колена)

Методами линеаризации системы в постановках краевой задачи с помощью численного решения в [6] были получены следующие зависимости:

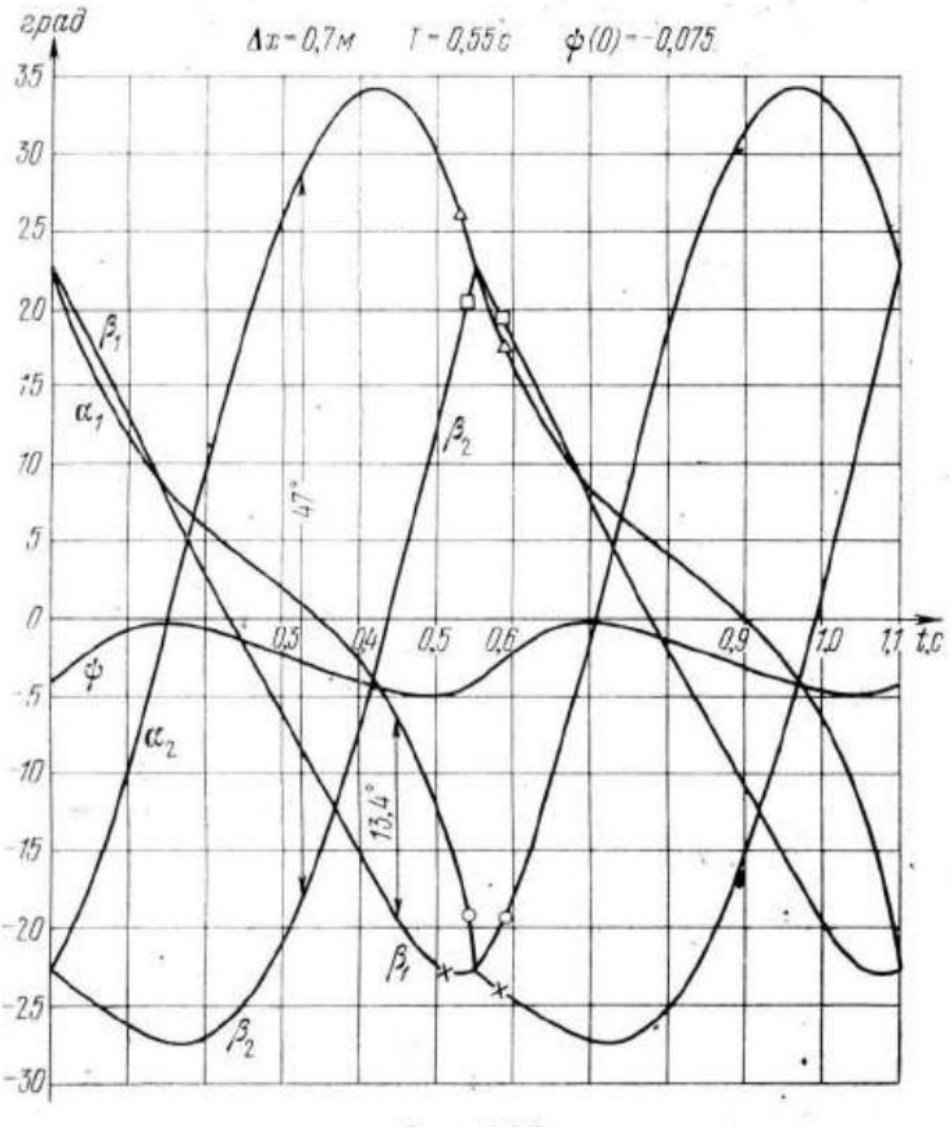


Рис.4а. Графики изменения углов в градусах для численного моделирования одноопорной ходьбы [6]

График периодический, петлеобразный, содержит один шаг в одноопорной фазе. Индекс «1» у опорной ноги, «2» — у переносимой. В краевых точках ($\alpha_1 \rightarrow \beta_1$, $\alpha_2 \rightarrow \beta_2$) — это означает, что в конце и в начале шага поставлены краевые условия на то, что нога выпрямлена. Также отметим, что угол отклонения корпуса имеет двойной период по сравнению с периодом изменения углов между звеньями и вертикалью (такой же результат отмечен и в [8]). Результат на рисунке 4б использован для проверки представленной модели. График используется для аппроксимации нужных зависимостей.

Движение имеет период T = 1.1с. Тогда круговая частота:

$$\omega = \frac{2\pi}{T} \simeq 5.7\text{Гц}$$

Используя метод разложения в ряд Фурье по синусам и косинусам до 2го порядка, подставляя некоторые точки на графике (по 12 точек на каждом), мы получили свои численные решения для координат(12).

(12)

α_1= *11.4sin(ωt) +14.5cos(ωt)-2.1sin(2ωt) +8.2cos(2ωt)*

β_1= *-11.8sin(ωt) +22.4cos(ωt)-2.8sin(2ωt) -0.4cos(2ωt)*

α_2= *8.4sin(ωt) -14.5cos(ωt) -5.6sin(2ωt) -8.5cos(2ωt)*

β_2= *-7.7sin(ωt) -24cos(ωt) +0.5sin(2ωt) +1.4cos(2ωt)*

ψ= *-2.3+1.7sin(2ωt) -1.5cos(2ωt)*

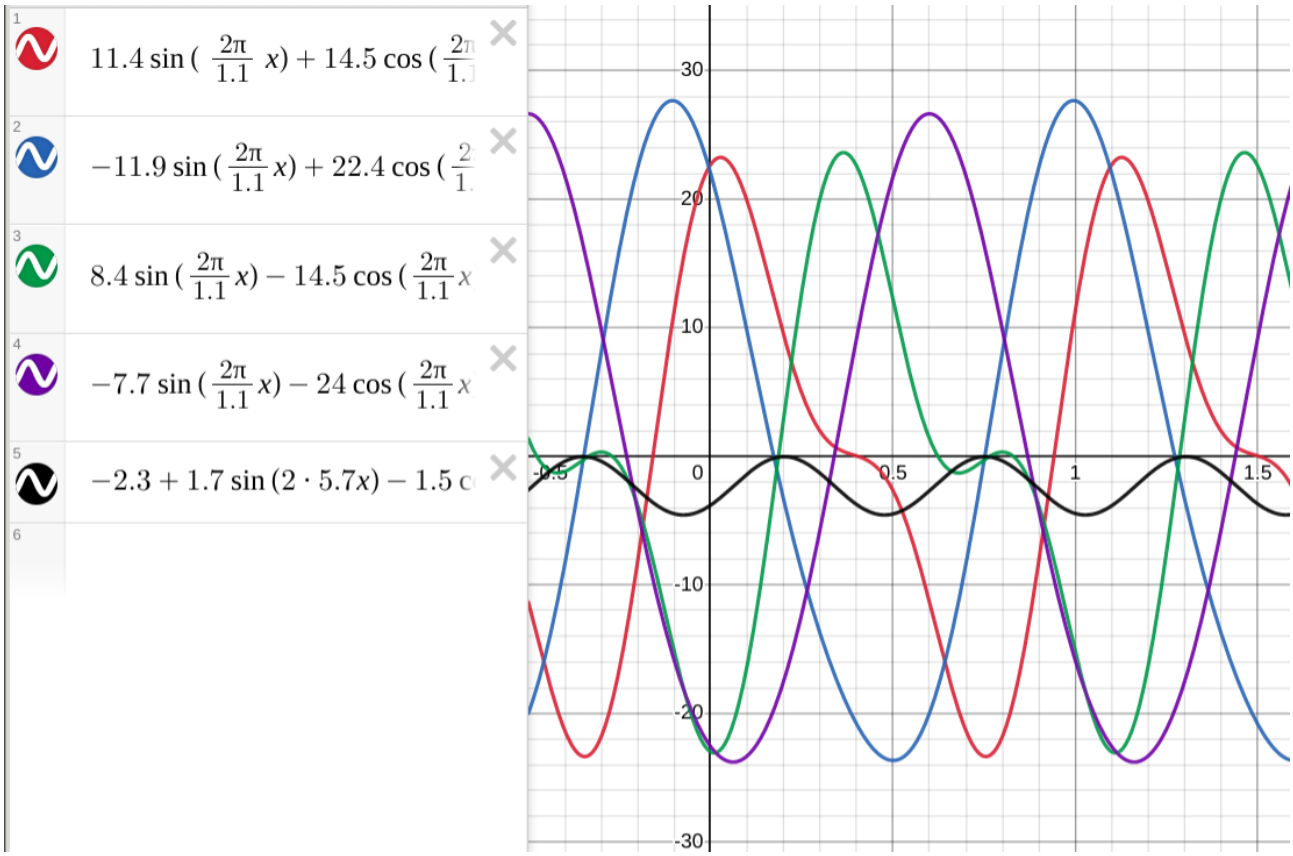


Рис.4б. Результат аппроксимации графиков изменения углов в градусах

Дифференцируем координаты по времени.

$\dot{\alpha}_1$= *11.4ωcos(ωt) -14.5ωsin(ωt)-4.2ωcos(2ωt) -16.4ωsin(2ωt)*

$\dot{\beta}_1$= *-11.8ωcos(ωt) -22.4ωsin(ωt)-5.6ωcos(2ωt) +0.8ωsin(2ωt)*

$\dot{\alpha}_2$= *8.4ωcos(ωt) +14.5ωsin(ωt) -11.2ωcos(2ωt) -17ωsin(2ωt)*

$\dot{\beta}_2$= *-7.7ωcos(ωt) +24ωsin(ωt) +ωcos(2ωt) +2.8ωsin(2ωt)*

$\dot{\psi} = 1.7\omega\cos(2\omega t) + 1.5\omega\sin(2\omega t)$

$\ddot{\alpha}_1 = -11.4\omega^2\sin(\omega t) - 14.5\omega^2\cos(\omega t) + 8.4\omega^2\sin(2\omega t) - 32.8\omega^2\cos(2\omega t)$

$\ddot{\beta}_1 = 11.8\omega^2\sin(\omega t) - 22.4\omega^2\cos(\omega t) + 11.2\omega^2\sin(2\omega t) + 1.6\omega^2\cos(2\omega t)$

$\ddot{\alpha}_2 = -8.4\omega^2\sin(\omega t) + 14.5\omega^2\cos(\omega t) + 22.4\omega^2\sin(2\omega t) - 34\omega^2\cos(2\omega t)$

$\ddot{\beta}_2 = 7.7\omega^2\sin(\omega t) + 24\omega^2\cos(\omega t) - 2\omega^2\sin(2\omega t) + 5.6\omega^2\cos(2\omega t)$

$\ddot{\psi} = -3.4\omega^2\sin(2\omega t) + 3\omega^2\cos(2\omega t)$

Подставляя результаты (12) и остальные численные значения (10) в (9.1 — 9.5) с помощью ПО Wolfram Mathematica 12, получим численные значения для моментов. Их также разложили до второй гармоники в ряд Фурье.

(13)

$q_1 = 96 + 35\sin(\omega t) + 15\cos(\omega t) - 2\sin(2\omega t) + 2\cos(2\omega t)$

$q_2 = 96 - 35\sin(\omega t) - 15\cos(\omega t) - 2\sin(2\omega t) + 2\cos(2\omega t)$

$u_1 = 175 - 57\sin(\omega t) - 50\cos(\omega t) + 5\sin(2\omega t) - 10\cos(2\omega t)$

$u_2 = -127 - 85\sin(\omega t) + 70\cos(\omega t) + 50\sin(2\omega t) - 31\cos(2\omega t)$

Прокомментируем зависимости моментов от времени. По модулю моменты q_i не превышают 140 Н*м, u_i не превышают 350 Н*м. Это соответсвует выводам из статьи Лавровского Э.К. [8] и статьи Антипова В.М. об энергозатратах [9]. По графикам видно, что примерно там, где для момента опорной ноги будет минимум, для переносной будет максимум, что также сопоставимо с имеющимися результатами. Все графики периодичны с периодом ω. Графики моментов в тазобедренном суставе (q_1 и q_2, возникающие между корпусом и ногами) получаются сдвигом друг относительно друга на половину периода — это подтверждает корректность решений уравнений. Графики моментов в коленном суставе различаются знаками за счет того, что в момент ходьбы обе ноги составляют противоположные по знаку углы с вертикалью и совершают сгибание-разгибание в противоположном порядке.

Ниже представлены изображения графики функций моментов от времени.

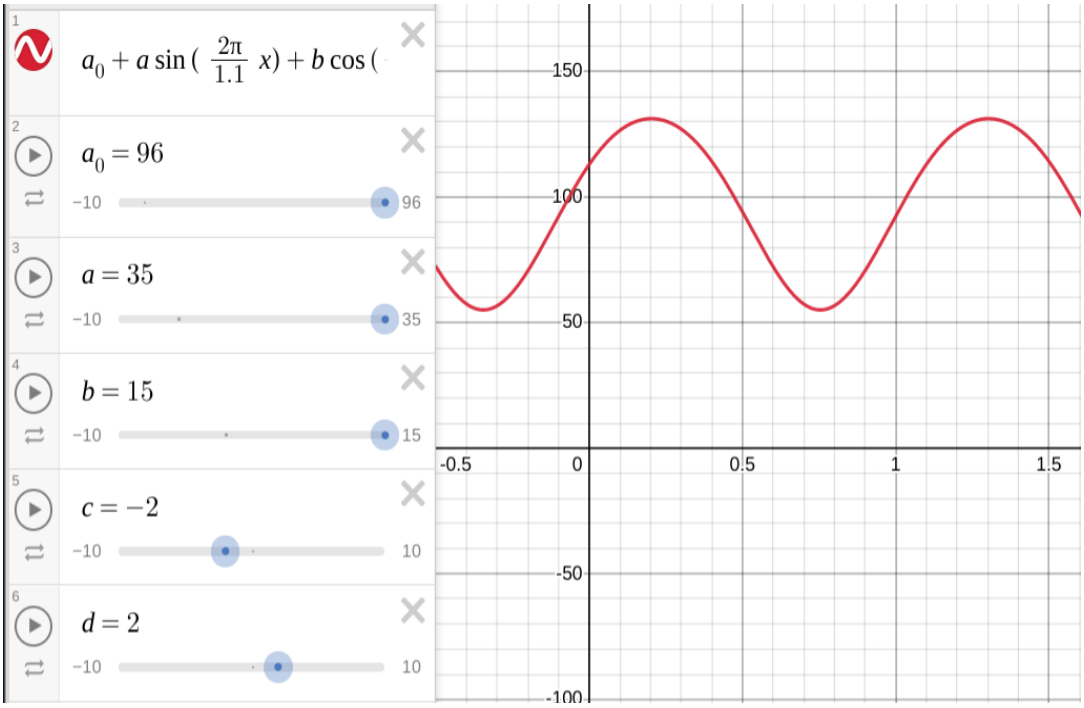


Рис.5а. Периодическая зависимость момента q_1 от времени в течение нескольких шагов с периодом 1.1 с.

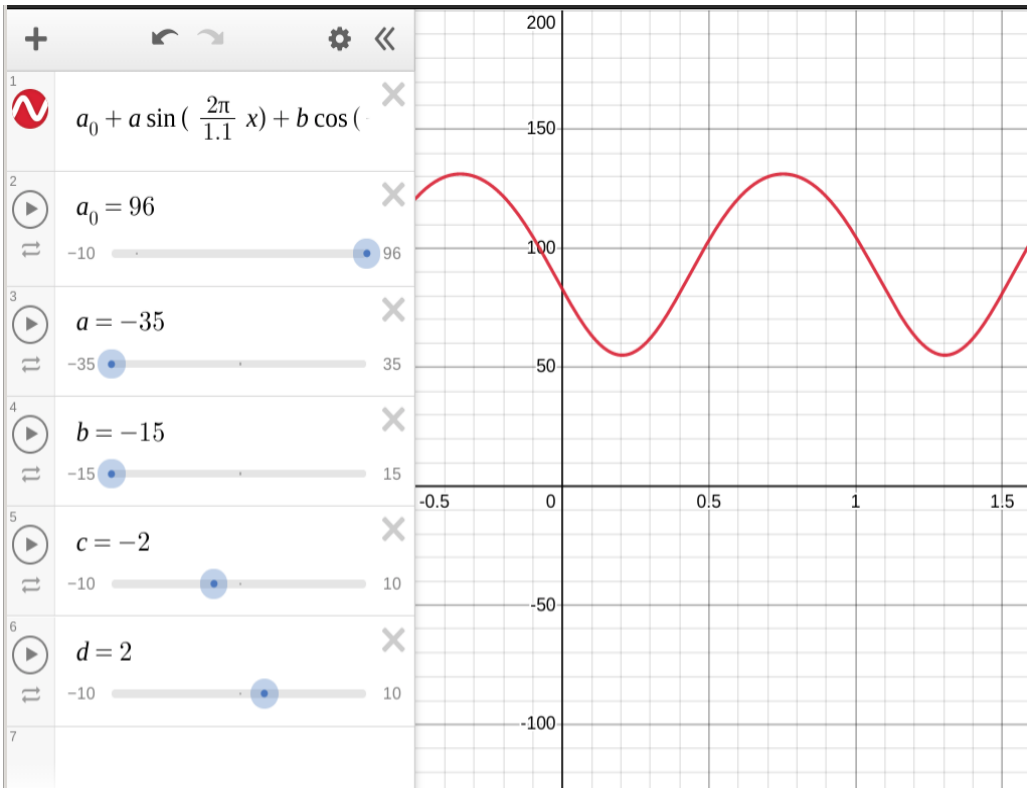


Рис.5б. Периодическая зависимость момента q_2 от времени в течение нескольких шагов с периодом 1.1 с.

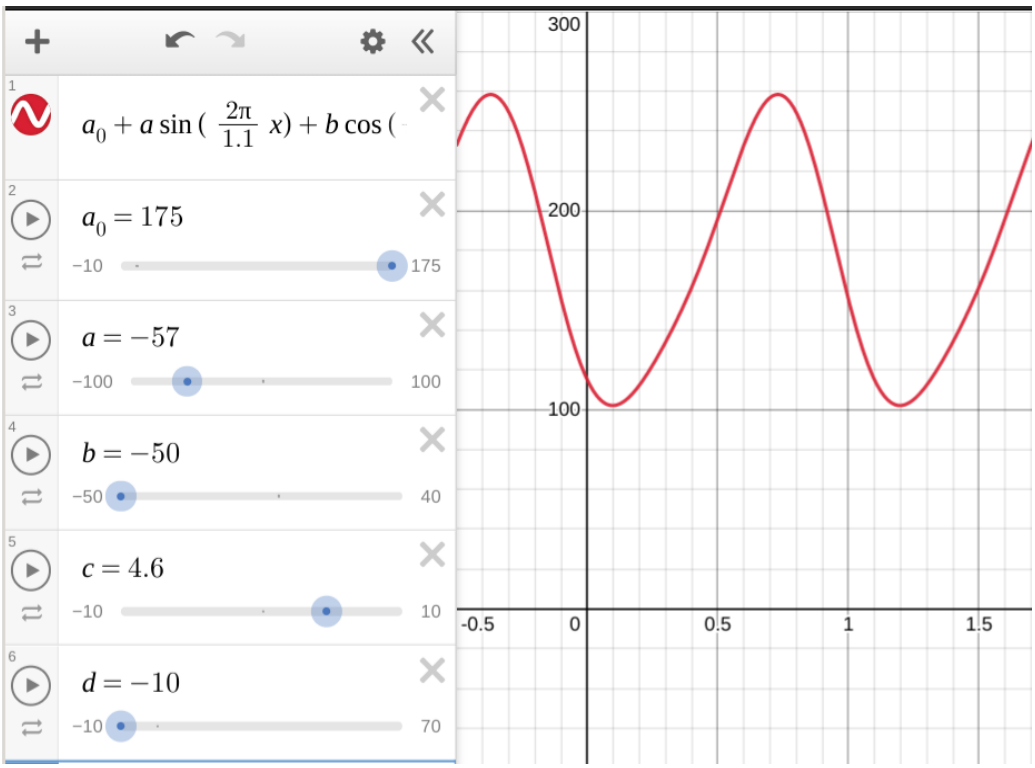


Рис.5в. Периодическая зависимость момента u_1 от времени в течение нескольких шагов с периодом 1.1 с.

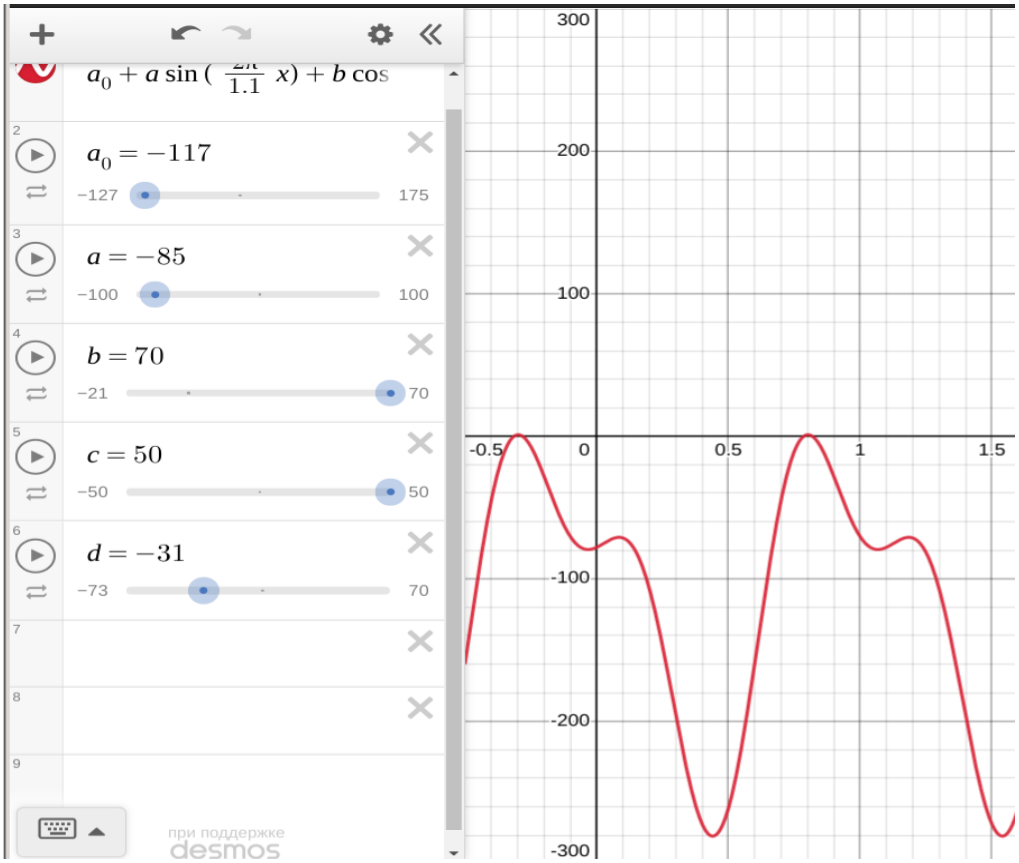


Рис.5г. Периодическая зависимость момента u_2 от времени в течение нескольких шагов с периодом 1.1 с.

Так как мы исследуем управление моментом в колене, а именно в опорной ноге, то нас больше интересует момент u_1 (Рис.5в)., и ниже отдельно представим зависимость u_1 и угла сгиба колена $\Omega_1 = 180 - \alpha_1 + \beta_1$ от времени.

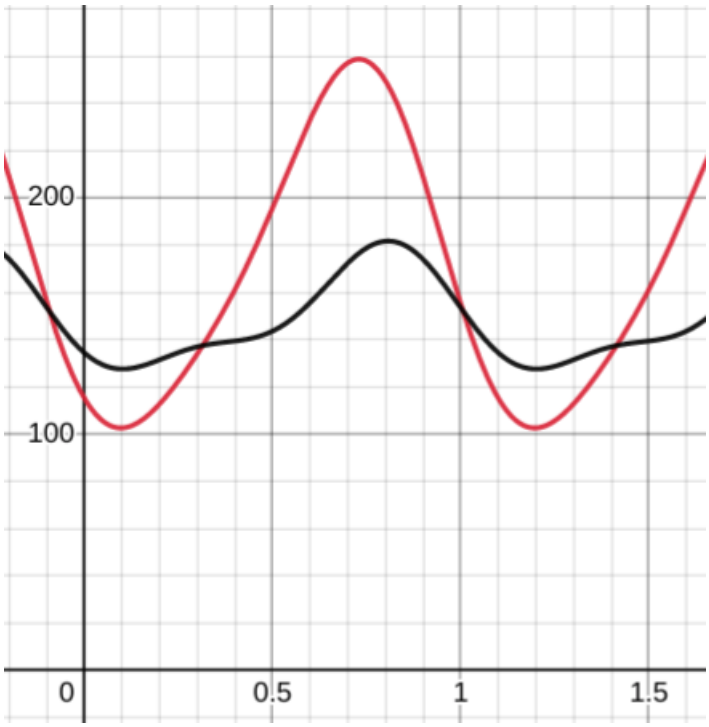


Рис.6. Периодическая зависимость момента u_1 (красный, Н*м) и угла сгиба колена Ω_1 (черный, °) от времени в течение нескольких шагов с периодом 1.1 с.

Как видим, некоторое представление управляющего момента может быть аппроксимировано уже с помощью одного только угла сгиба колена: минимумы и максимумы у графиков совпадают. В нашей модели экзоскелета управляющий момент включает в себя силу на нижнем датчике (подробнее об управляющем алгоритме в пункте 3.2).

Полученные зависимости моментов от времени могут быть использованы для исследования энергозатрат.

2.3. Устройство экспериментального образца

Первая модель экзоскелета имела 2 силовых датчика: один под предполагаемым местом пятки, второй — под ступней.

Рис.7. Первая механическая модель экзоскелета

Предполагалось, что датчики будут показывать разные значения (первый — вертикальную составляющую, второй — силу, обусловленную дополнительным весом), но проведенное исследование и анализ графиков показали противоположное. Ниже представлена динамическая схема для такого расположения датчиков.

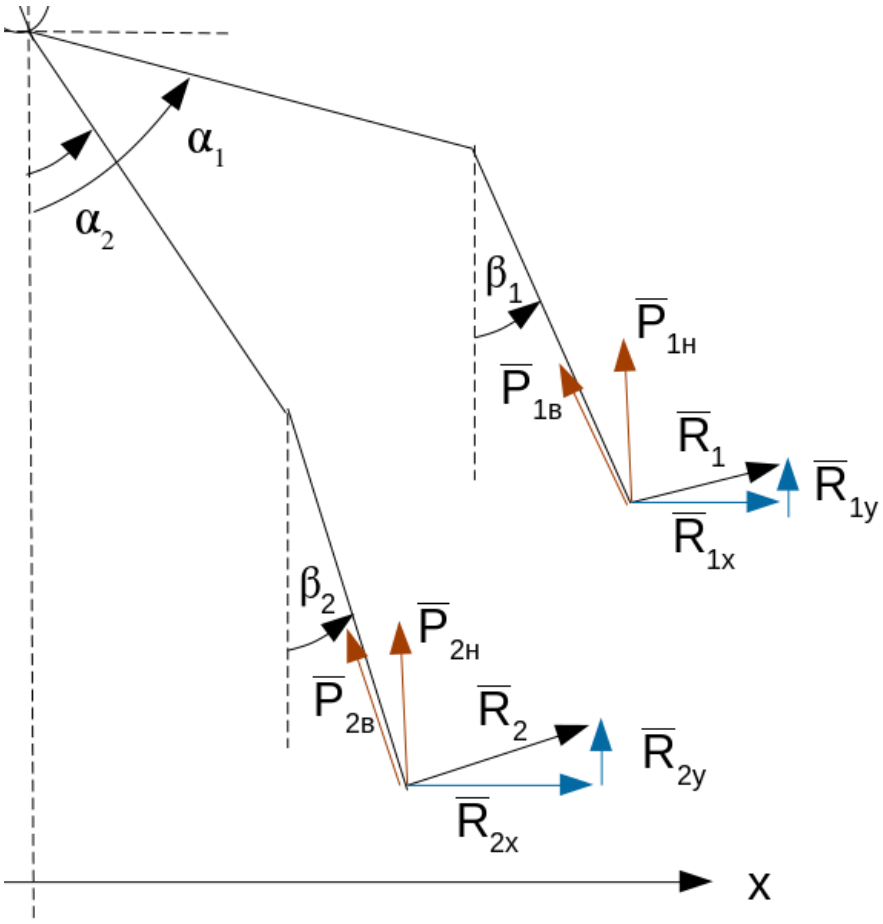
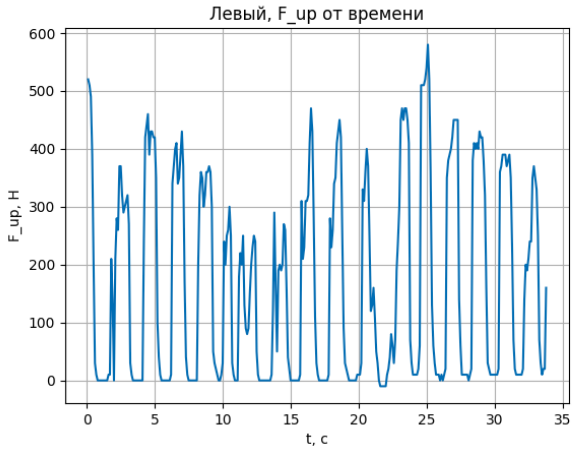


Рис.8. Динамическая схема первой модели с указанием сил на датчиках

Был поставлен эксперимент: движение в экзоскелете (ходьба + приседания + уступ), и построены графики зависимости силы в датчиках от времени для каждого типа движения.



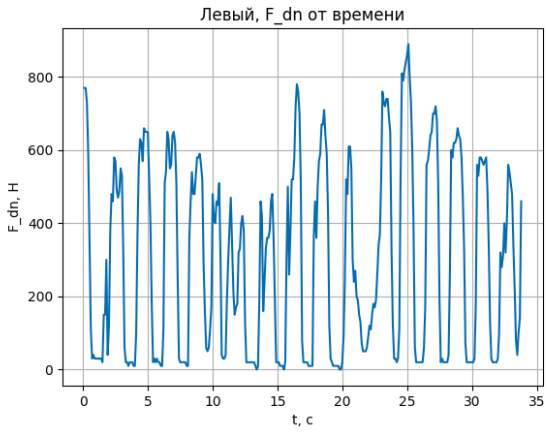


Рис.9а, 9б. Графики зависимости сил в датчиках левой ноги

Нарисуем график зависимости верхней силы от нижней (рис.10).

Заметим здесь ярко выраженную линейную зависимость с коэффициентом 1 с небольшим отклонением.

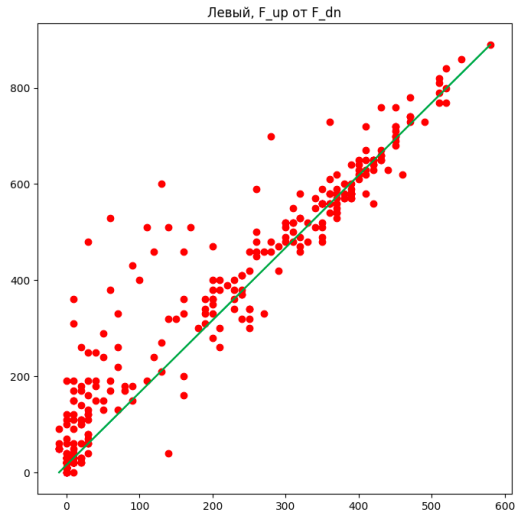


Рис.10. График зависимости сил в верхнем датчике от нижней в левой ноге

Метод наименьших квадратов показывает, что, действительно, коэффициент наклона кривой $k=0.999\approx1.0$, сдвиг $b=4.29\approx0$. (т. к. наши данные имеют порядок 800). Код представлен в приложении 7.1.

Эта модель может быть улучшена для дальнейшего исследования, и ее схема может быть упрощена — в ней достаточно всего одного силового датчика.

2.4. Переход ко второй модели

Поместим один датчик в голень экзоскелета. Он будет измерять силу, обусловленную весом тела и переносимых грузов, а также силой трения-сцепления.

Рис.11. Вторая механическая модель экзоскелета

Чтобы понять, какую силу он измеряет в терминах нашей математической модели, рассмотрим динамическую схему на рис.12. Заметим, что датчик иногда называют датчиком давления, но в действительности он измеряет силу.

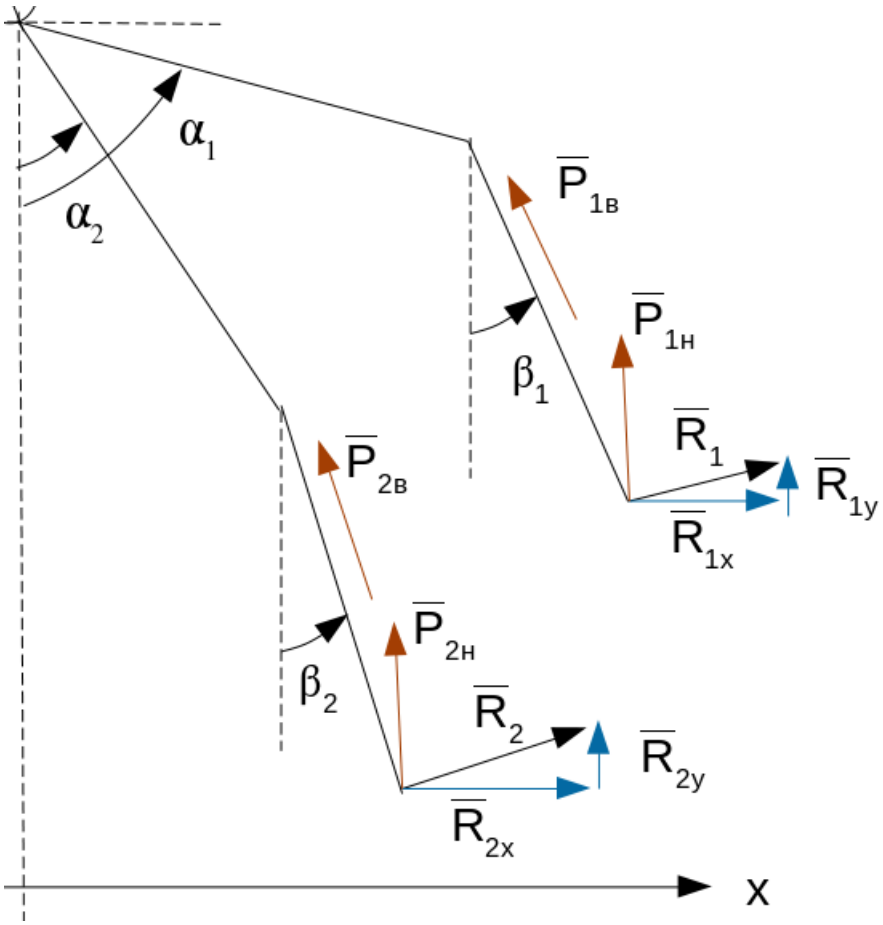


Рис.12. Динамическая схема с указанием сил на датчиках

Здесь видно из проекций на оси x, y, что:

$$P_{i\text{в}}cos(\beta_i) + P_{i\text{н}} = R_{iy}$$

$$P_{i\text{в}}sin(\beta_i) = R_{ix}$$

Также выразим наш измеряемый угол Ω_i из α_i, β_i:

$$\Omega_i = 180 - \alpha_i + \beta_i$$

Создаваемые двигателем моменты u_i действуют в коленных суставах. В модели для исследования энергетических оценок и получения рисунков ходьбы ограничимся измерением углов Ω_i, сил (проекций R_{iy}) и добавочным моментом только в коленных суставах (u_i).

Для аналогичных экспериментов (ходьба + приседания + уступ) построены графики зависимости сил в датчиках от времени и сил друг от друга (нижней от верхней). Чистота эксперимента гарантируется тем, что он был проведен при условии, что в одной ноге экзоскелета была старая модель, а в другой — уже новая, то есть, испытуемый был один и тот же. Код представлен в приложении 7.1.

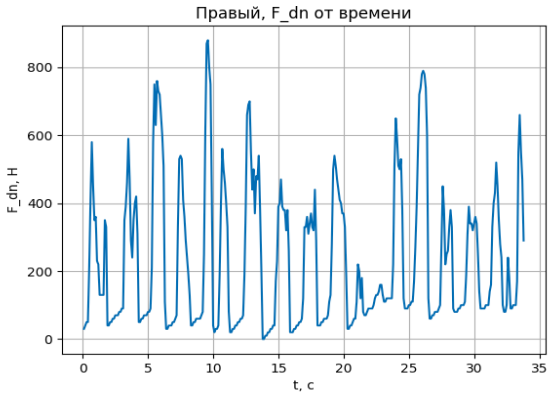
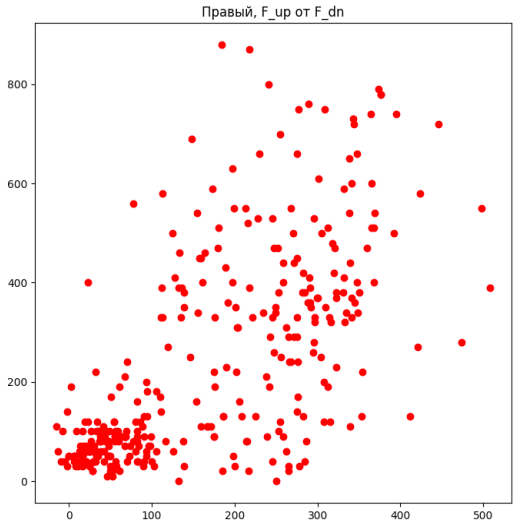


Рис.13а, 13б. Графики зависимости сил в датчиках правой ноги

На графике зависимости верхней силы от нижней, точки уже не ложатся на



одну прямую, как и было предсказано в математической модели.

Рис.14. График зависимости силы в верхнем датчике от нижней в правой ноге

2.5. Калибровка датчиков

Из различных исследований ходьбы, например, [2], можно составить общий порядок действий человека в течение шага. При осуществлении шага в одной ноге:

1) пятка отрывается от земли (сила в датчике под стопой F_{dn} равняется нулю)
2) сгибается нога (уменьшается угол сгиба колена Ω)
3) в воздухе нога разгибается (увеличивается угол сгиба колена Ω)
4) нога ставится на землю почти выпрямленной (при почти развернутом угле Ω, сила в датчике под стопой F_{dn} увеличивается)
5) в фазе переноса на другую ногу опора на эту ногу уменьшается (сила в датчике под стопой F_{dn} уменьшается к нулю до отрыва ноги от земли)

Проверим, корректно ли работают датчики в экзоскелете, используя эти соображения. Проведем эксперимент плоской регулярной ходьбы в экзоскелете, обработаем данные ходьбы и вырежем одну подборку данных для одной ноги между двумя нулями силы (эта подборка означает один шаг), после чего изобразим изменения двух величин на графике в зависимости от времени.

Период одного шагового цикла оказался равен 2.0 секунды. Все предположения 1)-5) об изменении показаний датчиков видим на графике зависимости показаний положения и силы от времени. То есть, наши датчики работают корректно. Код рисования этого и других графиков представлен в приложении 7.2.

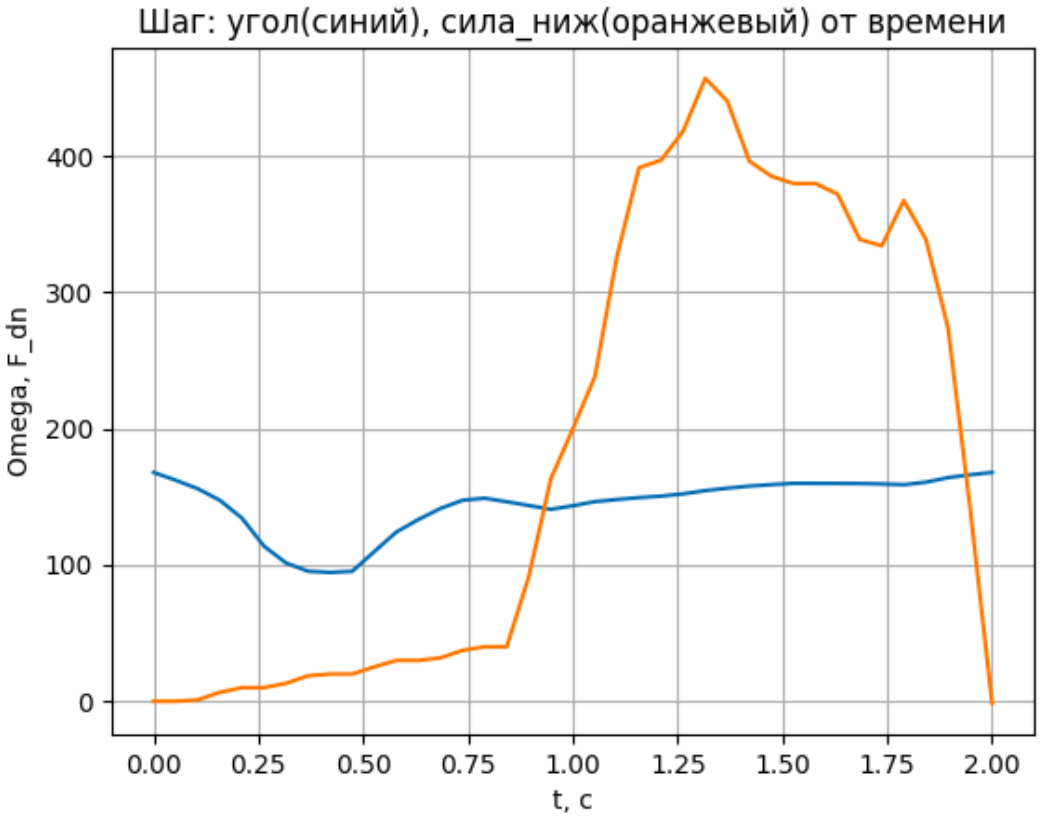


Рис.15. График зависимости угла(синий, °) и силы на нижнем датчике(оранжевый, Н) от времени(с) в правой ноге

Калибровочные данные

Так как датчики изначально показывали единицы изменения не в СИ, то проведем их калибровку. Возьмем показания каждого датчика при известной нагрузке и найдем 2 коэффициента (пропорциональности, С1 и сдвига, С2).

В итоге из показаний датчика можно получить данные в нужных единицах по формуле:

$X_{реал} = C_1 X_{дат} + C_2$

Датчик	Показания при образцовой нагрузке	Коэффициент пропорциональности	Коэффициент сдвига
Угол, левая нога	90° = -1544 ед. 180° = 35 ед.	1/17.5°	182°
Угол, правая нога	90° = -1578 ед. 180° = -94 ед.	1/16.5°	185°
Сила, левая нога	0 Н = 989 ед. 700 Н = 1059 ед.	1/10 Н	989 Н
Сила, правая нога	0 Н = 1010 ед. 700 Н = 1080 ед.	1/10 Н	1010 Н
Время	1 с = 10 ед.	1/10 с	0 с

3. Схема экзоскелета

3.1. Схема расположения датчиков и подключения плат

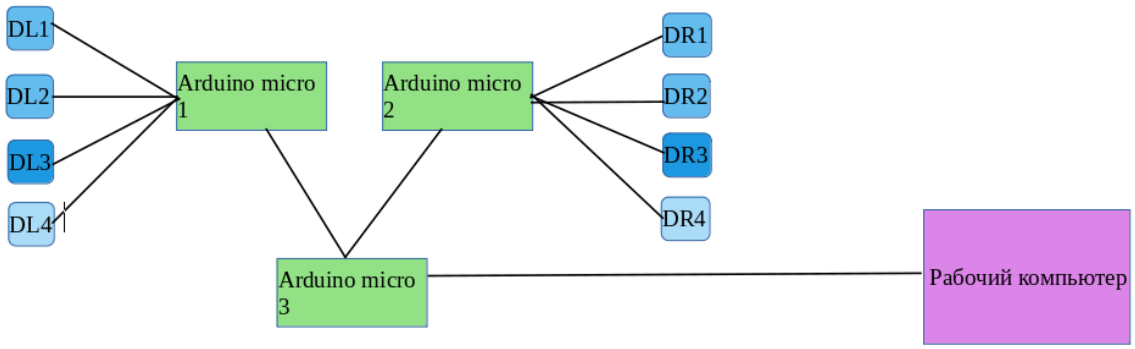


Рис.16. Общая схема датчиков и контроллеров

Плата Arduino micro 1 расположена в левой ноге и имеет 4 датчика: сила1, сила2, позиция, создаваемый момент.

Плата Arduino micro 2 расположена в правой ноге и симметрична первой.

Плата Arduino micro 3 расположена в корпусе скелета сзади и отвечает за соединение первых двух, на ней имеется Bluetooth адаптер для передачи снимаемых данных на рабочий компьютер, также с этого контроллера осуществляется управление обоими приводами.

Система подключается к компьтюьеру только для съема данных и сборки программы, в остальном она может функционировать автономно. Код для снятия и вывода данных на Delhi представлен в приложении 7.5.

3.2. Основной алгоритм управления

Как отмечено в предыдущей главе, каждая нога управляется алгоритмом с Arduino micro. Рассмотрим подробнее этот алгоритм.

Каждые 100 миллисекунд с контроллеров считываются данные, и на двигатель подается нужный дополнительный момент. Момент определяется из силы в верхнем датчике ($F_в$), реальной текущей позиции (Ω_{cur}), и программной заданной позиции (Ω_{prg}). Программная (добавочная) позиция используется для проверки работы двигателя и настройки работы. Если сила на верхнем датчике и позиция лежат в измеримом диапазоне, то задаем временную величину x. В простом модельном случае x равняется силе, умноженной на угол, со знаком «-»:

$$x = -F_в * \Omega_{cur}$$

Но с учетом экспериментальных данных, надо увеличить эту величину на постоянный добавочный момент, который подобран эмпирически. Получаем:

$$x = -F_в * \Omega_{cur} + k * \Omega_{prg}$$

При этом сила корректируется «нулем силы» — когда на датчике никто не стоит, то он показывает 1010 единиц. Позиция тоже корректируется, если в разных ногах стоят по-разному откалиброванные датчики. После такой операции, если x превышает максимально возможный момент M_{max}, то оставим x максимальным; если он ниже минимального — оставим минимальным. Этой величиной x зададим момент M_{add}.

Для каждой ноги данные калибровки и нулей, а, следовательно, диапазона значений, будут разными. Код для определения M_{add} и подачи его на двигатель представлен в приложении 7.6.

4. Графики (рисунки ходьбы) и их анализ

4.1. Графики силы, положения

Используя pyplot, изобразим графики снимаемых нами данных. Данные изначально записаны в виде текстового файла определенной структуры, из которого сделаны списки из параметров для каждого момента времени и иллюстрированы зависимости от времени. Код представлен в приложении 7.2.

Ниже представлены 4 графика снимаемых параметров для ходьбы.

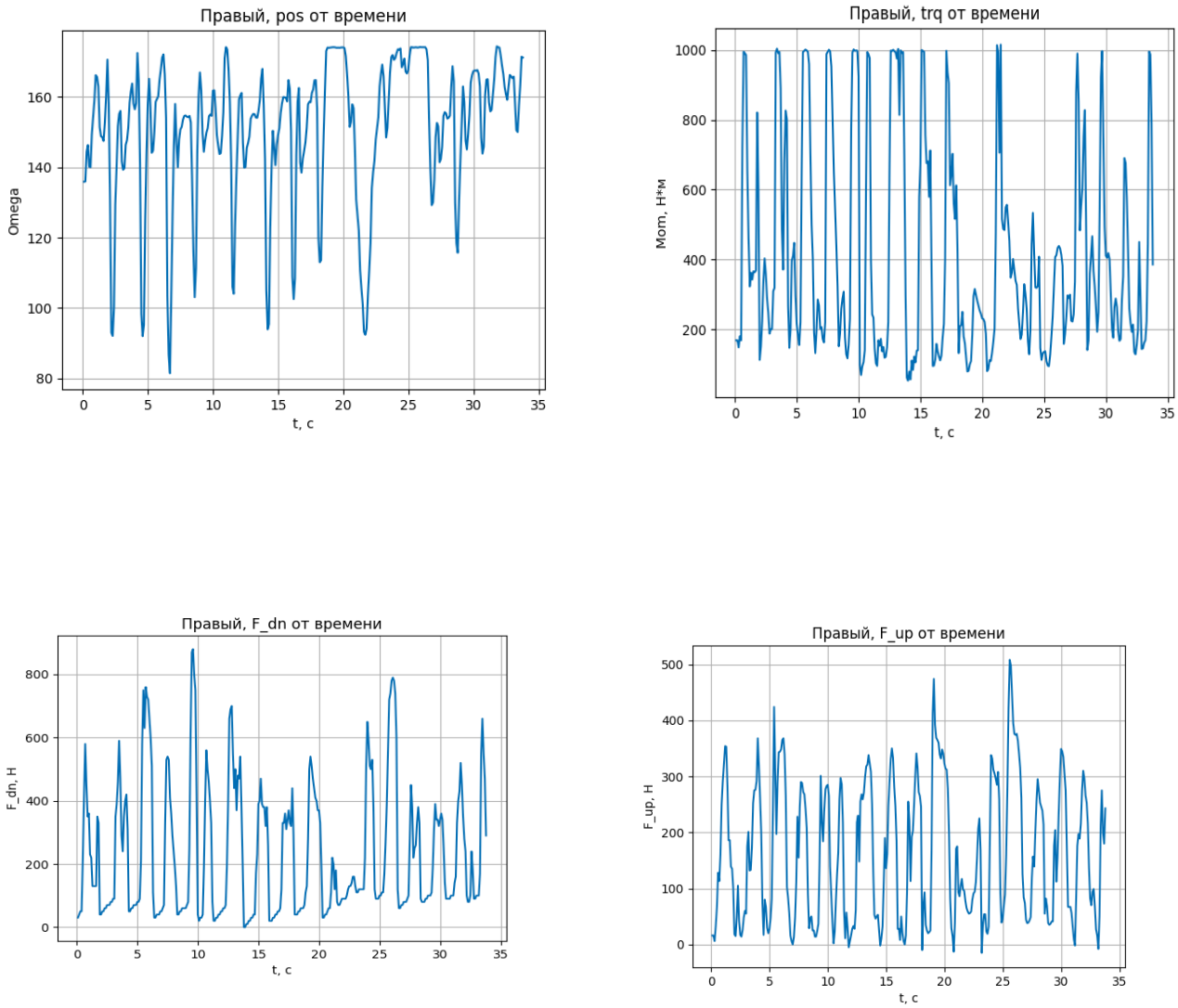


Рис.17. Графики зависимости снимаемых параметров от времени для ходьбы

Видны отчетливые пики со срывами и ямы с резким поднятием. Также ходьба имеет некоторую периодичность, и ее можно заметить в попарном изменении

силы на одной и другой ноге прямо в текстовых данных: пока на опорной ноге сила повышается примерно на 60 * 10 Н (вес испытуемого), на переносимой падает. Заметим также, что в реальной ходьбе в момент удара ноги о пол скачок силы может быть значительно больше веса испытуемого, но время этого скачка сопоставимо с 1 мс, а наши датчики считывают показания с периодичностью в 100 мс, поэтому, чаще всего, на графиках удар незаметен.

Визуально по рисункам можно отличить ходьбу от приседания и поднятия на уступ. Программная реализация поиска и определения такого отличия поможет классифицировать разные виды движения человека и строить более эффективные системы управления двигателем.

Будем строить классификацию на основе метода логистической регрессии. Логистическая регрессия — это широко известный метод в статистике, который используется для прогнозирования вероятности результата и является популярным для задач классификации. Алгоритм прогнозирует вероятность возникновения события путем приближения данных к логистической функции.

Для такой реализации нам потребуется несколько шагов:

1. Определить, что будет являться одной итерацией шага, поднятия на уступ, приседания.
2. Создать тренировочный датасет из шагов, поднятий, приседаний.
3. Тренировать модель.
4. Проверить точность классификации на некоторых проверочных данных.

4.2. Анализ графиков для плоской ходьбы

Определим, что в нашей программе один шаг — все снимаемые данные между двумя нулями сил. Программно это реализуется так: находим в массиве сил (в нашем опыте — нижних сил для правой ноги) все индексы, отвечающие скачкам более 20 единиц по силе вниз. Это означает начало шага одной ногой. Между двумя такими индексами расстояние должно быть порядка 19 — 25 ед. (что равняется примерно 2 секундам на цикл шага). Таких наборов данных в файле больше, чем с периодом более 20 единиц, поэтому рассматривать для датасета будем именно такие.

После чего берем все данные между этими индексами для каждого момента времени (допустим, это 4 позиции — два угла и две нижних силы), и все данные записываем в массив step — двумерный массив 20*4. Каждый полученный step прикрепляем к общему списку шагов. Вот так выглядит средний шаг на общем графике:

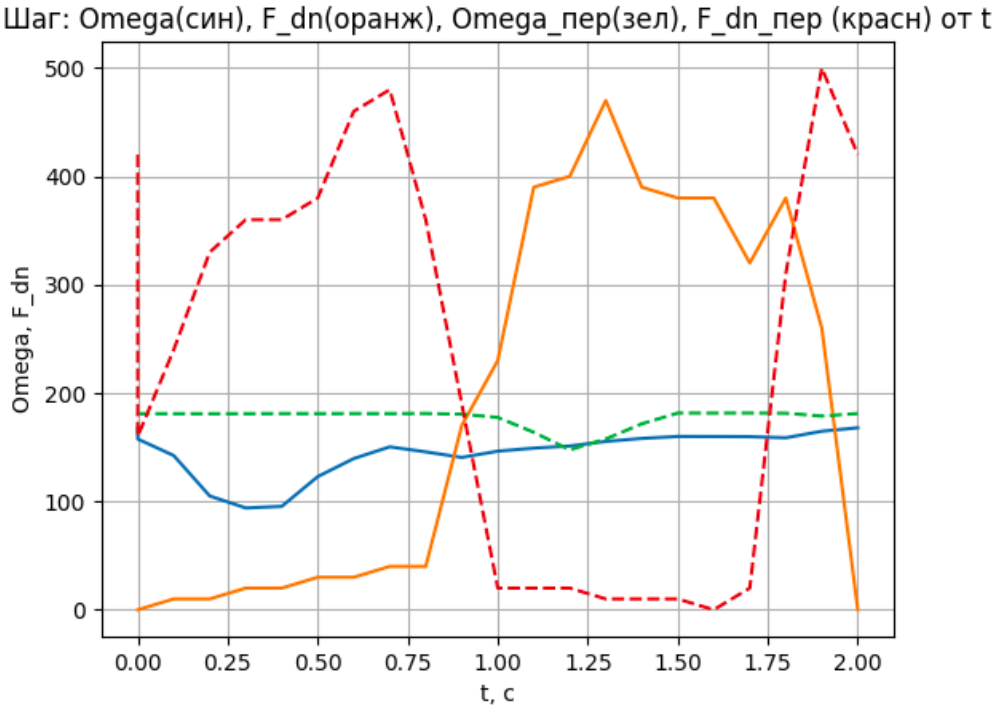


Рис.18. Изменение углов опорной(синий) и переносной(зеленый) ноги, и изменение сил на опорной(оранжевый) и переносной(красный) ноге на нижнем датчике в течение одного шага.

Все шаги записываем в csv-таблицу с пометкой «1» — это есть часть будущего датасета со всеми видами движения.

Похожие списки (например, приседаний) мы будем создавать и классифицировать позже.

Отметим, почему надо анализировать движение по одному шагу. Группу шагов рассматривать сложно и зачастую не имеет смысла, т.к. при ходьбе по неровной местности все шаги разные. Можно усреднить параметры шага, но усреднение будет неточным (трудно реализуемым) из-за разных периодов шагов и моментов, в который нога была согнута, разогнута и т.д. После разбора ходьбы на отдельные шаги сможем определить характерные энергетические характеристики шага, чтобы по данным понять, что это шаг. Аналогично поступим с поднятием на уступ и приседанием.

4.3. Анализ графиков для приседаний

Определим, что такое одно приседание. В отличие от шагов, приседание проще определять по координатам углов — скачков силы нет. В нашей программе одно приседание — все снимаемые данные между одним положением развернутого угла одной ноги и другим положением развернутого угла этой же ноги. Программно это реализуется так: находим в массиве углов (в нашем опыте — для правой ноги) все индексы, отвечающие минимумам (от 130 до 150 единиц по углу). Это означает, согласно калибровочным данным, начало сгибания ноги (приседания). Между двумя такими индексами расстояние должно быть от 20 до 30 (что равняется 2-3 секундам на цикл приседания).

После чего берем все данные между этими индексами для каждого момента времени (допустим, это 4 позиции — два угла и две нижних силы), и все данные записываем в массив squat — двумерный массив 20*4 (до 30*4). Каждый полученный squat прикрепляем к общему списку приседаний.

Все приседания записываем в csv-таблицу с пометкой «2» — это есть будущая часть датасета со всеми видами движения.

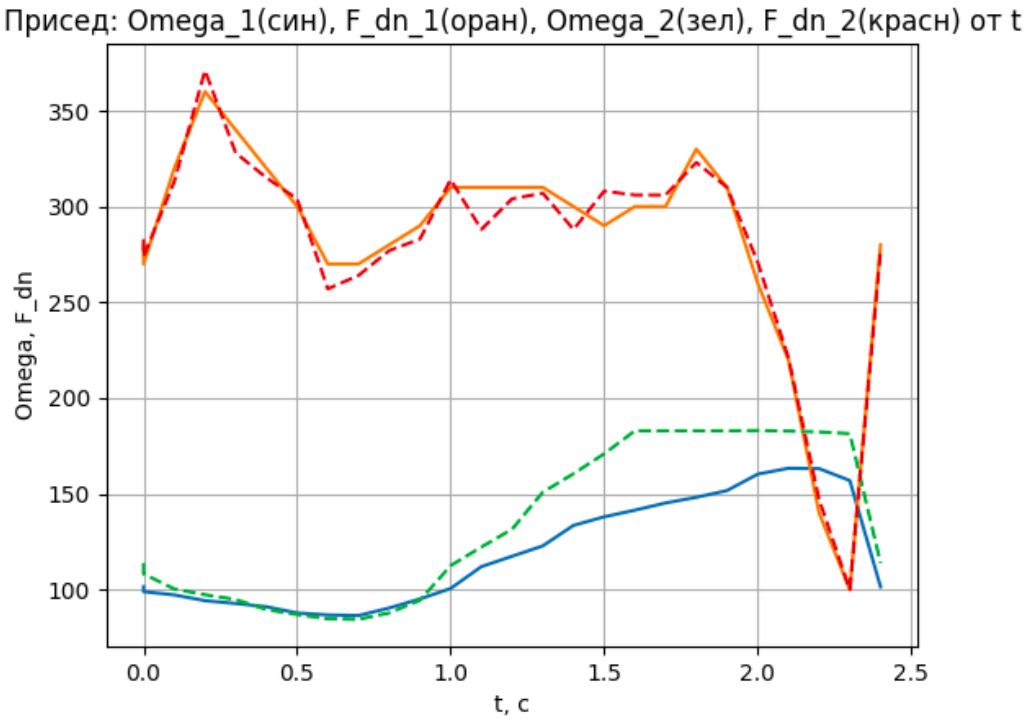


Рис.19. Изменение углов правой(синий) и левой(зеленый) ноги, и изменение сил на правой(оранжевый) и левой(красный) ноге на нижнем датчике в течение одного приседания.

4.4. Анализ графиков для поднятия на уступ

Определим, что такое одно поднятие. Как и в случае с шагом, поднятие можно определять по скачкам силы. В нашей программе одно поднятие — все снимаемые данные между двумя нулями сил. Программно это реализуется так: находим в массиве сил (в нашем опыте — нижних сил для правой ноги) все индексы, отвечающие скачкам более 20 единиц по силе вниз. Это означает начало поднятия на одну ногу. Между двумя такими индексами расстояние должно быть порядка 25 — 30 ед. (что равняется примерно 3 секундам на цикл поднятия).

После чего берем все данные между этими индексами для каждого момента времени (допустим, это 4 позиции — два угла и две нижних силы), и все данные записываем в массив stair — двумерный массив 25*4. Каждый полученный stair прикрепляем к общему списку зашагиваний на лестницу.

Все поднятия записываем в csv-таблицу с пометкой «3» — это есть часть датасета со всеми видами движения.

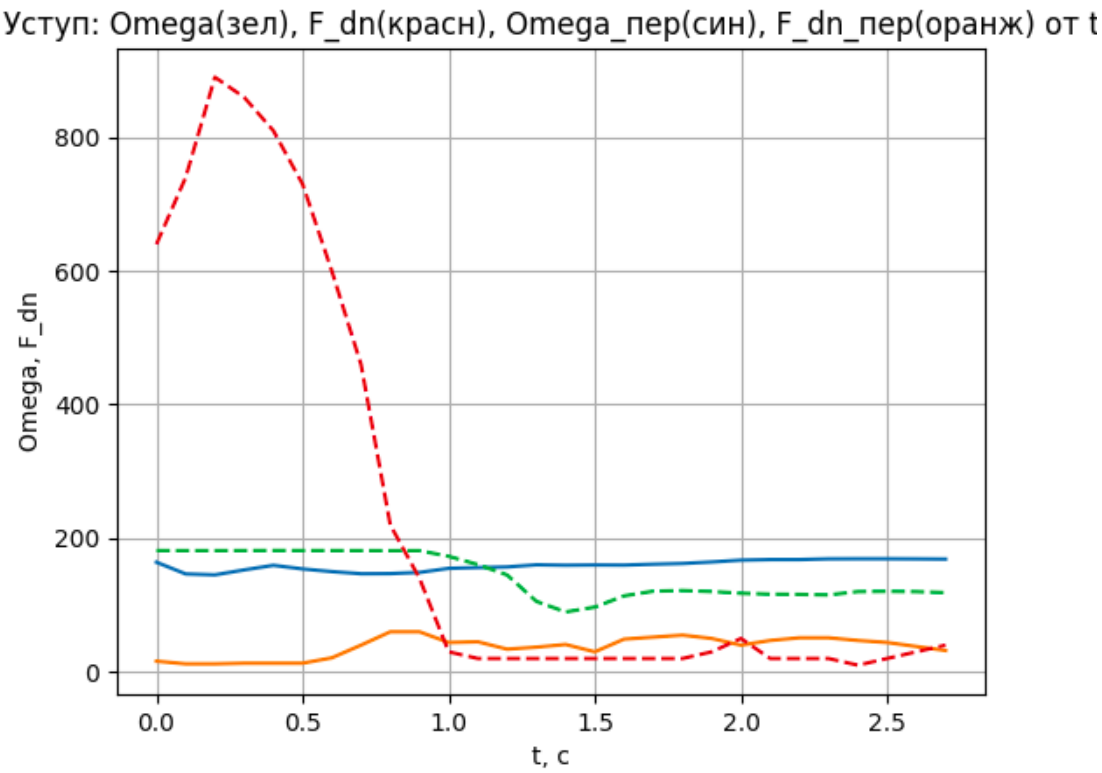


Рис.20. Изменение углов опорной(зеленый) и переносной(синий) ноги, и изменение сил на опорной(красный) и переносной(оранжевый) ноге на нижнем датчике в течение одного поднятия.

Стоит отметить, что поднятие на уступ в нашем эксперименте — не цикличное действие. Осуществлялось зашагивание сначала одной ногой, затем спуск и повтор действия для другой ноги. На графике видна такая зависимость — сначала большой пик силы в опорном датчике, затем наблюдается спуск.

4.5. Создание общего датасета

Проблема создания общего датасета состояла в том, что для каждого вида движения и для каждого конкретного шага длины массивов различались (физически это означает разный период движения). Но для общего анализа данных они должны иметь одну структуру, следовательно, надо придумать метод приведения массивов размера N*4 к массиву 20*4, т.е. к единому периоду, равному 2 с.

Метод кусочно-линейной непрерывной функции.

Все данные в массиве интерпретируются в виде графиков кусочно-линейной непрерывной функции. Затем при уплотнении(разрежении) массива на заданное место t подставляется значение функции в этой точке.

Метод интерполяции.

Данный метод является обобщением предыдущего для многочленов. Теоретическое описание метода изложено в [11]. Здесь мы подбираем многочлен, подходящий под заданные значения x,y. Пример интерполяции функции по точкам с помощью многочленов разных степеней показан ниже. Для примера мы взяли 8 точек из массива углов и интерполировали эту часть по точкам.

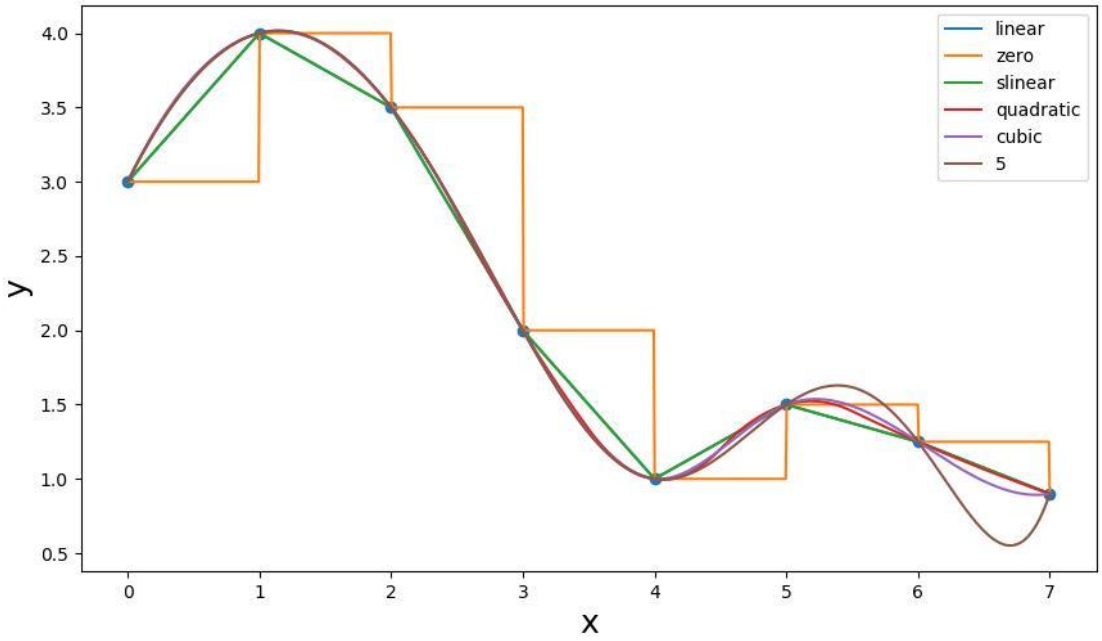


Рис.21. Интерполяция функции на отрезке

Применив такой метод, можно найти функцию, приближающую нашу зависимость на данном отрезке, и брать любые значения из отрезка для будущего набора данных. Для такой реализации был использован метод библиотеки scipy с названием interpolate.interp1d() — одномерная интерполяция.

Также, для того, чтобы все данные были численными, надо преобразовать массивы размера (20*4) в массивы (80*1) — тогда мы сможем классифицировать наши данные с помощью разных методов библиотеки sklearn. Добьемся этого с помощью метода numpy.reshape().

Таким образом из массивов разных размеров мы пришли к массивам одного размера — (80*1).

Объединив все три датасета в один и проставив метки соответствующим классам (1 — шаги, 2 — приседания, 3 — поднятия), мы получили датасет, готовый к исследованию (на нем можно обучать модели).

Код для его создания представлен в приложении 7.3.

4.6. Обучение логистической модели

Scikit-Learn — это Python-библиотека, впервые разработанная David Cournapeau в 2007 году. В этой библиотеке находится большое количество алгоритмов для задач, связанных с классификацией и машинным обучением в целом.

В нашей работе для классификации будут использованы метод опорных векторов, метод k-ближайших соседей и метод логистической регрессии [12] и сравнены их эффективности.

Для начала для чтения данных использована функция read_csv() из библиотеки pandas. После чего данные поделены на признаки (X) и метки (Y): метки — это значения 1,2 и 3, отвечающие принадлежности данных к тому или иному классу, а признаки — это все остальное (массив размера 80*1).

После чего выборка разделена на тренировочную: X_train, y_train (75%) и тестировочную: X_test, y_test (25%) с помощью train_test_split() из библиотеки sklearn. Проценты этих данных в процессе тестирования изменялись — например, тестовая выборка может иметь размер от 15% до 30% от всего датасета (соображения оптимальности выборки взяты из [13]).

Созданы три модели: для опорных векторов (SVC), для k-ближайших соседей и для логистической регрессии. Все модели взяты из sklearn. После чего они были обучены с помощью метода fit() из библиотеки sklearn, и были получены соответствующие их предсказаниям векторы SVC_prediction, KNN_prediction , LogReg_prediction.

По этим векторам можно определить точность наших методов. Точность определялась рядом методов: метрика accuracy — процент точности классификации, confusion matrix — матрица неточностей, которая показывает, сколько объектов попало под ту или иную классификацию, и classification report — полный отчет о классификации [14].

Для тестовой выборки размером 25% получили следующий результат:

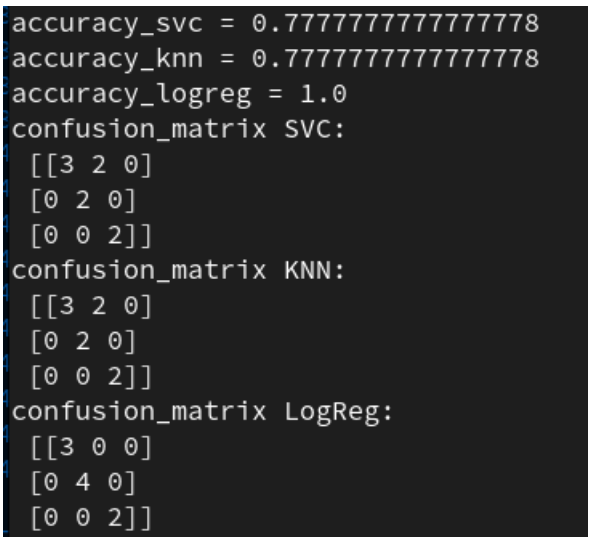


Рис.22. Оценка моделей для тестовой выборки размером 25% от общего датасета

Здесь видим, что первые два метода имеют точность порядка 78%, а логистическая регрессия справилась с классификацией идеально. Также наблюдаем в матрицах ошибок предсказуемый результат о том, что иногда поднятие классифицируется как шаг — они действительно похожи.

Перед созданием полного отчета дадим несколько определений.

Accuracy — это показатель, который описывает общую точность предсказания модели по всем классам. Он рассчитывается как отношение количества правильных прогнозов к их общему количеству.

Precision представляет собой отношение числа экземпляров, верно классифицированных как «1», к общему числу выборок с меткой «1» (распознанных правильно и неправильно). Precision измеряет точность модели при определении класса «1». Аналогично Precision считается и для классов «2», «3».

Recall рассчитывается как отношение числа «1»-выборок, корректно классифицированных как «1», к общему количеству экземпляров «1». Recall измеряет способность модели обнаруживать выборки, относящиеся к классу «1». Чем выше recall, тем больше «1» семплов было найдено.

F1-мера — это среднее гармоническое значение между precision и recall.

Support — количество фактических вхождений класса в наборе данных.

Макроусредненная оценка F1 (macro avg F1) вычисляется путем взятия среднего арифметического (также известного как невзвешенное среднее) всех оценок F1 для каждого класса.

Средневзвешенный балл F1 (weighted avg F) рассчитывается путем получения среднего значения всех баллов F1 для каждого класса с учетом каждого класса.

```
classification_report SVC:
              precision    recall  f1-score   support

           1       1.00      0.60      0.75         5
           2       0.50      1.00      0.67         2
           3       1.00      1.00      1.00         2

    accuracy                           0.78         9
   macro avg       0.83      0.87      0.81         9
weighted avg       0.89      0.78      0.79         9

classification_report KNN:
              precision    recall  f1-score   support

           1       1.00      0.60      0.75         5
           2       0.50      1.00      0.67         2
           3       1.00      1.00      1.00         2

    accuracy                           0.78         9
   macro avg       0.83      0.87      0.81         9
weighted avg       0.89      0.78      0.79         9

classification_report LogReg:
              precision    recall  f1-score   support

           1       1.00      1.00      1.00         3
           2       1.00      1.00      1.00         4
           3       1.00      1.00      1.00         2

    accuracy                           1.00         9
   macro avg       1.00      1.00      1.00         9
weighted avg       1.00      1.00      1.00         9
```

Рис.23. Отчет о классификациях разными методами на тестовой выборке размером 25% от общего датасета

Здесь видим полные отчеты о классификациях, включающие accuracy, precicison, recall, f1-score и средние их значения. Попробуем найти такую выборку, чтобы точности первых двух методов не совпадали.

```
classification_report SVC:
              precision    recall  f1-score   support

           1       1.00      0.50      0.67         4
           2       0.67      1.00      0.80         2
           3       0.50      1.00      0.67         1

    accuracy                           0.71         7
   macro avg       0.72      0.83      0.71         7
weighted avg       0.83      0.71      0.70         7

classification_report KNN:
              precision    recall  f1-score   support

           1       0.50      1.00      0.67         1
           2       1.00      0.75      0.86         4
           3       1.00      1.00      1.00         2

    accuracy                           0.86         7
   macro avg       0.83      0.92      0.84         7
weighted avg       0.93      0.86      0.87         7

classification_report LogReg:
              precision    recall  f1-score   support

           1       1.00      1.00      1.00         2
           2       1.00      1.00      1.00         3
           3       1.00      1.00      1.00         2

    accuracy                           1.00         7
   macro avg       1.00      1.00      1.00         7
weighted avg       1.00      1.00      1.00         7
```

Рис.24. Отчет о классификациях разными методами на тестовой выборке размером 18% от общего датасета

Здесь accuracy для метода опорных векторов равна 0.71, для метода KNN — 0.86, а для логистической регрессии по-прежнему 1.0. Были проведены тесты и на других выборках, где логистическая регрессия показала свою эффективность в нашей задаче многоклассовой классификации. Код для создания и обучения моделей классификации представлен в приложении 7.4.

Таким образом, нашу модель преобразования данных и их классификации можно использовать в дальнейшем исследовании движения человека.

5. Заключение

В результате работы был рассмотрен алгоритм управления приводами в коленном суставе экзоскелета нижних конечностей. Изучена математическая модель пятизвенной ходьбы, сопоставлена с механической моделью и может быть в дальнейшем использована для изучения других видов движения. Найдены теоретические зависимости углов и моментов в суставах в зависимости от времени в течение одного шага. Рассмотрено устройство экспериментального образца и различные положения силовых датчиков в экзоскелете.

Была проведена серия экспериментов: рассмотрено движение экзоскелета как в режиме плоской регулярной ходьбы по ровной горизонтальной поверхности, так и при ходьбе с препятствиями и приседаниях. При эксперименте фиксировались данные с силовых датчиков в стопе и в нижнем шарнире, угол сгиба колена, а также момент, создаваемый двигателем в колене. По этим данным были созданы рисунки ходьбы и приседаний. Также была создана логистическая модель, распознающая наборы данных и различающая один шаг от одного приседания и поднятия на уступ. При обработке данных использованы библиотеки с реализацией элементов искусственного интеллекта (scikit learn) для среды программирования python, реализующие современные методы исследования данных.

Полученная логистическая модель может быть использована для улучшения управляющего алгоритма, а математическая модель может быть использована при изучении движений человека и впоследствии уточнена.

На полученных результатах могут быть проведены сравнительные оценки моментов, полученных теоретически, с моментами, созданными дополнительно. Также впоследствии может быть произведена оценка эффективности экзоскелета, основанная на сравнении моментов.

6. Список литературы

1. Белецкий В. В., Кирсанова Т. С. Плоские линейные модели двуногой ходьбы //Известия АН СССР. Механика твёрдого тела. – 1976. – No. 4. – С. 51.

2. Белецкий В. В., «Двуногая ходьба: модельные задачи динамики и управления». — М.: «Наука». Главная редакция физико-математической литературы, 1984. —288 с.

3. Алексеева Л. А., Голубев Ю. Ф. Модель динамики шагающего аппарата//Изв. АН СССР, Техническая кибернетика. – 1975. – No 3. – С. 72—80.

4. Витензон А. С. Динамические фазы цикла ходьбы.— В кн.: Биомеханика. Рига, 1975, с. 251—257.

5. Viet Anh Dung Cai. Control of a Self-adjusting Lower Limb Exoskeleton for Knee Assistance. ROMANSY 21 - Robot Design, Dynamics and Control (pp.385-392)

6. Формальский А. М., «Перемещение антропоморфных механизмов» — М.: «Наука», Главная редакция физико-математической литературы, 1982. —368 с.

7. Белецкий В.В., Бербюк В. Е. Нелинейная модель двуногого шагающего аппарата, снабженного управляемыми стопами.— М.: Институт прикладной математики АН СССР, 1978, Препринт, С. 54— 67.

8. Лавровский Э. К., Письменная Е. В., Комаров П. А. О задаче организации ходьбы экзоскелетона нижних конечностей при помощи управления в коленных шарнирах //Российский журнал биомеханики. 2015. Т. 19, С. 158–176.

9. Антипов В. М., Карлов А. Е., Фѐдоров А. В., Аль М. Х. Х. Распределение энергозатрат в системе человек-экзоскелет // Вопросы методологии естествознания и технических наук: современный контекст. - 2019.- С. 109-112.

10. Гурфинкель В. С., Осовец С. М. Динамика равновесия вертикальной позы человека//Биофизика. – 1972. – т. 17. вып. 3. – С. 478—485.

11. Ибрагимов И. И. Методы интерполяции функций и некоторые их применения. 1971. С. 100 — 113.

12. Вьюгин В. В. Математические основы машинного обучения и прогнозирования //Электронное издание, М.: МЦНМО, 2014. С. 17 — 35.

13. Елистратова Е. А., Иванов Г. В. Онлайн-учебник по машинному обучению от ШАД //Школа анализа данных Яндекс, 2020

14. Бринк Х., Ричардс Дж., Феверолф М. Машинное обучение. -СПб .: Питер, 2017. (Серия «Библиотека программиста»). С. 144 — 152.

7. Приложение

7.1. Реализация метода наименьших квадратов на данных из файла '1_1.txt' для нахождения коэффициентов k_approx, b_approx. Рисование точечных графиков зависимости силы в верхнем датчике от силы в нижнем.

```
# 1mnk.py

import matplotlib.pyplot as plt
import numpy as np
from pylab import *

if __name__ == "__main__":
    time, pos_r, pos_l, pres_up_r, pres_up_l, pres_dn_r, pres_dn_l, trq_r, trq_l
= [], [], [], [], [], [], [], [], []
    f = open('1_1.txt','r')
    try:
    #работа с файлом
        text = f.readlines()
        for item in text:
            string = item.split()
            time.append(int(string[0]))
            pos_r.append(int(string[1]))
            pos_l.append(int(string[2]))
            pres_up_r.append(int(string[3]))
            pres_up_l.append(int(string[4]))
            pres_dn_r.append(int(string[5]))
            pres_dn_l.append(int(string[6]))
            trq_r.append(int(string[7]))
            trq_l.append(int(string[8]))
    finally:
        f.close()

    #делаем выборку до 150 строк таблицы
    pos_r_short, pdn_r_short = pos_r[0:150], pres_dn_r[0:150]

    fig, ax = plt.subplots()
    #точечный график зависимости правых нижних от правых верхних сил
    ax.scatter(pres_up_r, pres_dn_r, c="red", label="Правый, pres_up от
pres_dn")
    #заголовок для графика
    ax.set_title('Правый, pres_up от pres_dn')

    fig.set_figwidth(8)     #  ширина
    fig.set_figheight(8)    #  высота
    plt.savefig('pres_up_dn_right.png')

    figure()
    fig = plt.figure()
    ax = fig.add_subplot(111, label="scat")
    ax2 =fig.add_subplot(111, label="line1", frame_on=False)
    ax3 =fig.add_subplot(111, label="line2", frame_on=False)
    #точечный график зависимости левых нижних от левых верхних сил
    ax.scatter(pres_up_l, pres_dn_l, color="red", label="Левый, pres_up от
pres_dn")
    #размер списка
    N=len(pres_up_l)
```

```
xx = np.linspace(1000, 12000, N)
#предсказуемый график зависимости y=x синий
yy=xx
ax2.plot(xx, yy, color="C0")
ax2.set_xticks([])
ax2.set_yticks([])
#поиск средних значений
mx = np.array(pres_up_l).sum()/N
my = np.array(pres_dn_l).sum()/N
#скалярные произведения транспонированного вектора на обычный
a_2 = np.dot(xx.T, xx)/N
a_11 = np.dot(xx.T, yy)/N
#искомые коэффициенты
kk = (a_11 - mx*my)/(a_2 - mx**2)
bb = my - kk*mx
print('k_approx =',kk,'b_approx =',bb)
#полученный график зависимости y=x зеленый
#если предположение верно, зеленый будет лежать очень близко к синему
ff = np.array([kk*z+bb for z in range(N)])
ax3.plot(xx, ff, color="C2")
ax3.set_xticks([])
ax3.set_yticks([])
#заголовок для графика
ax.set_title('Левый, pres_up от pres_dn')

fig.set_figwidth(8)     #  ширина
fig.set_figheight(8)    #  высота
plt.savefig('pres_up_dn_left.png')
```

7.2. Чтение данных (параметров) из текстовых файлов и рисования графиков с сохранением их в файлы png.

2draw.py

```
import matplotlib.pyplot as plt
import numpy as np
from pylab import *

# построение графика с показом
def graph_1(name, l1, l2):
    plt.title(name)  # заголовок
    plt.xlabel("x")          # ось абсцисс
    plt.ylabel("y")     # ось ординат
    plt.grid()               # включение отображения сетки
    plt.plot(l1, l2)  # построение графика
    plt.show()

# построение графика с сохранением в файл
def graph_2(title, l1, l2, filename):
    plt.title(title)
    plt.xlabel("x")
    plt.ylabel("y")
    plt.grid()
    plt.plot(l1, l2)
    plt.savefig(filename)
    figure()

# построение точечного графика
def graph_dot(title, l1, l2, filename):
    plt.title(title)
    plt.xlabel("x")
    plt.ylabel("y")
    plt.grid()
    plt.scatter(l1, l2)
    plt.savefig(filename)
    figure()

# построение графика всех параметров от времени на одном графике
def graph_3(title, time, l1, l2, l3, l4, filename):
    plt.title(title)
    plt.xlabel("x")
    plt.ylabel("y")
    plt.grid()
    plt.plot(time, l1)
    plt.plot(time, l2)
    plt.plot(time, l3)
    plt.plot(time, l4)
    plt.savefig(filename)
    figure()


if __name__ == "__main__":
    time, pos_r, pos_l, pres_up_r, pres_up_l, pres_dn_r, pres_dn_l, trq_r, trq_l
= [], [], [], [], [], [], [], [], []
    f = open('1_1.txt', 'r')
    try:
       # работа с файлом
        text = f.readlines()
```

```
    for item in text:
        string = item.split()
        time.append(int(string[0]))
        pos_r.append(int(string[1]))
        pos_l.append(int(string[2]))
        pres_up_r.append(int(string[3]))
        pres_up_l.append(int(string[4]))
        pres_dn_r.append(int(string[5]))
        pres_dn_l.append(int(string[6]))
        trq_r.append(int(string[7]))
        trq_l.append(int(string[8]))
finally:
    f.close()
graph_2("Правый, pos от времени", time, pos_r, "pos__time_r.png")

graph_2(
    "Правый, pres_up от времени",
    time,
    pres_up_r,
    "pres_up__time_r.png")

graph_2(
    "Правый, pres_dn от времени",
    time,
    pres_dn_r,
    "pres_dn__time_r.png")

graph_2("Правый, trq от времени", time, trq_r, "trq__time_r.png")

graph_2("Левый, pres_up от времени", time,
        pres_up_l, "pres_up__time_l.png")

graph_2("Левый, pres_dn от времени", time,
        pres_dn_l, "pres_dn__time_l.png")
```

7.3. Создание датасета

```
# 3datasets.py

# фрагмент кода по созднию датасета шагов(функция + ее использование)
def pick_step(pres):
    #найдем скачок больше -20 по силе
    #это означает поднятие ноги
    step_indices = []
    for i in range(len(pres)-1):
        if pres[i+1] - pres[i] <= -20 and abs(pres[i+1] - pres[i]) >= 20:
            step_indices.append(i+1)
            step.append(pres[i+1])
    return step_indices



if __name__ == "__main__":
    time, pos_r, pos_l, pres_up_r, pres_up_l, pres_dn_r, pres_dn_l, trq_r,
trq_l, type_ = read_file('test30_09_nums_only.txt')
    # выбираем тип 1 - это шаги
    pos_r_1, t1 = pick_type(pos_r, type_, time, 1)
    pos_l_1, t1 = pick_type(pos_l, type_, time, 1)
    pres_dn_r_1, t1 = pick_type(pres_dn_r, type_, time, 1)
    pres_dn_l_1, t1 = pick_type(pres_dn_l, type_, time, 1)
    #выбираем индексы, опредденные силой Fdn под правой ногой
    step_indices = pick_step(pres_dn_r_1)
    print(step_indices)
    steps_arr = []
    for i in range(len(step_indices)-1):
        step = [1]
        #если шаг среднего размера - от 19 до 25 позиций по времени
        if step_indices[i+1] - step_indices[i] >= 19 and step_indices[i+1] -
step_indices[i] <= 25:
            for j in range(step_indices[i], step_indices[i+1]):
                step.append([ pos_r_1[j], pos_l_1[j], pres_dn_r_1[j],
pres_dn_l_1[j] ])
            print('\nstep',i,'=',step)
            steps_arr.append(step)

    myFile = open('example1.csv', 'w')
    with myFile:
        writer = csv.writer(myFile)
        writer.writerows(steps_arr)
```

7.4. Обучение модели

```
# 4classify.pyS
import pandas as pd
from sklearn.metrics import classification_report
from sklearn.metrics import confusion_matrix
from sklearn.metrics import accuracy_score
from sklearn.model_selection import train_test_split
from sklearn.neighbors import KNeighborsClassifier
from sklearn.linear_model import LogisticRegression
from sklearn.svm import SVC

data = pd.read_csv('dataset.csv')
```

```
# Проверяем, вс�' ли правильно загрузилось

print(data.head(5))
# ".iloc" принимает row_indexer, column_indexer
X = data.iloc[:,:-1].values
# Теперь выделим нужный столбец
y = data['result']
# test_size показывает, какой объем данных нужно выделить для тестового набора
# Random_state �" просто сид для случайной генерации
# Этот параметр можно использовать для воссоздания определ�'нного результата:
X_train, X_test, y_train, y_test = train_test_split(X, y, test_size=0.25,
random_state=27)

print(X_train)
print(y_train)

# метод опорных векторов
SVC_model = SVC()
# метод k-ближайших соседей
# �' KNN-модели нужно указать параметр n_neighbors
KNN_model = KNeighborsClassifier(n_neighbors=4)
# логистическая регрессия
logisticRegr = LogisticRegression()



SVC_model.fit(X_train, y_train)
KNN_model.fit(X_train, y_train)
logisticRegr.fit(X_train, y_train)



SVC_prediction = SVC_model.predict(X_test)
KNN_prediction = KNN_model.predict(X_test)
LogReg_prediction = logisticRegr.predict(X_test)

# Оценка точности �" простейший вариант оценки работы классификатора
print("accuracy_svc =", accuracy_score(SVC_prediction, y_test))
print("accuracy_knn =", accuracy_score(KNN_prediction, y_test))
print("accuracy_logreg =", accuracy_score(LogReg_prediction, y_test))
# Но матрица неточности и отч�'т о классификации дадут больше информации о
производительности
print("confusion_matrix SVC:\n", confusion_matrix(SVC_prediction, y_test))
print("confusion_matrix KNN:\n", confusion_matrix(KNN_prediction, y_test))
print("confusion_matrix LogReg:\n", confusion_matrix(LogReg_prediction, y_test))

print("classification_report SVC: \n", classification_report(SVC_prediction,
y_test))
print("classification_report KNN: \n", classification_report(KNN_prediction,
y_test))
print("classification_report LogReg: \n",
classification_report(LogReg_prediction, y_test))
```

7.5. Код для снятия и вывода данных на Delhi:

// draw.pas

```
unit draw;

interface
USES
  Windows, SysUtils, Stdctrls, Graphics, Dialogs, Messages;

VAR
  CanvT,CanvF: TCanvas;
//  Nc:byte;          //number of cells
  Rgn:TRect;         {rectangle to draw}
  FHeight,FWidth:integer;
  gdx:byte;

 procedure InitGraphT(Cnv:TCanvas; Left,Top,Width,Height:integer);
 procedure InitGraphF(Cnv:TCanvas; Left,Top,Width,Height:integer);
 procedure GrTMoveTo(x,y:integer);
 procedure GrTLineTo(x,y:integer);
 procedure GrFMoveTo(x,y:integer);
 procedure GrFLineTo(x,y:integer);
 procedure GrFPoint(x,y:integer);

implementation
var
GrTx0,GrTy0,GrFx0,GrFy0,GrXm,GrYm:integer;



Procedure InitGraphT(Cnv:TCanvas; Left,Top,Width,Height:integer);
begin
 Rgn.left:=Left; Rgn.Top:=Top;
 Rgn.Right:=Left+Width; Rgn.Bottom:=Top+Height;
 GrTx0:=Rgn.left+Width div 2; GrTy0:=Rgn.Top+Height div 2;
 GrXm:=Width-20; GrYm:=Height -10;
 CanvT:=Cnv;
 with CanvT do begin
  Brush.Color:=ClWhite; FillRect(Rgn);
  Pen.Color:=ClBlack;
  MoveTo(Left,GrTy0); LineTo(Left+Width,GrTy0);
  MoveTo(GrTx0,Top); LineTo(GrTx0,Top+Height);
  MoveTo(GrTx0+gdx*100,GrTy0-15); LineTo(GrTx0+gdx*100,GrTy0+15);      //1 sec
 end;
end;

Procedure InitGraphF(Cnv:TCanvas; Left,Top,Width,Height:integer);
begin
 Rgn.left:=Left; Rgn.Top:=Top;
 Rgn.Right:=Left+Width;  Rgn.Bottom:=Top+Height;
 GrFx0:=Rgn.left+5; GrFy0:=Rgn.Bottom-5;
 GrXm:=Width-10; GrYm:=Height-10;
 CanvF:=Cnv;
 with CanvF do begin
   Brush.Color:=ClWhite; FillRect(Rgn);
   Pen.Color:=ClBlack;
   MoveTo(GrFx0,GrFy0); LineTo(GrFx0+GrXm,GrFy0);
   MoveTo(GrFx0,GrFy0); LineTo(GrFx0,GrFy0-GrYm);
 end;
end;
```

```
 procedure GrTMoveTo(x,y:integer);
 begin
 CanvT.MoveTo(GrTx0+x,GrTy0-y);
 end;

 procedure GrTLineTo(x,y:integer);
 begin
 CanvT.LineTo(GrTx0+x,GrTy0-y);
 end;

 procedure GrFMoveTo(x,y:integer);
 begin
 CanvF.MoveTo(GrFx0+x,GrFy0-y);
 end;

 procedure GrFLineTo(x,y:integer);
 begin
 CanvF.LineTo(GrFx0+x,GrFy0-y);
 end;

procedure GrFPoint(x,y:integer);
begin
 CanvF.Ellipse(GrFx0+x-3,GrFy0-y-3,GrFx0+x+3,GrFy0-y+3);
end;

begin
 gdx:=3;
end.
```

7.6. Код на C для Arduino для управления одной из ног:

```
// CAN_motor.ino

#include <SPI.h>
#include "mcp_can.h"
#include <Wire.h>
#include "HX711.h"
#include <AS5600.h>
int16_t IDMotor=0x03;   // right leg 0x32,  left leg 0x42

const int spiCSPin = 4;
AS5600 ams5600;  // magnetic sensor
int ang, lang = 0;
// HX711 circuit wiring
const int LOADCELL1_DOUT_PIN =7; //  7 9
const int LOADCELL1_SCK_PIN = 6;  //  6 8
HX711 scale1;
byte bb=98;
byte bc=99;
byte bd=100;
uint32_t currentTime,previousTime,delta;
uint8_t FEEDBACK_EN = 0;
uint32_t timeInterval = 33;  // 3*33=99 ms one full cycle
unsigned char inByte,cmd,flSend;
uint8_t Step=0;
//float Pres1,Pres2;
uint16_t cPos, cVel,cTrq,cKnee;
uint16_t Pres1,Pres2;
```

```
MCP_CAN CAN(spiCSPin);
  unsigned char Mdat[8] = {0x80,0x00,0x80,0x00,0x00,0x01,0x48,0x00};
  unsigned char StartMdat[8] = {0xFF,0xFF,0xFF,0xFF,0xFF,0xFF,0xFF,0xFC};
  unsigned char StopMdat[8] = {0xFF,0xFF,0xFF,0xFF,0xFF,0xFF,0xFF,0xFD};
  unsigned char ZeroMdat[8] = {0xFF,0xFF,0xFF,0xFF,0xFF,0xFF,0xFF,0xFE};


uint8_t checkCAN()
{
  if( CAN_MSGAVAIL == CAN.checkReceive() )
  {
    uint8_t len = 0;
    uint8_t buf[8] = {0};
//    int16_t Dat1 = 0, Dat2 = 0, Dat3 = 0;
   digitalWrite(LED_BUILTIN, HIGH);
    CAN.readMsgBuf(0,&len, buf);
    unsigned long canId = CAN.getGPI();
    cPos=(buf[1]<<8)+buf[2]; cVel=(buf[3]<<4)+(buf[4]>>4);
cTrq=((buf[4]&0x0F)<<8)+buf[5];
   /* Dat1 = unpack16(  buf );
Dat2 = unpack16( &buf[2] );
Dat3 = unpack16( &buf[4] );*/

//    ReceiveCAN( canId, Dat1, Dat2, Dat3);
//      if( canId == IDSetup ) {  bme1.begin(0x76);   delay(100);
bme2.begin(0x77);  delay(100);}
    return 0;
  }
  else
  {
    return 1;
  }
}

void setup()
{
 previousTime=millis();
  Serial1.begin(9600); delay(100);
 Wire.begin(); delay(100);
  scale1.begin(LOADCELL1_DOUT_PIN, LOADCELL1_SCK_PIN); delay(100);
  CAN.begin(CAN_1000KBPS,0,0);  delay(100);
//  CAN.trySendMsgBuf(IDMotor,0,0,8,StartMdat); //(ID, 0, 8, stmp);
 cmd=2; flSend=0; Step=0;
}

void loop()
{
  long reading;
unsigned char val;
uint16_t vl;
  checkCAN();
  currentTime = millis();
  delta = currentTime - previousTime;
  //посылка данных
  if ( delta  > timeInterval)
  {
    previousTime = currentTime; Step++;
    if (Step==1) {      cKnee=ams5600.getRawAngle(); //CAN_Setup(IDSens, idat1,
idat2, idat3);
    if (flSend) {
      Serial1.write(bb); Serial1.write(bb);
     Serial1.write(cPos>>8); Serial1.write(cPos);  Serial1.write(cVel>>8);
Serial1.write(cVel);
```

```
     Serial1.write(cTrq>>8); Serial1.write(cTrq);Serial1.write(Pres1>>8);
Serial1.write(Pres1);
      Serial1.write(Pres2>>8); Serial1.write(Pres2); Serial1.write(cKnee>>8);
Serial1.write(cKnee);
       Serial1.write(bc); Serial1.write(bd); Serial1.write(bc);
Serial1.write(bd);
   /* Serial.write(cPos>>8); Serial.write(cPos);
Serial.write(cVel>>8); Serial.write(cVel);
Serial.write(cTrq>>8); Serial.write(cTrq);*/
    digitalWrite(LED_BUILTIN, LOW);}
  }      //cKnee !!!
//     Serial.print(idat2);  Serial.print("    ");Serial.println(idat3);}
    if (Step==2) { if (scale1.is_ready()) { scale1.set_gain(128);  //next chan a
        reading = scale1.read(); Pres1=reading;}
    }
//        CAN_Setup(0x0022, 1000, 5000, 100);}

    if (Step>2) {Step=0;
    if (scale1.is_ready()) {scale1.set_gain(32);  //next chan b
        reading = scale1.read(); Pres2=reading;}
//        digitalWrite(LED_BUILTIN, HIGH);
        CAN.sendMsgBuf(IDMotor,0,8,Mdat);
//        if (scale2.is_ready()) {reading = scale2.read()>>4; idat3=reading;}
    }
//         CAN.trySendMsgBuf(0x0001,0,0,8,dat); //(ID, 0, 8, stmp);
//         CAN.trySendMsgBuf(IDMotor,0,0,8,dat);} //(ID, 0, 8, stmp);
  //проверка приема компьютером
   while (Serial1.available()>0) {
         inByte = Serial1.read();
           switch (inByte) {
            case 0: flSend=0; CAN.sendMsgBuf(IDMotor,0,8,StopMdat); break; //
101 -> 'e'
            case 1: flSend=1;  CAN.sendMsgBuf(IDMotor,0,8,ZeroMdat);
//delay(10);
                              CAN.sendMsgBuf(IDMotor,0,8,StartMdat); break; // 98
-> 'b'
            case 2: cmd=2; break; //position control
            case 3: cmd=3; break; //velocity control
            case 4: cmd=4; break; //Kp control
            case 5: cmd=5; break; //Kd control
            case 6: cmd=6; break; //torque control
default: val=inByte-10;  // !!! torque control
                switch (cmd) {
                  case 2: vl=0x8000+30*(val-64); Mdat[0]=vl>>8; Mdat[1]=vl;
break;
                  case 3: break;
                  case 4: Mdat[4]=val; break;
                  case 5: break;
                  case 6: Mdat[6]=(Mdat[6]&0xF0)|(val>>3); Mdat[7]=val<<5;
break;
                }

           }
   }


  }
}
```