Московский государственный университет имени М.В. Ломоносова
Факультет вычислительной математики и кибернетики
Кафедра нелинейных динамических систем и процессов управления

Ван Юйфэн

Применение методов машинного обучения для управления многозвенными роботами

МАГИСТЕРСКАЯ ДИССЕРТАЦИЯ

Научный руководитель:
д.ф-м.н., Заведующий кафедрой НДСиПУ,
В.В. Фомичёв

Москва, 2024

Содержание

1 Введение

В последние годы, вдохновляясь реальными змеями, значительное внимание уделяется управлению движением роботов-змей. Обладая длинным и узким телом, а также множеством степеней свободы, роботы-змеи могут имитировать движения биологических змей и реализовывать гибкие геометрические изменения [1]. Благодаря этому они способны выполнять различные задачи, недоступные другим типам роботов, и адаптироваться к неизвестным и сложным условиям окружающей среды. Роботы-змеи находят применение в самых разных областях, например, при ликвидации последствий катастроф, в спасательных операциях, разведке, медицинской хирургии и сложных космических операциях [2].

Змеи испытывают сопротивление на земле от объектов типа камней и эффективно используют брюшные чешуйки для продвижения вперед благодаря трению. Большая часть современных исследований сконцентрирована на колесных роботах-змеях, которые относительно просты в управлении, однако не могут воспроизвести модели движения биологических змей и адаптироваться к сложным и изменчивым условиям [3]. В этой работе мы фокусируемся на бесколесном роботе-змее, чье движение более напоминает движения настоящих змей. Робот передвигается благодаря трению между его шатунами и землей. Существует два типа трения между змеей и землей: кулоновское трение, зависящее от массы, и вязкое трение, зависящее от скорости [4].

Рис. 1: Робот-змея

Змея-робот состоит из множества соединенных звеньев жесткого тела, его система имеет большое количество степеней свободы, в то время как регулятор имеет малую размерность, что делает систему сложноуправляемой. Сначала необходимо определить математическую модель динамики робота, которая точно описывает вза-

имосвязь между состоянием робота, трением и выходами регулятора. Состояние робота должно включать длины отдельных звеньев, их массы, положения, скорости, углы между звенями и т.д. Во второй главе мы представим математическую модель динамики робота и подробно опишем процесс построения этой модели, заканчивая получением динамических уравнений системы.

В этой статье мы рассмотрим две задачи: движение робота-змеи по прямой и движение робота-змеи к целевой точке. Для каждой задачи мы попытаемся найти решение на обоих симуляторах.

Мы будем исследовать целесообразность движения робота-змеи путем построения математической модели на симуляторе Simulink, а также немодельные методы управления путем построения модели робота и неизвестной среды, более похожей на реальность, на симуляторе MuJoCo.

Управление бионическими роботами часто достигается путем моделирования животных походок. Один из ключевых механизмов движения, используемый биологическими змеями для передвижения вперед, - это боковая ундуляция. В процессе этого движения змея создает периодические волнообразные изгибы по всему телу от головы к хвосту. Для имитации этого движения в роботе-змее угловые изменения каждого из его сегментов настраиваются согласно синусоидальной функции. Демонстрация осуществимости этой техники включает в себя изучение различных стратегий управления и выполнение задач, таких как перемещение вперед, изменение скорости и следование заданной траектории. Результаты этих исследований, представленные в разделе 5.1 и 5.2, подтверждают осуществимость управления с использованием симулятора.

Однако в реальных условиях окружающей среды, где характеристики, такие как модели трения и коэффициенты трения, могут быть неопределенными и изменчивыми, представленные методы могут оказаться недостаточными. Поэтому в исследовании рассматривается использование альтернативных подходов к управлению, включая метод чередования между быстрым и медленным управлением. Для каждого режима используется генетический алгоритм для оптимизации контрольных циклов, что позволяет роботу достигать быстрого движения по прямой. Детали этого метода изложены в разделе 5.3.

Для улучшения адаптивности и производительности робота-змеи в непредсказуемых условиях применяется алгоритм обучения с подкреплением PPO (Proximal Policy Optimization). Этот подход, проверенный в задачах непрерывного управления, фокусируется на оптимизации крутящего момента выходных осцилляторов, управ-

ляющих движением каждого сустава. В данной работе особое внимание уделяется анализу возможностей алгоритма PPO в контексте управления бионическими роботами, подтверждая его потенциал для повышения эффективности и устойчивости управления в динамичной среде.

Система управления робота-змеи состоит из системы осцилляторов, каждый из которых отвечает за генерацию периодического крутящего момента для управления движением робота. Мы разработали механизм обучения с подкреплением для создания нейронной сети с политикой, входами которой являются состояния робота в течение определенного периода времени, а выходами - определенные параметры осцилляторов, которые обеспечивают серию сигналов, позволяющих роботу автономно определять амплитуду колебаний на выходе каждого регулятора на основе текущего и предыдущих состояний. Мы будем использовать алгоритм PPO [5] для оптимизации параметров этих осцилляторов таким образом, чтобы робот мог эффективно имитировать боковое волнообразное движение биологической змеи, а также точно менять направление и двигаться к цели. В экспериментах выходы осцилляторов не просто фиксированы или заданы, а могут динамически регулироваться в зависимости от взаимодействия робота с окружающей средой в реальном времени. Решив задачу слежения за точкой цели, мы можем изначально реализовать задачу слежения за траекторией движения.

В разделе 5.4 данной работы подробно описаны результаты применения этого подхода к задаче отслеживания цели, демонстрирующие способность робота-змеи ориентироваться в окружающей среде и адаптировать свое поведение при изменении собственного состояния. Результаты экспериментов будут использованы для анализа конкретного влияния алгоритма PPO и оптимизации параметров осциллятора на производительность робота.

2 Описание модели змеи-робота

2.1 Положение и форма робота

Робот-змея на плоскости состоит из N соединительных звеньев длиной $2l$, поэтому существует $N-1$ точек соединения. Каждое звено имеет массу m, распределение массы равномерное, так что центроид звена (CM) находится в центральной точке, предполагая, что робот-змея движется по плоскости, всего существует $N+2$

степеней свободы, включая N степеней свободы, описывающих угол звена, и 2 степени свободы, описывающие положение робота [4].

Угол каждого звена относительно оси x:

$$\boldsymbol{\theta} = [\theta_1, \theta_2 ... \theta_N]^T \in \mathbb{R}^2 \tag{2.1.1}$$

Центр масс робота (если есть координаты центра масс каждого соединительного звена (x_i, y_i)):

$$\mathbf{p} = \begin{bmatrix} p_x \\ p_y \end{bmatrix} = \begin{bmatrix} \frac{1}{N}\sum_{i=1}^{N} x_i \\ \frac{1}{N}\sum_{i=1}^{N} y_i \end{bmatrix} \in \mathbb{R}^2 \tag{2.1.2}$$

Вектор координат оси x и вектор координат оси y:

$$\begin{aligned} \mathbf{X} &= [x_1, x_2, \ldots, x_N]^\top \in \mathbb{R}^N, \\ \mathbf{Y} &= [y_1, y_2, \ldots, y_N]^\top \in \mathbb{R}^N. \end{aligned} \tag{2.1.3}$$

Абсолютный угол (угол относительно оси x) звена равен θ_i. Угол между i-ым и $\{i+1\}$-ым звеном и $\phi_i = \theta_{i+1} - \theta_i$. Таким образом, связь между координатами соседних звеньев:

$$\begin{cases} x_{i+1} - x_i = l\cos\theta_i + l\cos\theta_{i+1}, \\ y_{i+1} - y_i = l\sin\theta_i + l\sin\theta_{i+1} \end{cases} \tag{2.1.4}$$

Путем матричного преобразования:

$$\begin{cases} \mathbf{DX} + l\mathbf{A}\cos\boldsymbol{\theta} = 0, \\ \mathbf{DY} + l\mathbf{A}\sin\boldsymbol{\theta} = 0 \end{cases} \tag{2.1.5}$$

где:

$$\mathbf{A} = \begin{bmatrix} 1 & 1 & 0 & \ldots & 0 \\ 0 & 1 & 1 & \ldots & 0 \\ & & \ldots & & \\ 0 & 0 & \ldots & 1 & 1 \end{bmatrix} \in \mathbb{R}^{(N-1)\times N} \qquad \mathbf{D} = \begin{bmatrix} 1 & -1 & \ldots & 0 & 0 \\ 0 & 1 & -1 & \ldots & 0 \\ & & \ldots & & \\ 0 & 0 & \ldots & 1 & -1 \end{bmatrix} \in \mathbb{R}^{(N-1)\times N}$$

Поэтому векторы координат каждого соединительного звена:

$$\begin{cases} \mathbf{X} = \mathbf{T}^{-1} \begin{bmatrix} -l\mathbf{A}\cos\boldsymbol{\theta} \\ p_x \end{bmatrix} = -l\mathbf{K}^T \cos\boldsymbol{\theta} + \mathbf{e}p_x \\ \mathbf{Y} = \mathbf{T}^{-1} \begin{bmatrix} -l\mathbf{A}\sin\boldsymbol{\theta} \\ p_y \end{bmatrix} = -l\mathbf{K}^T \sin\boldsymbol{\theta} + \mathbf{e}p_y \end{cases} \tag{2.1.6}$$

где:

$$\mathbf{e} = \begin{bmatrix} 1 & 1 & \dots & 1 \end{bmatrix}^T \in \mathbb{R}^n$$

$$\mathbf{T} = \begin{bmatrix} \mathbf{D} \\ \frac{1}{N}\mathbf{e}^T \end{bmatrix} \in \mathbb{R}^{N\times N},$$

$$\mathbf{T}^{-1} = \left[\mathbf{D}^T \left(\mathbf{D}\mathbf{D}^T\right)^{-1} \mathbf{e}\right] \in \mathbb{R}^{N\times N}$$

$$\sin\boldsymbol{\theta} = \begin{bmatrix} \sin\theta_1 & \sin\theta_2 & \dots & \sin\theta_n \end{bmatrix}^T \in \mathbb{R}^n$$

$$\cos\boldsymbol{\theta} = \begin{bmatrix} \cos\theta_1 & \cos\theta_2 & \dots & \cos\theta_n \end{bmatrix}^T \in \mathbb{R}^n$$

$$\mathbf{K} = \mathbf{A}^T \left(\mathbf{D}\mathbf{D}^T\right)^{-1} \mathbf{D} \in \mathbb{R}^{n\times n}$$

Предположим, что робот-змея расположен в обратной последовательности, то есть голова является N-ым звеном.

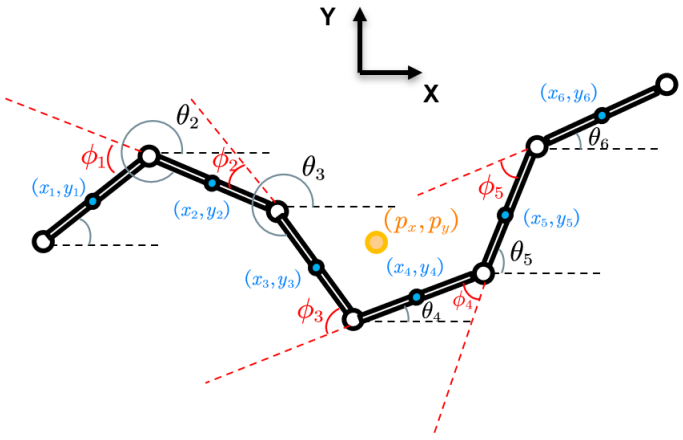


Рис. 2: Форма робота-змеи

2.2 Модель трения

Робот-змея, движущийся по плоской поверхности, постоянно изменяет форму своего тела, чтобы вызвать силы трения о землю, которые двигают робота вперед. При движении по плоскости для движения робота-змеи очень важно, чтобы силы трения о землю на звеньях были анизотропными, что означает, что коэффициент трения, описывающий силу трения в тангенциальном направлении звена c_t, μ_t(параллельно звену, т.е. вдоль оси x звена), отличается от коэффициента трения, описывающего силу трения в направлении, нормальном к звену c_n, μ_n (перпендикулярно звену, т.е. вдоль оси y звена). Это свойство трения проявляется у биологических змей. Мы решили использовать модель анизотропного вязкого трения. Мы предполагаем, что силы вязкого трения о землю действуют только на СМ звеньев. Трение, действующее на i-ое звено в раме, связано с направлением движения:

2.2.1 Модель анизотропного вязкого трения

Во-первых, мы рассмотрим модель анизотропного вязкого трения. Мы предполагаем, что силы вязкого трения о землю действуют только на ЦМ звеньев. Трение, действующее на i-ое звено в раме, связано с направлением движения:

$$\mathbf{f}_{R,i}^{i} = -\begin{bmatrix} c_t & 0 \\ 0 & c_n \end{bmatrix} \mathbf{v}_i \tag{2.2.1}$$

где $\mathbf{v}_i$ - скорость i-го звена в направлении движения. Трение, действующее на i-е звено в рамке x-y:

$$\mathbf{f}_{R,i} = \mathbf{R}_i \mathbf{f}_{R,i}^{i} = -\mathbf{R}_i \begin{bmatrix} c_t & 0 \\ 0 & c_n \end{bmatrix} \mathbf{v}_i = -\mathbf{R}_i \begin{bmatrix} c_t & 0 \\ 0 & c_n \end{bmatrix} \mathbf{R}_i^T \begin{bmatrix} \dot{x}_i \\ \dot{y}_i \end{bmatrix} \tag{2.2.2}$$

где:

$$\mathbf{v}_i = \mathbf{R}_i \begin{bmatrix} \dot{x}_i \\ \dot{y}_i \end{bmatrix}, \mathbf{R}_i = \begin{bmatrix} \cos\theta_i & -\sin\theta_i \\ \sin\theta_i & \cos\theta_i \end{bmatrix}$$

После преобразования матрицы:

$$\mathbf{f}_{R,i} = \begin{bmatrix} \mathbf{f}_{R,x,i} \\ \mathbf{f}_{R,y,i} \end{bmatrix} = -\begin{bmatrix} c_t \cos^2\theta_i + c_n \sin^2\theta_i & (c_t - c_n)\sin\theta_i \cos\theta_i \\ (c_t - c_n)\sin\theta_i \cos\theta_i & c_t \sin^2\theta_i + c_n \cos^2\theta_i \end{bmatrix} \begin{bmatrix} \dot{x}_i \\ \dot{y}_i \end{bmatrix} \tag{2.2.3}$$

Силы вязкого трения глобальной рамы на всех звеньях:

$$\mathbf{f}_R = \begin{bmatrix} \mathbf{f}_{R,x} \\ \mathbf{f}_{R,y} \end{bmatrix} = - \begin{bmatrix} c_t \left(\mathbf{C}_\theta\right)^2 + c_n \left(\mathbf{S}_\theta\right)^2 & \left(c_t - c_n\right) \mathbf{S}_\theta \mathbf{C}_\theta \\ \left(c_t - c_n\right) \mathbf{S}_\theta \mathbf{C}_\theta & c_n \left(\mathbf{C}_\theta\right)^2 + c_t \left(\mathbf{S}_\theta\right)^2 \end{bmatrix} \begin{bmatrix} \dot{\mathbf{X}} \\ \dot{\mathbf{Y}} \end{bmatrix} \in \mathbb{R}^{2N} \tag{2.2.4}$$

где:

$$\mathbf{S}_\theta = \operatorname{diag}(\sin \boldsymbol{\theta}) \in \mathbb{R}^{n \times n}$$

$$\mathbf{C}_\theta = \operatorname{diag}(\cos \boldsymbol{\theta}) \in \mathbb{R}^{n \times n}$$

2.2.2 Модель анизотропного кулоновского трения

Далее мы рассмотрим модель анизотропного кулоновского трения.

Как и в модели вязкого трения, мы определяем силу кулоновского трения, действующую на i-е звено, как.

$$\mathbf{f}_{R,i}^i = -mg \begin{bmatrix} \mu_t & 0 \\ 0 & \mu_n \end{bmatrix} sgn(\mathbf{v}_i) \tag{2.2.5}$$

где $\mathbf{v}_i$ - скорость i-го звена в направлении движения.Трение, действующее на i-е звено в рамке x-y:

$$\mathbf{f}_{R,i} = \mathbf{R}_i \mathbf{f}_{R,i}^i = -mg \mathbf{R}_i \begin{bmatrix} \mu_t & 0 \\ 0 & \mu_n \end{bmatrix} sgn(\mathbf{v}_i) = -\mathbf{R}_i \begin{bmatrix} \mu_t & 0 \\ 0 & \mu_n \end{bmatrix} \mathbf{R}_i^T sgn\left(\begin{bmatrix} \dot{x}_i \\ \dot{y}_i \end{bmatrix} \right) \tag{2.2.6}$$

Силы вязкого трения глобальной рамы на всех звеньях:

$$\mathbf{f}_R = \begin{bmatrix} \mathbf{f}_{R,x} \\ \mathbf{f}_{R,y} \end{bmatrix} = -mg \begin{bmatrix} \mu_t \mathbf{C}_\theta & -\mu_n \mathbf{S}_\theta \\ \mu_t \mathbf{S}_\theta & \mu_n \mathbf{C}_\theta \end{bmatrix} sgn\left(\begin{bmatrix} \mathbf{C}_\theta & \mathbf{S}_\theta \\ -\mathbf{S}_\theta & \mathbf{C}_\theta \end{bmatrix} \begin{bmatrix} \dot{\mathbf{X}} \\ \dot{\mathbf{Y}} \end{bmatrix} \right) \in \mathbb{R}^{2N} \tag{2.2.7}$$

2.3 Динамика робота

Теперь приведем уравнение движения робота в терминах углового ускорения $\ddot{\boldsymbol{\theta}}$ угла поворота звена и ускорения $\ddot{\boldsymbol{p}}$ положения центра массы. Согласно второму закону Ньютона, силы на каждом звене в направлениях x и y следующие:

$$m\ddot{x}_i = f_{R,x,i} + h_{x,i} - h_{x,i-1} \quad , \quad m\ddot{y}_i = f_{R,y,i} + h_{y,i} - h_{y,i-1} \tag{2.3.1}$$

где $h_{z,i}, h_{z,i-1}$ представляют совместные ограничения $\{i-1\}$-ого звена и $\{i+1\}$-ого звена соответственно в направлении z.

$$m\ddot{\mathbf{X}} = \mathbf{f}_{R,x} + \mathbf{D}^T\mathbf{h}_x \quad , \quad m\ddot{\mathbf{Y}} = \mathbf{f}_{R,y} + \mathbf{D}^T\mathbf{h}_y \tag{2.3.2}$$

где $\mathbf{h}_x = [h_{x,1}, ..., h_{x,N}]^T \in \mathbb{R}^N$,$\mathbf{h}_y = [h_{y,1}, ..., h_{y,N}]^T \in \mathbb{R}^N$

Возьмите вторую производную от приведенной выше формулы (???):

$$\mathbf{D}\ddot{\mathbf{X}} = l\mathbf{A}(\mathbf{C}_\theta\dot{\boldsymbol{\theta}}^2 + \mathbf{S}_\theta\ddot{\boldsymbol{\theta}}) \quad , \quad \mathbf{D}\ddot{\mathbf{Y}} = l\mathbf{A}(\mathbf{S}_\theta\dot{\boldsymbol{\theta}}^2 - \mathbf{C}_\theta\ddot{\boldsymbol{\theta}}) \tag{2.3.3}$$

Поскольку $\mathbf{e}^T\mathbf{D}^T = 0$, мы можем сделать вывод, найдя вторую производную от $\mathbf{p}$ – ускорение робота в целом в направлениях осей x и y связано только с вектором трения $\mathbf{f}_R$:

$$\begin{bmatrix} \ddot{p}_x \\ \ddot{p}_y \end{bmatrix} = \frac{1}{N}\begin{bmatrix} \mathbf{e}^T\ddot{\mathbf{X}} \\ \mathbf{e}^T\ddot{\mathbf{Y}} \end{bmatrix} = \frac{1}{Nm}\begin{bmatrix} \mathbf{e}^T\mathbf{f}_{R,x} \\ \mathbf{e}^T\mathbf{f}_{R,y} \end{bmatrix} \tag{2.3.4}$$

Это уравнение просто утверждает, что ускорение ЦМ робота-змеи равно сумме внешних сил, действующих на робота, деленной на его массу. Баланс крутящего момента для звена i определяется следующим образом:

$$J\ddot{\theta}_i = u_i - u_{i-1} - l\sin\theta_i\,(h_{x,i} + h_{x,i-1}) + l\cos\theta_i\,(h_{y,i} + h_{y,i-1}) \tag{2.3.5}$$

где J - момент инерции.

Через вышеприведенные уравнения (2.3.2) и (2.3.3) мы можем получить уравнение силы ограничения сустава $\mathbf{h}_x$, $\mathbf{h}_y$:

$$\begin{aligned} \mathbf{h}_x &= \left(\mathbf{D}\mathbf{D}^\mathbf{T}\right)^{-1}\left(ml\mathbf{A}(\mathbf{C}_\theta\dot{\boldsymbol{\theta}}^2 + \mathbf{S}_\theta\ddot{\boldsymbol{\theta}}) - \mathbf{D}\mathbf{f}_{R,x}\right) \\ \mathbf{h}_y &= \left(\mathbf{D}\mathbf{D}^\mathbf{T}\right)^{-1}\left(ml\mathbf{A}(\mathbf{S}_\theta\dot{\boldsymbol{\theta}}^2 - \mathbf{C}_\theta\ddot{\boldsymbol{\theta}}) - \mathbf{D}\mathbf{f}_{R,y}\right) \end{aligned} \tag{2.3.6}$$

Объединив вышеприведенные уравнения, мы можем получить уравнения движения о $\ddot{\boldsymbol{\theta}}$ and $\ddot{\boldsymbol{p}}$:

$$\begin{cases} \mathbf{M}_\theta\ddot{\boldsymbol{\theta}} + \mathbf{W}\dot{\boldsymbol{\theta}}^2 - l\mathbf{S}_\theta\mathbf{K}\mathbf{f}_{R,x} + l\mathbf{C}_\theta\mathbf{K}\mathbf{f}_{R,y} = \mathbf{D}^\mathbf{T}\mathbf{u} \\ Nm\ddot{\boldsymbol{p}} = Nm\begin{bmatrix} \ddot{\boldsymbol{p}}_x \\ \ddot{\boldsymbol{p}}_y \end{bmatrix} = \begin{bmatrix} \mathbf{e}^T\mathbf{f}_{R,x} \\ \mathbf{e}^T\mathbf{f}_{R,y} \end{bmatrix} \end{cases} \tag{2.3.7}$$

где:

$$\mathbf{M}_\theta = J\mathbf{I}_N + ml^2\mathbf{S}_\theta\mathbf{V}\mathbf{S}_\theta + ml^2\mathbf{C}_\theta\mathbf{V}\mathbf{C}_\theta, \quad \mathbf{W} = ml^2\mathbf{S}_\theta\mathbf{V}\mathbf{C}_\theta - ml^2\mathbf{C}_\theta\mathbf{V}\mathbf{S}_\theta,$$
$$\mathbf{V} = \mathbf{A}^T\left(\mathbf{D}\mathbf{D}^\mathbf{T}\right)^{-1}\mathbf{A},$$

Мы выражаем угол, положение, скорость и угловую скорость каждого звена как состояние системы, а вход момента, действующее на соединение каждого звена, как управление, и получаем нелинейную систему, которая может точно представлять движение робота-змеи.

$$\mathbf{x} = \left[\boldsymbol{\theta}^T, \mathbf{p}^T, \dot{\boldsymbol{\theta}}^T, \dot{\mathbf{p}}^T\right]^T \in \mathbb{R}^{2N+4} \tag{2.3.8}$$

$$\dot{\mathbf{x}} = \begin{bmatrix} \dot{\boldsymbol{\theta}} \\ \dot{\mathbf{p}} \\ \ddot{\boldsymbol{\theta}} \\ \ddot{\mathbf{p}} \end{bmatrix} = \mathbf{F}\left(\mathbf{x}, \mathbf{u}\right) \tag{2.3.9}$$

3 Среда моделирования

В этой главе мы представим две среды для моделирования движения робота-змеи, расскажем о конкретных параметрах модели робота-змеи и среды, а также об их соответствующих преимуществах.

Мы сконструируем робот-змею с шестью звеньями и пятью контроллерами. Длина и масса каждого звена равны, а в каждой точке соединения звеньев расположен контроллер.

3.1 Моделирование математической модели на основе программы Matlab с симулятором Simulink

Во-первых, мы будем использовать симулятор Simulink для математического моделирования системы, описанной в главе II. Преимущество использования Simulink заключается в том, что система может быть точно смоделирована на основе физически выведенных математических моделей, что поможет нам лучше проверить осуществимость управления движением робота-змеи и обеспечить теоретическую основу для последующих исследований.

Мы построим среду в соответствии с моделью, описанные выше, а параметры среды и робота приведены в таблице 1. В процессе решения системы обыкновенных дифференциальных уравнений мы будем использовать решатель, ode45 который основан на методе Рунге-Кутты.

3.2 Моделирование модели на основе симулятора MuJoCo

Затем мы воспользуемся программой моделирования MuJoCo [6], чтобы смоделировать движение настоящего робота-змеи. Преимущества использования MuJoCo заключаются в его высоко оптимизированном физическом движке, который обеспечивает быстрое и точное моделирование физики, что чрезвычайно важно для моделирования сложных динамических систем, таких как роботы-змеи. MuJoCo разработан для работы с сильно нелинейными системами, обеспечивает быстрое обнаружение столкновений и моделирование трения, а также предоставляет мощные инструменты визуализации, что делает его идеальным для приложений робототехники.

Мы проведем эксперименты на симуляторе MuJoCo с методами управления, не основанными на точных физических моделях.

При разработке модели мы представляем звенья робота-змеи рядом прямоугольников одинаковой длины и массы, задаем коэффициент кулоновского трения с землей, задаем совместные моменты в точках соединения звеньев и добавляем угловые ограничения, а конкретные параметры показаны в таблице 2.

Таблица 1: Параметры в Matlab

количество звена N	6
длина звена $2l$	0.14
масса звена m	1
c_t	0.5
c_n	3
момент инерции J	0.0016

Таблица 2: Параметры в MuJoCo

количество звена N	6
длина звена l	0.08
масса звена m	1
μ_t	3
μ_n	5
момент инерции J	0.0016

4 Постановка задачи

В этом разделе мы подробно описываем постановку экспериментальных задач и методику их решения. Данное исследование посвящено двум основным проблемам управления движением роботов-змей :

4.1 Движение робота-змеи по прямой

Во-первых, целью задачи является способность робота-змеи непрерывно и стабильно двигаться по прямой траектории без внешних помех [7]. Эта способность является ключевой метрикой для оценки эффективности базовых алгоритмов управления движением робота-змеи .

Для достижения этой цели мы сначала применили стратегию управления на основе обратной связи на симуляторе Simulink с известными параметрами окружающей среды, в частности, использовали ПД-регулятора для управления угловыми изменениями между звеньями робота, чтобы реализовать цель движения по прямой.

Затем мы исследовали гибридный метод управления, основанный на быстрых и медленных преобразованиях управления и генетических алгоритмах, чтобы найти оптимальные параметры, которые могут заставить робота двигаться быстро и линейно в неизвестных условиях окружающей среды. Мы реализуем этот алгоритм на симуляторе MuJoCo.

4.2 Движение робота-змеи к целевой точке

Вторая задача заключалась в разработке алгоритма, который позволил бы роботу-змее распознавать целевую точку и автоматически перенаправлять свое движение для достижения этой точки [8]. Сложность этой задачи заключается в том, что робот должен уметь точно менять направление, сохраняя при этом плавность и стабильность движения.

Сначала мы проводим исследование на основе физической модели в симуляторе Simulink, где робот меняет направление движения путем постоянного переключения целевой точки и изменения выхода контроллера головного звена, учитывая известные параметры окружающей среды.

Вышеописанный метод не может адаптироваться к ситуации неизвестных параметров среды, а изменение только выхода головного звена не позволяет полностью использовать потенциал контроллеров робота для выполнения более сложных задач. Поэтому для выполнения задачи перемещения змеи к произвольной цели в неизвестной среде на симуляторе MuJoCo мы попробуем спроектировать нейронную сеть на основе состояния робота для изменения выхода каждого контроллера. Мы попытаемся спроектировать нейронную сеть на основе состояния робота для изменения выходов каждого контроллера для выполнения задачи перемещения змеи к произ-

вольной цели, а затем для выполнения задачи перемещения робота по произвольной траектории на основе этого.

5 Основные подходы к решению

5.1 Решение задачи движения робота по прямой с использованием с использованием метода виртуального управления ограничениями

5.1.1 Идея решения

Идея решения возникла из литератур [4] и [11]. Мы используем $\bar{\boldsymbol{\phi}} = [\phi_1, \phi_2 ... \phi_{N-1}, \theta_N]^T \in \mathbb{R}^N$ для комплексного представления угла между звеньями робота и угла θ_N, используемого для управления направлением движения, где:

$$\phi_i = \theta_i - \theta_{i-1}, i = 1, ..., N-1$$

$$\boldsymbol{\phi} = [\phi_1, \phi_2 ... \phi_{N-1}]^T \in \mathbb{R}^{N-1}$$

Мы заставляем робота двигаться вперед, управляя формой тела робота-змеи. Предположим, что в момент времени t желаемый угол $\phi_{1,ref}, \phi_{2,ref} ... \phi_{N-1,ref}$ задан:

$$\phi_{i,ref} = \alpha \sin(\lambda + (i-1)\delta), i = 1, ..., N-1 \tag{5.1.1}$$

$$\boldsymbol{\phi}_{ref} = [\phi_{1,ref}, \phi_{2,ref} ... \phi_{N-1,ref}]^T \in \mathbb{R}^{N-1}$$

Таким образом, мы можем превратить управление движением робота в задачу отслеживания ряда заданных углов $\phi_{i,ref}$. Эти заданные углы выступают в качестве виртуальных ограничений, направляющих робота к желаемому изменению формы, что позволяет ему выполнять задачу движения вперед.

Среди них положительная константа α определяет волнистость робота. Когда значение α становится больше, угол между двумя соседними звеньями робота в состоянии движения становится больше, и боковое колебание робота увеличивается.

Положительная константа δ связана с количеством звеньев робота, что обеспечивает волнообразное состояние формы тела робота, похожее на функцию синуса.

Функция $\lambda(t)$ управляет скоростью изменения угла наклона звеньев робота, тем самым может управлвлять скорость движения робота вперед. В идеале, когда функция $\lambda(t)$ является некоторой линейной функцией $\lambda(t) = \tilde{\lambda} \cdot t$, скорость движения робота v_t также сходится к константе [4].

Далее мы предположим, что λ является линейной функцией, и выведем уравнение управления для равномерного движения робота-змеи:

$$\mathbf{DM^{-1}D}^T\mathbf{u} = \left(\mathbf{DM^{-1}}\right)\left(\mathbf{M}_\theta\ddot{\boldsymbol{\theta}} + \mathbf{W}\dot{\boldsymbol{\theta}}^2 - l\mathbf{S}_\theta\mathbf{K}\mathbf{f}_{R,x} + l\mathbf{C}_\theta\mathbf{K}\mathbf{f}_{R,y}\right) \tag{5.1.2}$$

$$\mathbf{u} = \left(\mathbf{DM^{-1}D}^T\right)^{-1}\left(\mathbf{DM^{-1}W}\dot{\boldsymbol{\theta}}^2 - l\mathbf{DM^{-1}S}\mathbf{C}_\theta^T\mathbf{f}_R + \mathbf{D}\ddot{\boldsymbol{\theta}}\right) \tag{5.1.3}$$

Предположим, что $\tilde{\boldsymbol{\phi}} = \boldsymbol{\phi} - \boldsymbol{\phi}_{ref}$. Напишем асимптотически устойчивые обыкновенные дифференциальные уравнения для $\tilde{\boldsymbol{\phi}}$:

$$\ddot{\tilde{\boldsymbol{\phi}}} + k_d\dot{\tilde{\boldsymbol{\phi}}} + k_p\tilde{\boldsymbol{\phi}} = 0 \tag{5.1.4}$$

где k_d и k_p - положительные константы. Таким образом, при выполнении следующих условий, $\boldsymbol{\phi}$ стабильно равен 0, и форма тела робота-змеи может соответствовать желаемому углу и двигаться вперед с равномерной скоростью:

$$\mathbf{D}\ddot{\boldsymbol{\theta}} = \ddot{\boldsymbol{\phi}} = \ddot{\boldsymbol{\phi}}_{ref} - k_d(\dot{\boldsymbol{\phi}} - \dot{\boldsymbol{\phi}}_{ref}) - k_p(\boldsymbol{\phi} - \boldsymbol{\phi}_{ref}) \tag{5.1.5}$$

В итоге мы можем получить уравнение закона управления для робота-змеи, движущегося вперед с равномерной скоростью:

$$\begin{aligned}\mathbf{u} = \left(\mathbf{DM^{-1}D}^T\right)^{-1}\Big(&\mathbf{DM^{-1}W}\dot{\boldsymbol{\theta}}^2 - l\mathbf{DM^{-1}S}\mathbf{C}_\theta^T\mathbf{f}_R \\ &+\ddot{\boldsymbol{\phi}}_{ref} - k_d(\dot{\boldsymbol{\phi}} - \dot{\boldsymbol{\phi}}_{ref}) - k_p(\boldsymbol{\phi} - \boldsymbol{\phi}_{ref})\Big)\end{aligned} \tag{5.1.6}$$

5.1.2 Анализ результатов

Мы определяем параметры управления виртуальными ограничениями в таблице 3:

Мы задаем начальное состояние робота следующим образом: центр масс находится в начале координат, а все звенья параллельны оси Ox.

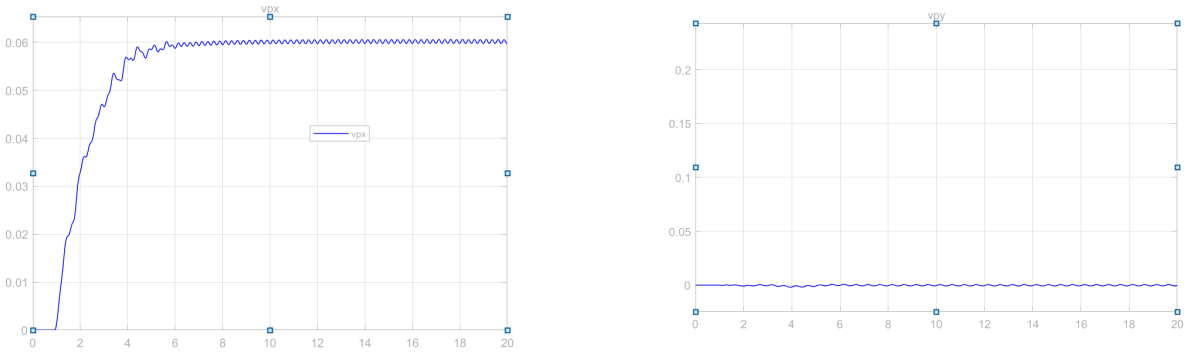
Таблица 3: Параметры управления виртуальными ограничениями

α	λ	δ
0.15	12.5838	1.2566

Мы ограничиваем диапазон выходного момента $[-1, 1]$ и фильтруем высокочастотный выходной сигнал с помощью фильтра низких частот [9]. Это позволит роботу сформировать эффективную змеевидную позицию в начале движения.

С помощью уравнения (5.1.6), управляющего движением робота-змеи, мы получаем следующие результаты:

Во-первых, показано изменение скорости центра масс робота в направлении оси x v_x и оси y v_y (на рисунке 3):



(a)

Рис. 3: Скорость цетра массы роботы

Видно, что через некоторое время робот-змея стремится к определенному значению скорости и выполняет задачу по перемещению вдоль оси x с равномерной скоростью.

Таким образом, виртуальное управление с ограничениями эффективно для данной модели робота-змеи, и он способен выполнять задачу по перемещению по прямой линии с равномерной скоростью.

Во-вторых, мы показываем численную вариацию регулятора робота на рисунке 4:

Видно, что после периода колебаний выход регулятора становится периодическим. В то же время, чем ближе регулятор к голове или хвосту робота, тем меньше максимальный выход регулятора.

На рисунке 5 показано различные морфологии робота-змеи во время движения:

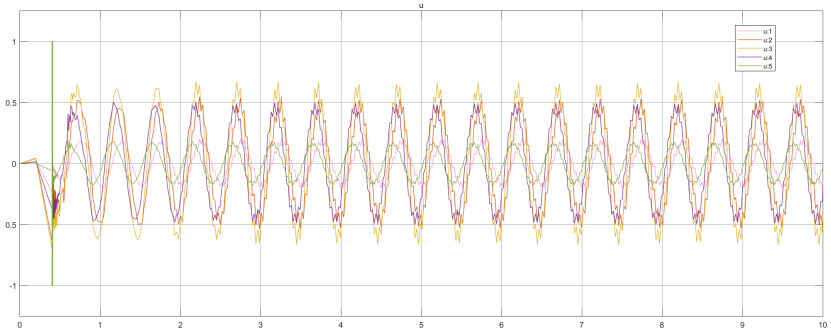


Рис. 4: Численная вариация регулятора робота

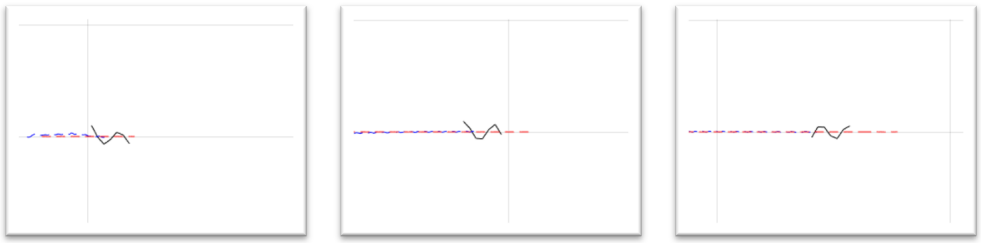


Рис. 5: Различные морфологии робота-змеи

5.2 Решение задачи движения робота к целевой точке с использованием скользящего режима

5.2.1 Идея решения

Мы управляем $\ddot{\theta}_N$, управляй крутящий момент в точке соединения головы, тем самым управлять направление движения робота.

Пусть θ_{ref} - ожидаемый угол, и $\tilde{\theta} = \theta_N - \theta_{ref}$, мы используем обыкновенное дифференциальное уравнение [10]:

$$\ddot{\tilde{\theta}} + k_d\dot{\tilde{\theta}} + k_p\tilde{\theta} = 0 \tag{5.2.1}$$

, $\tilde{\theta}$ является асимптотически устойчивым, так что мы можем сделать так, чтобы угол θ_N достиг желаемого значения. Установите поверхность скольжения:

$$S = \tilde{\theta} + K_n\dot{\tilde{\theta}} \tag{5.2.2}$$

Для того чтобы сделать $\tilde{\theta}$ и $\dot{\tilde{\theta}}$ экспоненциально устойчивыми, мы используем экспоненциальный закон достижения скользящего режима управления:

$$\dot{S} = -\epsilon_2 sgn(S) - k_2 S \tag{5.2.3}$$

Поэтому :

$$u_2 = (-\epsilon sgn(S) - k_2 S - K_n \dot{\tilde{\theta}}) \tag{5.2.4}$$

Таким образом, закон управления можно записать как:

$$\begin{aligned} \mathbf{u} = \left(\mathbf{DM^{-1}D}^T\right)^{-1} \Big(& \mathbf{DM^{-1}W}\dot{\boldsymbol{\theta}}^2 - l\mathbf{DM^{-1}SC}_\theta^T \mathbf{f}_R \\ & + \ddot{\boldsymbol{\phi}}_{ref} - k_d(\dot{\boldsymbol{\phi}} - \dot{\boldsymbol{\phi}}_{ref}) - k_p(\boldsymbol{\phi} - \boldsymbol{\phi}_{ref}) \\ & + \mathbf{b}(u_2 + k_d\dot{\theta}_N + k_p\theta_N)\Big) \end{aligned} \tag{5.2.5}$$

где: $\mathbf{b} = [0, 0, \ldots, 0, 1]^T \in \mathbb{R}^{n-1}$

5.2.2 Анализ результатов

Для задач отслеживания траектории, Мы можем получить желаемую траекторию, задав желаемую траекторию $\bar{\mathbf{p}}$, и использовать

$$\theta_{ref} = arctg(\frac{\bar{p}_x - p_x}{\bar{p}_y - p_y})$$

для получения целевого угла θ_{ref}, чтобы мы могли выполнить некоторые простые задачи отслеживания траектории [11].

На рисунке 6 показан робот-змея, отслеживающий траекторию $x^2+(y-1.5)^2 = 2.25$, где красная кривая фиксирует изменение желаемой траектории $\bar{\mathbf{p}}$. Синяя кривая отражает изменение центра масс робота.. Видно, что робот по-прежнему хорошо справляется с отслеживанием траекторий с более быстрым изменением угла.

5.3 Решение задачи движения робота по прямой с использованием метода изменения быстрого-медленного управления

5.3.1 Идея решения

Быстрое движение трехзвенного робота-змеи рассматривается в литературе [12]. Этот подход основан не на модели трения, а на изменении состояния робота-змеи с помощью большого управляющего момента для выполнения задачи локомоции. На модели многозвенного робота такой подход невозможен из-за сложности робота-змеи,

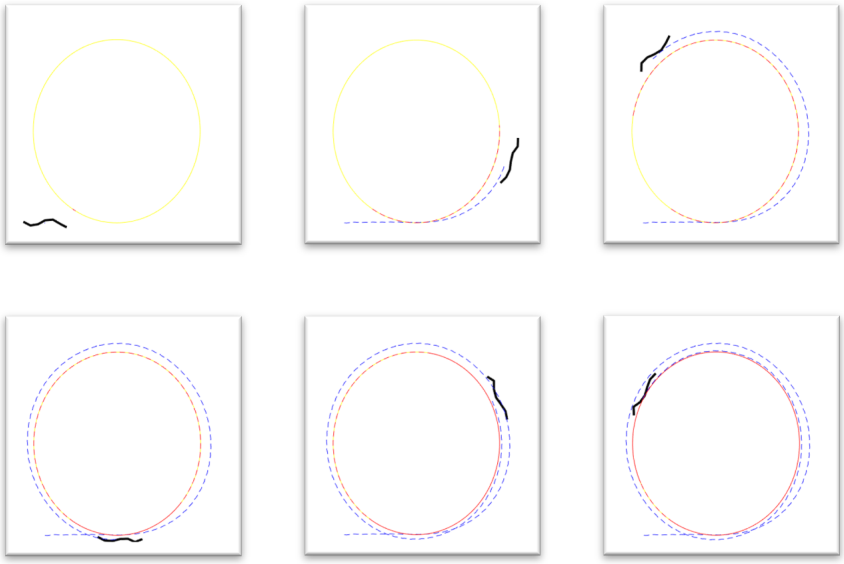


Рис. 6: Робот-змея, отслеживающий траекторию $x^2 + (y - 1.5)^2 = 2.25$

который представляет собой нелинейную систему. Однако это дает нам новую идею: мы попробуем преобразовать управляющий момент каждого звена для выполнения задачи перемещения по прямой линии с быстрым и медленным.

Наше решение заключается в следующем:

Для каждого регулятора задаем пять параметров, а именно: время начала управления T_i^0, период быстрого управления T_i^1, период медленного управления T_i^2, момент быстрого управления u_i^1 и момент медленного управления u_i^2.

В начале управления регуляторов между каждым звеном робота начинают периодическое движение в соответствии с их соответствующим временем запуска, сначала выдавая быстрый управляющий момент в периоде быстрого управления, а затем выдавая медленный управляющий момент в периоде медленного управления. Параметры регуляторов не связаны друг с другом.

Структура этого подхода показана на рисунке 7.

В режиме чередовании быстрого и медленного управления (изменения угла) в каждой совместной точке, нужно настроить сочетание медленного и быстрого движений. Для этого используется генетический алгоритм для поиска оптимальных параметров в каждой совместной точке для движения вперед. В этой задаче размерность параметров составляет 25.

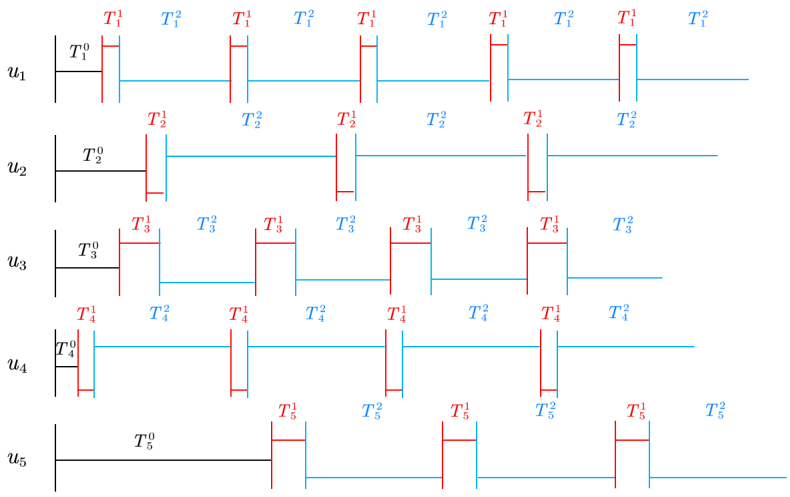


Рис. 7: Периодическое преобразование быстро-медленного управления

5.3.2 Генетический алгоритм

Генетический алгоритм - это алгоритм оптимизации, моделирующий естественный отбор [13]. Они решают оптимизационные задачи, имитируя механизмы наследования, мутации и естественного отбора в процессе биологической эволюции. В генетических алгоритмах решение абстрагируется как "особь" , а весь набор решений составляет "популяцию" . Каждая особь представлена строкой "хромосом" (т.е. параметров). Каждая итерация алгоритма имитирует цикл биологической эволюции, включающий процессы отбора, кроссинговера (спаривания и рекомбинации) и мутации.

Основные шаги генетического алгоритма следующие:

Инициализация: начальная популяция генерируется случайным образом.

Оценка: каждая особь в популяции оценивается для определения ее фитнес-функции.

Отбор: особи отбираются на основе их пригодности, и особи с высокой пригодностью имеют более высокую вероятность быть отобранными для следующего поколения.

Скрещивание: отобранные особи рекомбинируются путем кроссинговера хромосом для получения новых особей.

Мутация: изменение определенных генов у определенных особей с определенной вероятностью с целью внесения нового генетического разнообразия.

Новое поколение: новые особи образуют новую популяцию, которая заменяет старую.

Повторение: повторять шаги 2-6 до тех пор, пока не будет достигнуто условие завершения.

5.3.3 Процесс моделирования

Диапазоны параметров контроллеров в этой задаче приведены в таблице 3.

Для каждой "особи" в соответствии с параметрами ее "хромосомы" разрабатывается метод управления роботом-змеей и проводится полное моделирование на симуляторе MuJoCo.

Для оценки преимуществ и недостатков "особи" мы будем использовать следующую фитнес-функцию.

$$f = e^{-c \cdot p_y} \tag{5.3.1}$$

Если центр масс робота расположен ближе к оси x, его фитнес-функция будет выше.

Процесс оптимизации будет осуществляться в соответствии с алгоритмом, описанным в разделе 5.3.2. Некоторые параметры алгоритма заданы в таблице 4.

Значения параметров приведены в таблице 5.

Таблица 4

T_i^0	[0.0,1.0]
T_i^1	[0.0,0.3]
T_i^2	[0.3,1.5]
u_i^1	[-0.3,0.3]
u_i^2	[-0.03,0.03]

Таблица 5

Количество итераций	400
Количество особей в популяции	200
Количество выбранных особей	40

Таблица 6

T_i^0	[0.50,0.85,0.58,0.31,0.43]
T_i^1	[0.09,0.38,0.02,0.21,0.02]
T_i^2	[1.13,0.48,1.05,0.32,0.47]
u_i^1	[-0.03,-0.26,-0.12,0.24,0.34]
u_i^2	[0.023,0.038,0.001,-0.155,-0.027]

5.3.4 Анализ результатов

Параметры, полученные генетическим алгоритмом, применяются к симулятору MuJoCo, и на рисунке 8 будет показана траектория движения центра масс робота.

Видно, что робот в основном может двигаться в положительном направлении оси Ox.

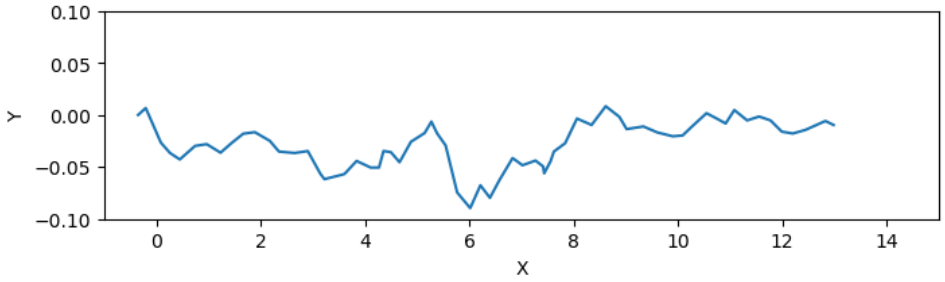


Рис. 8: Траектория движения по прямой

На рисунке 9 показаны различные модели робота во время движения.

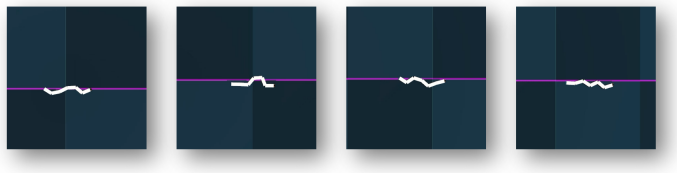


Рис. 9: Различные формы робота во время движения

Это доказывает эффективность нашего решения.

5.4 Решение задачи движения робота к целевой точке с использованием метода обучения с подкреплением и осциллятора

5.4.1 Осциллятор CPG

Центральный генератор упорядоченной активности (Central Pattern Generator) (ЦГУА,CPG) — вид биологических нейронных сетей (подразумевается, например, отдельная область в спинном мозге), отличительной особенностью которых является способность подавать ритмически упорядоченные моторные сигналы без обратной связи и без постоянного входного возбуждения [14].

Метод управления на основе CPG, который в последнее время широко применяется в бионических роботах, является новой бионической стратегией управления, которая имитирует принцип, согласно которому центральная нервная система позвоночных может спонтанно генерировать ритмические сигналы, и достигает сигналов высокой степени свободы с помощью небольшого количества управляющих параметров в верхних слоях, чтобы обеспечить сигналы управления движением для

распределенных контроллеров суставов роботов. Модель походки робота, построенная по принципу CPG, позволяет избежать зависимости от точности моделирования динамики и легче реализуется в сложном управлении.

Для подачи управляющих сигналов роботу-змее мы выбрали модель осциллятора Мацуока [15], которая способна обеспечить более плавное, стабильное и энергоэффективное управление движением, чем другие модели осцилляторов на основе CPG. Далее кратко описывается структура осциллятора Мацуока.

В модели осциллятора Матсуока каждый нейрон состоит из двух частей, которые моделируют возбуждающую (excitatory) и тормозную (inhibitory) активность нейронов, соответственно. Каждый нейрон в осцилляторе может быть представлен набором нелинейных разностных уравнений:

$$\begin{aligned}
\tau_r \dot{x}_i^e &= -x_i^e - a z_i^f - b y_i^e - \sum_{j=1}^{N} w_{ji} y_j^e + u_i^e \\
\tau_a \dot{y}_i^e &= z_i^e - y_i^e \\
\tau_r \dot{x}_i^f &= -x_i^f - a z_i^e - b y_i^f - \sum_{j=1}^{N} w_{ji} y_j^f + u_i^f \\
\tau_a \dot{y}_i^f &= z_i^f - y_i^f,
\end{aligned} \tag{5.4.1}$$

где:

$$z_i^q = g(x_i^q) = \max(0, x_i^q)$$

Переменные в модели:

x_i^e и x_i^f: представляют состояние (i)-го возбуждающего и тормозного нейрона, соответственно.

y_i^e и y_i^f: представляют выход (i) возбуждающего и тормозного нейрона, соответственно.

z_i^e и z_i^f: представляют состояние самовозбуждения (i) возбуждающего и тормозного нейрона соответственно.

Параметры в модели:.

τ_r: постоянная времени состояния нейрона (x), которая определяет скорость обновления состояния нейрона. Меньшее значение (τ_r) делает состояние более отзывчивым на изменения входного сигнала.

τ_a: постоянная времени самомотивированного состояния (z), которая контролирует скорость динамического изменения самомотивированного состояния.

a: сила связи между самовозбуждением и состоянием нейронов, которая определяет, насколько сильно состояние самовозбуждения подавляет состояние нейронов.

b: сила обратной связи от выхода нейрона (y) к состоянию нейрона (x), которая влияет на ингибирующий эффект выхода на состояние.

w_{ji}: синаптический вес, соединяющий выход (j) нейрона с (i) нейроном. Этот вес определяет, как выходы других нейронов влияют на состояние текущего нейрона.

u_i^e и u_i^f: внешние входы возбуждающего и тормозного нейронов, соответственно, которые используются для изменения амплитуды выходного сигнала нейрона.

Выходное уравнение для каждого нейрона осциллятора выглядит следующим образом:

$$u_i = z_i^e - z_i^f = g(x_i^e) - g(x_i^f) \tag{5.4.2}$$

Схема модели осциллятора показана на рисунке 10

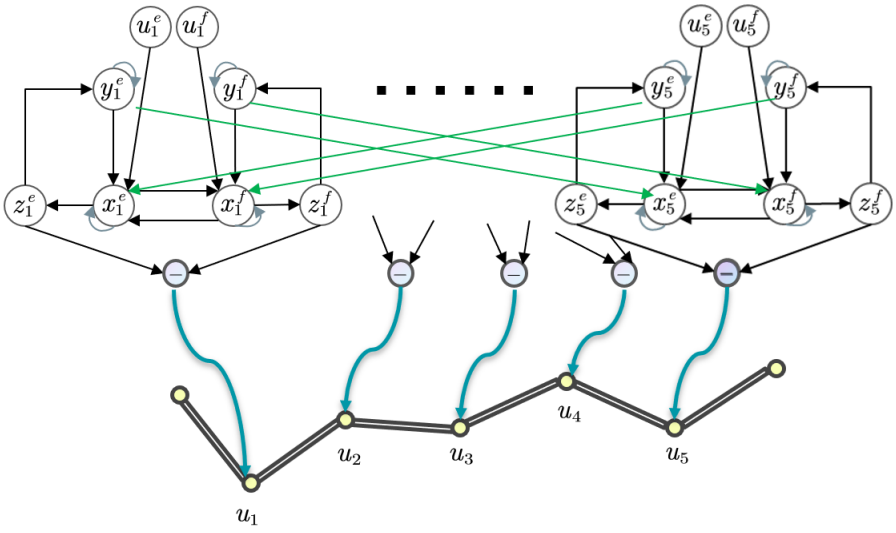


Рис. 10: Осциллятор Мацуока для робота-змеи

Регулируя параметры в модели осциллятора, мы можем получить набор периодических сигналов. Регулируя параметры u_i^e и u_i^f, мы можем изменять амплитуду

выходного сигнала осциллятора без изменения частоты, что дает теоретическую основу для управления роботом для изменения направления движения.

Если параметры u_i^e и u_i^f одинаковы для каждого контроллера, то амплитуды будут одинаковыми и выход осциллятора (на рисунке 11) будет похож на выход, полученный от ПД-регулятора в 5.1 (рис. 4).

Мы попробуем использовать выход этого осциллятора в качестве контроллера для точек соединения между звеньями робота-змеи, и результаты покажут, что робот может двигаться вперед за счет трения о землю, но не может контролировать направление. Однако этот эксперимент демонстрирует возможность использования осциллятора Мацуока для управления движением робота, и если мы сможем регулировать значения сигналов u_i^e и u_i^f для каждого нейрона (на рисунке 12), то сможем решить задачу перемещения робота в любом направлении.

Следующая задача - найти способ изменять сигнал осциллятора в зависимости от состояния робота и таким образом управлять направлением его движения.

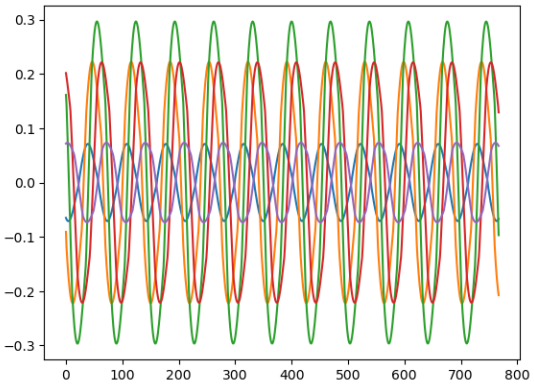


Рис. 11: Выходы осциллятор Мацуока при условия $u_i^e = u_i^f$

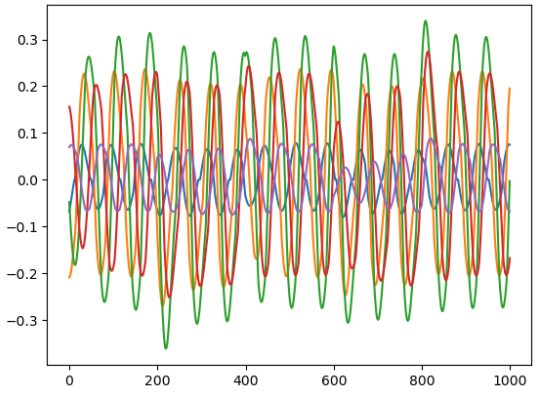


Рис. 12: Выходы осциллятор Мацуока при условия $u_i^e \neq u_i^f$

5.4.2 Алгоритм Policy Proximal Optimization

Обучение с подкреплением (Reinforcement Learning)- это метод машинного обучения, который позволяет агенту учить оптимальную политику путем проб и ошибок в среде. Процесс обучения с подкреплением является итеративным и рекурсивным, в котором интеллект постоянно пробует различные действия, получает обратную связь от среды, узнает, какие действия получают большее награду, и таким образом постепенно улучшает свою стратегию принятия действий [16] .

Задача RL моделируется как дискретно-непрерывный марковский процесс принятия решений (MDP), состоящий из (S, A, P, R, γ), где

S - непрерывное пространство состояний,

A - непрерывное пространство действий,

P - динамика переходов в среде, который определяет вероятности перехода из одного состояния в другое для данного действия $p(s_{t+1}|s_t, a_t)$.

R - функция награды, $R(s_t, a_t) = r_t \in \mathbb{R}$, связанные с текущим состоянием и действием

$\gamma \in [0, 1]$ - коэффициент дисконтирования, определяющий важность будущих наградов по отношению к текущей награде.

Политика π является стохастической и отображается из состояний в распределение действий $S \to A$. Цель агента - выучить политику $\pi(a_t|s_t)$, которая максимизирует ожидаемую награду:

$$J(\pi) = \mathbb{E}_{\tau\sim\pi}\left[\sum_{t=0}^{\infty}\gamma^t r_t|s_0 = s\right]. \tag{5.4.3}$$

где $(\tau \sim \pi)$ обозначает всю траекторию MDP (τ), созданную в соответствии с политикой (π), а (r_t) - награды на шаге (t).

В глубоком обучении с подкреплением политика представляется аппроксимацией нелинейной функции, т.е. нейронной сетью $\pi(a_t|s_t; \boldsymbol{\theta})$, параметризованной $\boldsymbol{\theta}$. На вход этой нейронной сети подается текущее состояние агента, а на выходе - вероятность выбора определенного значения действия. Метод поиска политики используют параметризованную политику и непосредственно ищут оптимальные параметры этой политики.

Одним из методов RL является метод актор-критик. Ключевая идея подходов актор-критик заключается в одновременном обучении функции ценности и политики. Критик оценивает действия актора на основе изученной функции ценности и предоставляет обратную связь, с помощью которой актор обновляет параметры своей политики. Предлагаемый подход следует алгоритм актор-критик и обучается как параметрам политики, так и аппроксиматору функции ценности. Это означает, что для функции ценности также необходима нейронная сеть $v(s_t; \mathbf{w})$, параметризованная $\mathbf{w}$.

Политика обучается с помощью алгоритма Proximal Policy Optimization (PPO) [5]. PPO - один из наиболее успешных алгоритмов RL, основанных на политике. Алгоритм PPO представляет собой приближенную версию оптимизации политики

в доверительном регионе (TRPO) с градиентами первого порядка. Цель алгоритма PPO улучшает устойчивость обучения, ограничивая изменение политики на каждой итерации. Задача оптимизации в PPO решается на основе следующей суррогатной функции потерь :

$$L^{CLIP}(\boldsymbol{\theta}) = \hat{\mathbb{E}}_t \left[\min \left(r_t(\boldsymbol{\theta}) \hat{A}_t, \text{clip}(r_t(\boldsymbol{\theta}), 1-\epsilon, 1+\epsilon) \hat{A}_t \right) \right] \tag{5.4.4}$$

где:

$r_t(\boldsymbol{\theta}) = \frac{\pi_\theta(a_t|s_t;\boldsymbol{\theta}))}{\pi_{\theta_{\text{old}}}(a_t|s_t;\boldsymbol{\theta}))}$ - это отношение вероятностей, показывающее, насколько новая политика π_θ отличается от старой политики $\pi_{\theta_{\text{old}}}$ в смысле вероятности принятия действия a_t в состоянии s_t.

Элемент $\hat{A}_t$ - это оценка преимущества в момент времени t, которая показывает выгоду от принятия действия a_t в состоянии s_t по сравнению с ожидаемым значением.

Чтобы вычислить преимущество $\hat{A}_t$, мы используем Generalized Advantage Estimation (GAE):

$$\hat{A}_t = \delta_t + (\gamma\lambda)\delta_{t+1} + \ldots + (\gamma\lambda)^{T-t+1}\delta_{T-1} \tag{5.4.5}$$

где $\delta_t = r_t + \gamma v(s_{t+1}; \mathbf{w})) - v(s_t; \mathbf{w})$ -временные различия, $v(s_t; \mathbf{w})$ - функция ценности, а γ и λ - коэффициенты дисконтирования для награды и преимущества, соответственно.

Используя функцию цели, политика обновляется путем градиентного восхождения по $L^{CLIP}(\boldsymbol{\theta})$. Градиент обрезанной цели определяется следующим образом:

$$\nabla_\theta L^{CLIP}(\boldsymbol{\theta}) = \hat{\mathbb{E}}_t \left[\min \left(\nabla_\theta \log \pi_\theta(a_t|s_t) \hat{A}_t, \nabla_\theta \text{clip}(r_t(\boldsymbol{\theta}), 1-\epsilon, 1+\epsilon) \hat{A}_t \right) \right] \tag{5.4.6}$$

Обновление параметров для нейронных сетей политики:

$$\boldsymbol{\theta_{new}} \leftarrow \boldsymbol{\theta_{old}} + \beta \nabla_\theta L^{CLIP}(\boldsymbol{\theta}) \tag{5.4.7}$$

Для нейронной сети ценности функция потерь имеет вид

$$L^{VF}(w) = \hat{\mathbb{E}}_t \left[(v(s_t; \mathbf{w}) - v_t^{target})^2 \right] \tag{5.4.8}$$

где $(v(s_t;;\mathbf{w})$ - оценка сетью текущих значений ценности состояния (s_t), а $v_t^{target})$ - целевое значение состояния, которое оценивается путем накопления дисконтированных награды.

Градиент функции потерь относительно параметра $\mathbf{w}$:

$$\nabla_w L^{VF}(\mathbf{w}) = \hat{\mathbb{E}}_t \left[2(v(s_t;\mathbf{w}) - v_t^{target}) \nabla_w v(s_t;\mathbf{w}) \right] \tag{5.4.9}$$

Обновление параметров для нейронных сетей ценности:

$$\mathbf{w}_{new} = \mathbf{w}_{old} - \alpha \nabla_w L^{VF}(\mathbf{w}) \tag{5.4.10}$$

где α и β - скорости обучения нейронной сети политики и нейронной сети ценности, соответственно.

Это правило обновления гарантирует, что политика не будет меняться слишком резко, что помогает поддерживать стабильный прогресс обучения, особенно в пространствах непрерывных действий.

Для непрерывных пространств действий политика $\pi_\theta(a|s)$ обычно параметризуется как гауссовское распределение, где нейронная сеть выдает среднее и ковариацию распределения. Такая параметризация позволяет алгоритму эффективно справляться со сложностями непрерывных пространств действий.

Тщательно настраивая такие параметры, как порог обрезания ϵ, коэффициент γ и λ для GAE, PPO может быть адаптирован к различным средам и задачам обучения с подкреплением.

5.4.3 Обучение с подкреплением для CPG-регулятора робота-змеи

Из анализа раздела 5.1 следует, что мы можем выполнить задачу перемещения робота-змеи с помощью модели осциллятора CPG, но мы не можем контролировать направление движения робота, поэтому нам нужно найти способ управлять направлением движения робота, регулируя параметры осциллятора.

Одна простая реализация аналогична описанной в разделе 5.2, т. е. отслеживание целевой точки. Однако, в отличие от метода, описанного в разделе 5.2, период изменения параметра осциллятора не должен быть слишком коротким, так как в этом случае сигнал будет меняться слишком быстро и не сможет эффективно направлять движение робота. Кроме того, метод, описанный в разделе 5.2, изменяет

только выход первого звена, что не позволяет полностью использовать потенциал многомерного пространства движений.

Поэтому мы будем менять амплитуду сигнала всех нейронов осциллятора каждые 1 секунду, что соответствует идее принятия дискретных состояний для формирования марковского процесса принятия решений в обучении с подкреплением. Таким образом, многомерная выходная нейронная сеть обучается с помощью метода обучения с подкреплением, чтобы робот-змея мог принимать оптимальное управляющее решение в различных состояниях, и этот метод позволяет достичь задачи движения робота к целевой точке.

Далее мы отдельно опишем идею и реализацию этого метода.

Пространство состояний

Входные данные нейронной сети - пространство состояний робота - должны содержать, по возможности, все значения состояний, которые могут повлиять на выбор действия.

Поскольку наша задача - обучить робота-змею двигаться к цели, все значения в пространстве состояний должны быть относительно точки цели. Преимущество такого подхода заключается в том, что при успешном обучении роботу не нужно учитывать абсолютные значения состояний в окружающей среде, что значительно снижает сложность обучения. Например, если скорость робота в окружающей среде известна, а его относительное положение относительно цели известно, потребуется некоторое обучение, чтобы робот узнал, хороша или плоха скорость в окружающей среде относительно цели... Кроме того, использование только относительных значений состояния также упрощает экспериментальный дизайн робота-змеи, мы можем позволить начальному состоянию робота быть фиксированным, и нам не нужно учитывать, влияет ли состояние робота в окружающей среде (например, начальный угол каждого звена в среде и начальное направление движения робота в среде) на выбор действия [17].

В момент t мы выбираем состояние $\mathbf{s} = [\phi, v_1, v_2, d, \boldsymbol{\theta}]$ как:

- $\phi(t) \in \mathbb{R}$ - Угол между вектором перемещения от центра масс робота к целевой точке $\vec{d}(t)$ и вектором скорости робота относительно окружающей среды $\vec{v}(t)$.
- $v_1(t) = |\vec{v}(t)| \cos \phi(t) \in \mathbb{R}$ - Скорость робота в желаемом направлении движения.

- $v_2(t) = |\vec{v}(t)| \sin \phi(t) \in \mathbb{R}$ - Скорость робота, направленная перпендикулярно $v_1(t)$, указывающая на разницу между направлением движения робота и желаемым направлением.
- $d(t) = |\vec{d}(t)| \in \mathbb{R}$ - Расстояние между центром масс робота и целевой точкой.
- $\boldsymbol{\theta}(t) \in \mathbb{R}^6$ - Вектор, состоящий из угла между первым звеном и $\vec{d}$ и угла между каждым звеном.

где:

$\vec{d}(t) = (x_2 - x_1, y_2 - y_1)$ - Разница между целевой точкой (x_2, y_2) и центром масс робота (x_1, y_1).

$\vec{v}(t) = (v_x, v_y)$ - Скорость робота в среде (v_x, v_y в направлениях Ox и Oy, соответственно), определяемая скоростью центра масс робота.

На рисунке 13 показана взаимосвязь между состояниями.

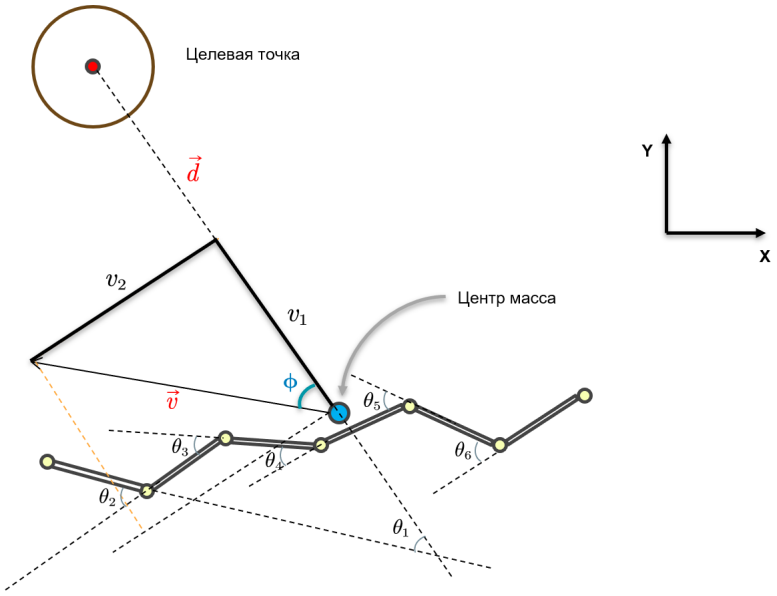


Рис. 13: Пространство состояний

Во время обучения иногда датчик скорости робота выдает очень большое значение "глюка потому что осциллятор используется для изменения амплитуды сигнала через регулярные промежутки времени, а не для управления роботом на каждом временном шаге, как в разделе 5.2, поэтому скорость сильно колеблется. Поэтому мы выбираем скользящее окно и берем среднее значение скорости за последние 0,1 секунды, $\hat{\vec{v}}(t)$ как скорость в момент времени t. Этот метод может эффективно га-

рантировать стабильность данных без потери точности, что благоприятно для последующего процесса обучения.

Поскольку мы рассматриваем угловые значения между звеньями, угловые скорости могут быть очень большой, если частота осциллятора выше в течение цикла длительностью 1 секунда, а это значит, что угловое значение $\boldsymbol{\theta}(t)$ в определенный момент времени не дает полного описания состояния робота в течение этого цикла.И роботу может быть трудно выучить оптимальную функцию значения состояния. Решение состоит в том, чтобы записывать состояние каждые 0,1 секунды в течение цикла. В конце цикла все данные о значении состояния используются как описание состояния в этом цикле. Таким образом, входной слой нейронной сети представляет собой 100-мерный вектор, содержащий 10 данных, записанных во время этого цикла [18].

Пространство действий

В разделе 5.1 мы проанализировали параметры, которые могут влиять на выход осциллятора. Мы сосредоточились на двух наборах параметров, u_i^e и u_i^f, которые могут непосредственно изменить направление движения робота.

Чтобы уменьшить размерность пространства действий, мы можем разложить эти два параметра с помощью следующего уравнения:

$$\begin{aligned} u_i^e &= u_m(\hat{u}_i + \tilde{u}_i) \\ u_i^f &= u_m(\hat{u}_i - \tilde{u}_i) \end{aligned} \tag{5.4.11}$$

где:

u_m управляет максимально допустимым значением выходного сигнала осциллятора.

$\hat{u}_i = 1$ - это постоянная величина, которая обеспечивает базовые кинематические возможности робота-змеи.

$\tilde{u}_i$ ограничен определенным интервалом, который представляет собой разницу между положительной и отрицательной амплитудами i-го нейрона осциллятора в данном цикле, и указывает на выбор направления i-го регулятора робота.

Преимущество этой конструкции в том, что она позволяет роботу не терять способность двигаться вперед за счет трения о землю, из-за резких изменений амплитуды выходного сигнала осциллятора, и в то же время приобретает способность менять направление и уменьшает размерность пространства действия. Поэтому мы

будем использовать эту конструкцию для преобразования регулирования параметров u_i^e и u_i^f в регулирование параметров $\tilde{u}_i$.

Преимущество такой конструкции заключается в том, что робот может поддерживать относительно стабильную скорость движения за счет непрерывного синусоидального движения и не теряет способность двигаться вперед за счет трения о землю из-за резкого изменения амплитуды выходного сигнала осциллятора, что приводит к потере скорости движения. Кроме того, робот приобретает способность менять направление движения и уменьшает размерность пространства движения. Поэтому мы будем использовать эту конструкцию для преобразования регулирования параметров u_i^e и u_i^f в регулирование параметров $\tilde{u}_i$.

Структура нейронных сетей

Мы представим структуру нейронной сети политики и нейронной сети ценности по отдельности

Если значение состояния робота на $(0.1 \cdot i)$-й секунде 1-секундного цикла равно $\mathbf{s}^i = [\phi^i, v_1^i, v_2^i, d^i, \boldsymbol{\theta}^i] \in \mathbb{R}^{10}$, тогда на входные слои нейронной сети политики и нейронной сети ценности поступают значения состояний $\mathbf{s} = [\mathbf{s}^1, \mathbf{s}^2, ..., \mathbf{s}^{10}] \in \mathbb{R}^{100}$

Выходной слой нейронной сети политики представляет собой 5-мерный сигнальный вектор, который после получения значений ограничивается $\tilde{u_i} \in [-0.1, 0.1]$.Выходом нейронной сети ценности является вещественное число, которое представляет собой оценку состояния.

Скрытый слой нейронной сети политики состоит из двух слоев, каждый из которых содержит 256 нейронов, и структурирован как полностью связанная сеть. Функция активации каждого слоя - функция $tanh$.

Структура скрытого слоя нейронной сети ценности такая же, как и нейронной сети политики, но функция активации каждого слоя - функция ReLu.

На рисунке 14 показана структура нейронной сети политики $\pi(\mathbf{s}; \boldsymbol{\theta})$ с нейронной сетью ценности $v(\mathbf{s}; \mathbf{w})$.

Функция награды

Хорошая функция награды эффективно направляет алгоритм обучения к целевому поведению. Она должна точно отражать цели задачи, гарантируя, что каждая выданная награда или наказание будут направлены на то, чтобы подтолкнуть алгоритм к желаемому результату.

В нашем решении мы задаем функцию награды в момент времени t:

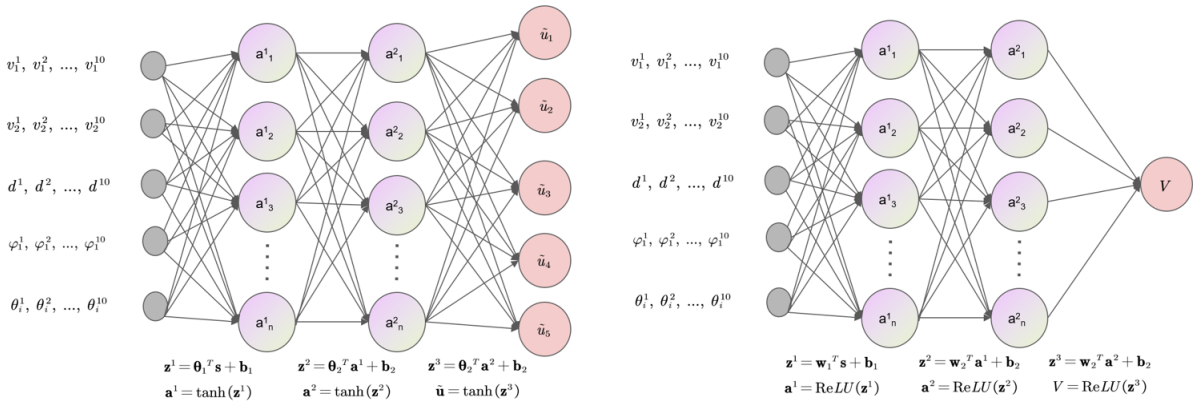


Рис. 14: Структура нейронной сети политики и ценности

$$r_t = (c_1 v_1 - c_2|v_2| + c_3|v|) \cdot e^{-c_4 \cdot d} + c_5 \cos\phi \cdot I(d < d_0) \tag{5.4.12}$$

где:

v_1- скорости робота в желаемом направлении движения, которая принадлежит вектору текущего состояния робота.

v_2 - скорость робота, направленная перпендикулярно v_1.

$|v|$ - скаляр скорости робота относительно окружающей среды.

d - расстояние между центром масс робота и целевой точкой.

ϕ - угол между вектором перемещения от центра масс робота к целевой точке и вектором скорости робота относительно окружающей среды.

$I(d < d_0)$ - функция индикатора, указывающая, находится ли центр масс робота в радиусе d_0 от целевой точки.

Функция вознаграждения, разработанная таким образом, является разумной и эффективной.

Во-первых, среда анализирует текущее состояние робота и поощряет действия, которые двигают робота к цели, и наказывает действия, которые двигают робота от цели, что достигается заданием v_1 и v_2. Во-вторых, добавляется вознаграждение за абсолютную скорость $|v|$, чтобы стимулировать робота двигаться как можно быстрее, а не оставаться в определенной позиции. Перед достижением цели награда зависит от расстояния d до цели, и чем ближе к цели, тем выше награда. Мы можем найти оптимальную функцию вознаграждения, регулируя четыре параметра c_1, c_2, c_3 и c_4.

Когда робот достигает целевой точки, он получает большое вознаграждение, которое задается индикаторной функцией $I(d < d_0)$ и параметром c_5. Это вознагражк-

дение связано с направлением движения робота при входе в целевую зону. Если робот попадает в целевую зону с направлением движения к целевой точке, он получает большое вознаграждение, и наоборот - малое вознаграждение. Эта идея реализуется с помощью $\cos\phi$.

Процесс обучения

Мы пошагово опишем процесс обучения робота-змеи на симуляторе MuJoCo.

1. Инициализация

Сначала мы завершим параметризацию модели робота-змеи, модели осциллятора, нейронных сетей и процесса обучения.

Для модели робота-змеи мы уже определили это в главе 3.2

Для модели осциллятора (5.4.1) мы задаем параметры, как показано в таблице 7.

Для функции награды мы задаем параметры, как показано в таблице 8.

Соответствующие параметры алгоритма обучения с подкреплением приведены в таблице 9.

Мы написали программу на языке python и использовали mujoco_py, python-интерфейс к физическому движку MuJoCo. Для построения и обучения нейронных сетей мы использовали библиотеку машинного обучения pytorch.

Для инициализации двух нейронных сетей мы используем инициализацию kaiming из библиотеки pytorch.

Таблица 7

a	2
b	10
τ_r	0.3125
τ_a	0.3125
w_{ji} $(j=i+1)$	$\pi/2$
w_{ji} $(j\neq i+1)$	0

Таблица 8

c_1	10
c_2	5
c_3	2
c_4	0.5
c_5	20

Таблица 9

β	2e-4
α	2e-4
γ	0.96
λ	0.95
ϵ	0.2
batch size	256

2. Выполнить сброс среды

В начале обучения мы задаем начальное состояние робота-змеи следующим образом: все звенья расположены на оси Ox, начальная скорость равна 0, а центр масс находится в точке $(-1.2, 0)$, как показано на рис. 15.

Центр масс не помещен в начало координат, потому что мы хотим, чтобы робот-змея достиг хорошей начальной скорости и начальной походки после периода запуска, в течение которого сигнал осциллятора не будет меняться. По окончании периода запуска центр масс робота сможет двигаться близко к началу координат и будет иметь начальную скорость в направлении оси x.

Причина установки времени запуска заключается в том, что начальная скорость робота-змеи равна 0, а угол между звеньями отсутствует, поэтому он еще не сформировал позу змеи, и такое особое значение состояния может легко привести к нестабильности параметров нейронной сети.

Мы установили время начала обучения равным 5 секундам. С 5-й секунды начинается процесс обучения, и следующая серия действий и состояний рассматривается как эпизод, пока среда не будет сброшена.

Перед началом каждого эпизода нам необходимо установить целевую точку. В этой задаче мы задали случайное появление целевой точки в области с центром в начале координат, радиусом от 1,5 до 2,5 м, центральной линией вдоль оси x и углом сектора 120 градусов, как показано на рисунке 16.

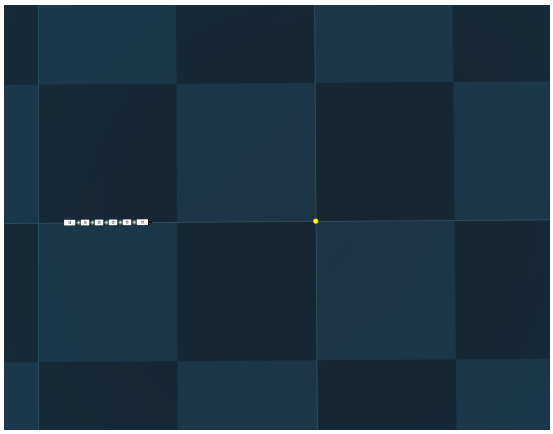


Рис. 15: Начальное состояние ($t = 0$)

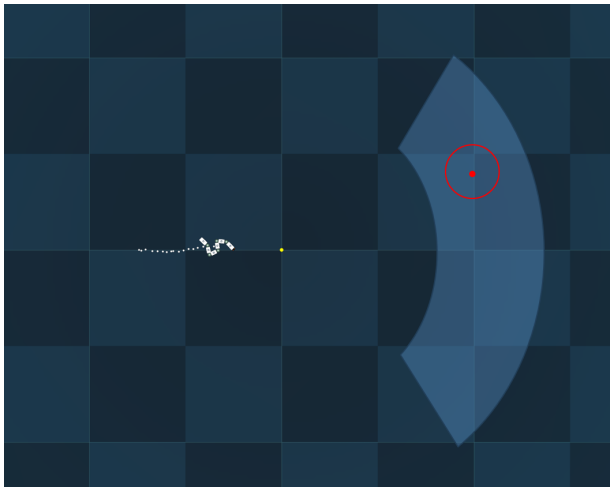


Рис. 16: Состояние после запуска ($t = 5$)

3. Выбор действий

Для задачи многомерным непрерывным пространством действий стратегическая нейронная сеть обычно выдает вектор средних $\boldsymbol{\mu}$ и вектор стандартных отклонений $\boldsymbol{\sigma}$, соответствующих размерностям действий. Эти выходные параметры определяют многомерное гауссовское распределение $\mathcal{N}(\boldsymbol{\mu}, \mathrm{diag}(\boldsymbol{\sigma}^2))$, где $\mathrm{diag}(\boldsymbol{\sigma}^2)$ обозначает вектор значений, начинающийся с $\boldsymbol{\sigma}^2$ в качестве диагональных элементов, что

указывает на то, что каждое измерение действия является независимым и имеет различные дисперсии.

В процессе симуляции мы будем собирать и сохранять состояние робота несколько раз в соответствии с заданным периодом времени. В конце цикла текущее время устанавливается равным t, и вектор состояния $\mathbf{s}_t$ в это время вводится в нейронную сеть политики. Нейронная сеть выдает вектор среднего $\boldsymbol{\mu}_t$ и вектор стандартного отклонения $\boldsymbol{\sigma}_t$ той же размерности, что и пространство действий, описывающие распределение вероятностей действий в данном состоянии.

Чтобы определить выбор действия в данный момент, вектор действий $\mathbf{a}_t$ выбирается из гауссова распределения $\mathcal{N}(\boldsymbol{\mu}_t, \mathrm{diag}(\boldsymbol{\sigma}_t^2))$ выхода сети. Этот процесс выбора может быть выражен как:

$$\tilde{\mathbf{u}}_t \sim \mathcal{N}(\boldsymbol{\mu}_t, \mathrm{diag}(\boldsymbol{\sigma}_t^2)) \tag{5.4.13}$$

Выбранное действие $\tilde{\mathbf{u}}_t$ изменит амплитуду осциллятора, и робот обновит состояние своего собственного осциллятора в соответствии с этим действием, изменит свое собственное состояние и будет взаимодействовать со средой, получая от нее обратную связь, как описано ниже.

Если значение действий выходит за границы, они обрезается.

На рисунке 17 представлен процесс, в ходе которого агент (робот) принимает решение и действия на основе состояния предыдущего периода.

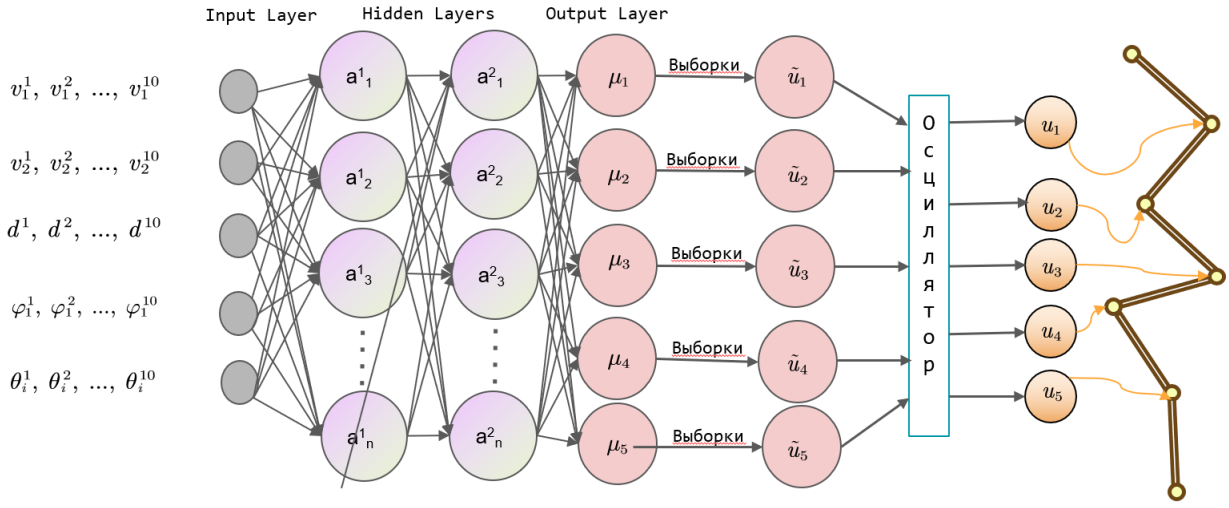


Рис. 17: Схема процесса решения и действия

4. Выполнение действий и взаимодействие со средой

Действия $\tilde{\mathbf{u}}_t$ изменяют амплитуду выхода каждого нейрона осциллятора в следующем цикле, т.е. максимальное и минимальное значения выхода. Робот получает новое управление благодаря новой динамике осциллятора и реализует эти управления на симуляторе MuJoCo.

В конце нового периода генерируется фрагмент данных, включающий:

–старые состояния предыдущего периода,

–новые состояния этого периода,

–выбранные действия,

–награду, полученную в соответствии с новым состоянием с помощью функции награды,

–вероятность того, что такие действия будут выбрано,

–индикаторную функцию, закончился ли эпизод или нет.

Этот фрагмент данных будет помещен в буфер.

5. Постоянный сбор данных

Пока эпизод не прерван, выполните шаги 3 и 4.

6. Условия прекращения эпизода

Существует 3 возможности прекращения эпизода.

1. Центр масс робота достигает целевой области.

2. Робот отклоняется от целевого направления. В этой задаче, если значение состояния $v_1 < 0$ в конце последних трех периодов, то считается, что робот отклонился от целевого направления.

3. Достигнут максимальный шаг обучения эпизода.

Как только эти ситуации возникают, эпизод заканчивается, среда сбрасывается, создается новый эпизод, и шаги 3 и 4 продолжаются, и так далее.

7. Обновления в сети

Если буфер данных заполнен, нейронные сети политики и нейронные сети ценности обновляются в соответствии с алгоритмом, описанным в 5.4.2.

8. Окончание обучения

Мы поставили условие, что обучение проходит успешно, если робот достигает целевой точки в 80 из последних 100 эпизодов.

5.4.4 Анализ результатов

После примерно 800 эпизодов и около 80 обновлений нейронной сети мы получили хороший результат, используя алгоритм PPO на симуляторе MuJoCo в соответствии с вышеуказанной процессом обучения и параметрами.

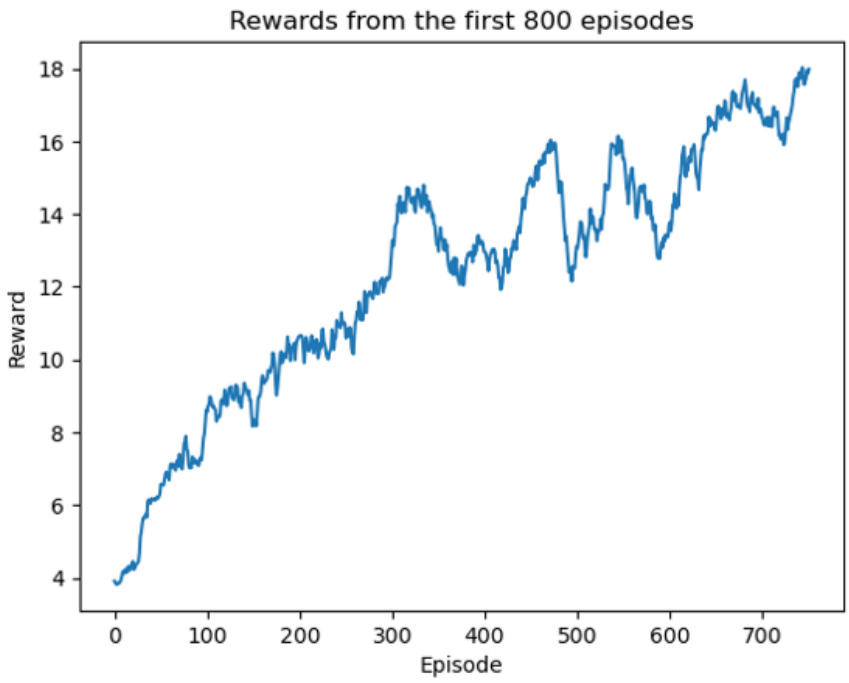


Рис. 18: Значения награды каждого эпизода

На рисунке 18 показаны значения награды, полученные для каждого эпизода в процессе обучения (оптимизация с помощью скользящего окна). Наблюдается четкая тенденция к росту значений награды по мере обучения. Это демонстрирует эффективность сочетания осциллятора с нейронной сетью с политикой, обученной с помощью обучения с подкреплением, для достижения движения робота к целевой точке.

Для демонстрации результатов мы проводим эксперимент. В этом эксперименте заранее задаются пять целевых точек:

$$(1.0, -1.0), (1.5, 1.0), (3.0, 1.0), (4.0, 2.0), (5.0, 0.0)$$

и робот последовательно движется к ним. Если робот достигает целевой зоны (круговой зоны с центром в целевой точке радиусом $0,3$ м), целевая точка меняется.

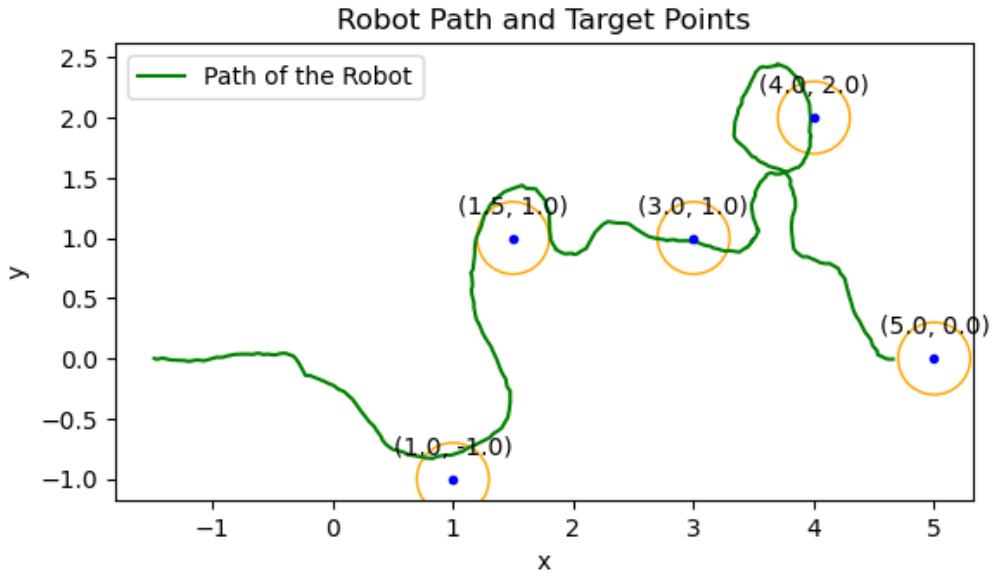


Рис. 19: Результат движения к целевой точке (1)

На рисунках 19, 20 и 21 показано движение робота-змеи к целевой точке. На рисунке 22 показано скалярное значение скорости робота во время его движения. Видно, что робот отлично справляется с задачей слежения за целевыми точками, и может поддерживать определенную скорость. Однако остается проблема, связанная с тем, что робот не может контролировать угол движения в окружающей среде, когда он достигает целевых точек. Это требует дальнейшего изучения.

6 Заключение

В этой статье мы создали физическую модель робота-змеи и продемонстрировали, что робот-змея может выполнять змееподобные извилистые движения в условиях трения с помощью метода управления виртуальными ограничениями, а также может управлять направлением движения робота, контролируя угол наклона головы.

Кроме того, мы исследуем два метода управления для неизвестных сред.

Во-первых, мы используем генетический алгоритм для поиска оптимальных параметров движения змеи по прямой линии, чередуя быстрое и медленное управление. Этот метод не основан на модели трения и имеет хороший потенциал для применения.

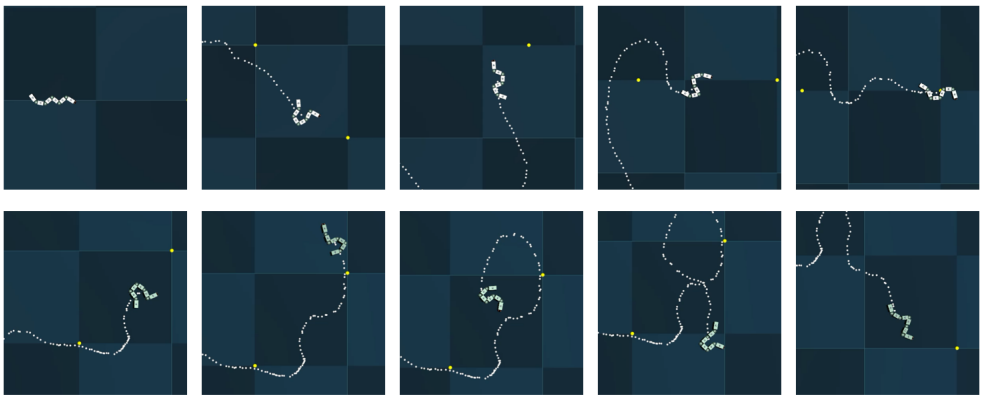


Рис. 20: Результат движения к целевой точке (2)

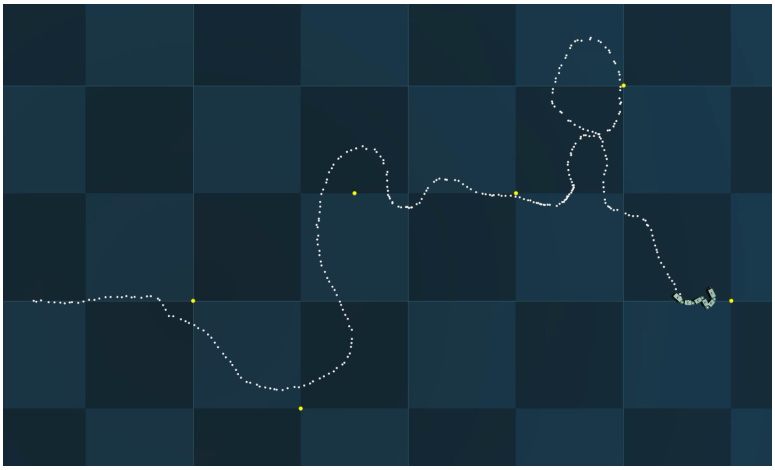


Рис. 21: Результат движения к целевой точке (3)

Во-вторых, осциллятор используется для моделирования периодического выходного сигнала робота-змеи, а алгоритм обучения с подкреплением - для обучения робота выбирать сигналы осциллятора в различных состояниях, чтобы изменить направление движения робота и выполнить задачу движения к целевой точке.

Приведенные выше исследования показывают, что эти методы эффективны для решения сложной задачи управления движением робота-змеи.

Список литературы

[1] *Owen T. "Biologically Inspired Robots: Snake-Like Locomotors and Manipulators"by Shigeo Hirose Oxford University Press, Oxford, 1993, 220pages, incl. index (£40). Robotica. 1994;12(3):282-282.*

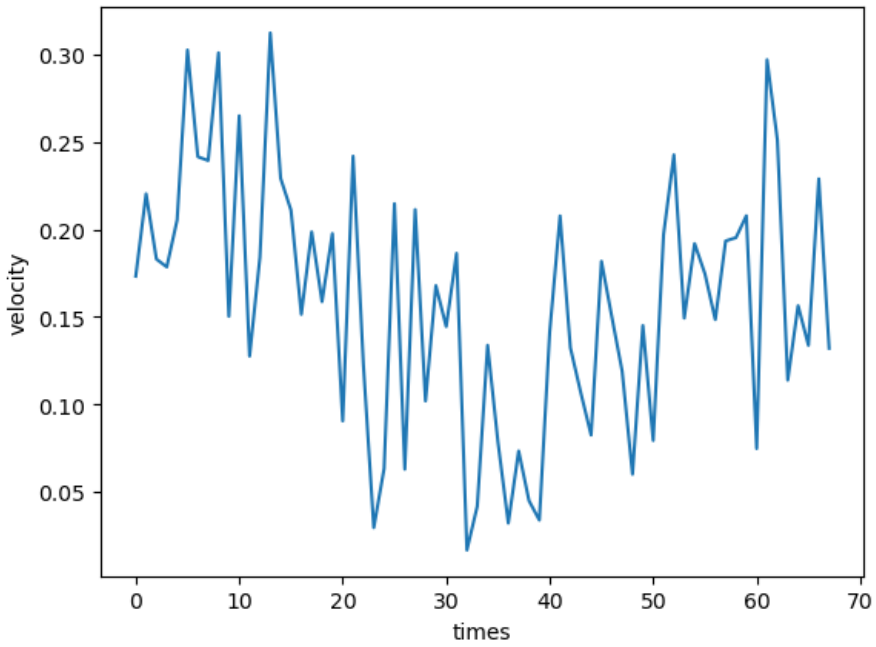


Рис. 22: Скалярное значение скорости

[2] *Liu,J., Tong,Y., Liu,J. "Review of snake robots in constrained environments."Robotics and Autonomous Systems, 141, 103785 (2021).*

[3] *M. Tanaka and K. Tanaka, "Control of a Snake Robot for Ascending and Descending Steps,"in IEEE Transactions on Robotics, vol. 31, no. 2, pp. 511-520, April 2015*

[4] *P.Liljeback, K.Y.Pettersen, Ø.Stavdahl and J. T. Gravdahl, "Snake robots: modelling, mechatronics, and control", Springer-Verlag, London, 2013.*

[5] *Schulman, J., Wolski, F., Dhariwal, P., Radford, A., Klimov, O. Proximal Policy Optimization Algorithms. arXiv preprint arXiv:1707.06347 (2017).*

[6] *E. Todorov, T. Erez, and Y. Tassa, "MuJoCo: A physics engine for model-based control," in Proc. IEEE/RSJ Int. Conf. Int. Robots Syst., 2012, pp. 5026–5033*

[7] *P. Liljebäck, K. Y. Pettersen, Ø. Stavdahl and J. T. Gravdahl, "A simplified model of planar snake robot locomotion,"2010 IEEE/RSJ International Conference on Intelligent Robots and Systems, Taipei, Taiwan, 2010, pp. 2868-2875*

[8] *Z. Bing, C. Lemke. "Energy-efficient and damage-recovery slithering gait design for a snake-like robot based on reinforcement learning and inverse reinforcement learning" // Neural Netw, 2020 Sep:129:323-333.*

[9] *Rezapour, E., Pettersen, K.Y., Liljebäck, P. et al. Path following control of planar snake robots using virtual holonomic constraints: theory and experiments. Robot. Biomim. 1, 3 (2014).*

[10] *J. Mukherjee, I. N. Kar and S. Mukherjee, "Adaptive sliding mode control for head-angle and velocity tracking of planar snake robot,"2017 11th Asian Control Conference (ASCC), Gold Coast, QLD, Australia, 2017, pp. 537-542*

[11] *A. Mohammadi, E. Rezapour, M. Maggiore and K. Y. Pettersen, "Maneuvering Control of Planar Snake Robots Using Virtual Holonomic Constraints,"in IEEE Transactions on Control Systems Technology, vol. 24, no. 3, pp. 884-899, May 2016*

[12] *Черноусько Ф. Л., Болотник Н. Н. "Динамика мобильных систем с управляемой конфигурацией", 2022.*

[13] *Mitchell, M. (1996). "An introduction to genetic algorithms."The MIT Press.*

[14] *Xiaodong Wu, Shugen Ma, "CPG-based control of serpentine locomotion of a snake-like robot Mechatronics, Volume 20, Issue 2, 2010, Pages 326-334*

[15] *Matsuoka, K. Mechanisms of frequency and pattern control in the neural rhythm generators. Biol. Cybernetics 56, 345–353 (1987).*

[16] *Richard S. Sutton, David McAllester, Satinder Singh, and Yishay Mansour. 1999. Policy gradient methods for reinforcement learning with function approximation. In Proceedings of the 12th International Conference on Neural Information Processing Systems (NIPS'99). MIT Press, Cambridge, MA, USA, 1057–1063.*

[17] *X. Liu, R. Gasoto, Z. Jiang, C. Onal and J. Fu, "Learning to Locomote with Artificial Neural-Network and CPG-based Control in a Soft Snake Robot,"2020 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS), Las Vegas, NV, USA, 2020, pp. 7758-7765*

[18] *Jiayu Wang, Chuxiong Hu, and Yu Zhu, "CPG-Based Hierarchical Locomotion Control for Modular Quadrupedal Robots Using Deep Reinforcement Learning"IEEE Robotics and Automation Letters (Volume: 6, Issue: 4, October 2021)*