# Ассимиляция данных процесса культивирования под модель микробного сообщества

Автор: Станислав Новиков
email: stanislav.novikov@gmail.com
Код исследования доступен по ссылке: https://github.com/Run4rest/Data-assimilation

## Краткое резюме работы

Ассимиляции данных для мат.модели микробного сообщества выполнена 3-мя разными способами на среднем ноутбуке CPU - AMD Ryzen 5 4600U, 16.0 GB RAM . Хорошие результаты показал подход MVN (см.ниже). RL SAC демонстрирует сходимость, но видимо требует большей вычислительной мощности

## Постановка задачи

Предположим что процесс культивирования сообщества микроорганизмов описывается системой уравнений (1). У нас есть серия измерений $X$ и $S$ в нескольких контрольных точках. Необходимо подобрать параметры так, чтобы отклонения расчетных значений $X$, $S$ из (1) и измерений $X$, $S$ в контрольных точках были минимальны. Часть параметров мы считаем константами в течении всего процесса. Часть параметров может изменяться. Однако в рамках интервала между соседними контрольными точками все параметры неизменны.
Цели исследования:

- проверка алгоритма RL SAC для подбора параметров (1)
- проверка других алгоритмов ассимиляции данных

$$\frac{dS}{dt} = \left(S_0 - S\right)D - \gamma\mu_X X$$

$$\frac{dX}{dt} = \left(\mu_X - D\right)X$$

$$\frac{dY}{dt} = \left(\mu_Y - D\right)Y$$

$$\frac{dA}{dt} = v_A X - DA - q_A Y$$

$$\frac{dB}{dt} = v_B X - DB - q_B Y \qquad \textbf{(1)}$$

$$q_A = g_A \frac{1}{1+\frac{A}{K_A}+\frac{B}{K_B}} \frac{\mu_{Amax} A}{K_A}$$

$$q_B = g_B \frac{1}{1+\frac{A}{K_A}+\frac{B}{K_B}} \frac{\mu_{Bmax} B}{K_B}$$

$$\mu_Y = \frac{1}{1+\frac{A}{K_A}+\frac{B}{K_B}} \left( \frac{\mu_{Amax} A}{K_A} + \frac{\mu_{Bmax} B}{K_B} \right)$$

$$\mu_X = \mu_0 \frac{I_A}{I_A + A} \frac{I_B}{I_B + B} \frac{S}{K_s + S}$$

Пусть параметры точного решения имеют следующие значения:
$S_0 = 100; \gamma = 1; v_A, = 0.0001; v_B = 0.009, g_A = 0.3; g_B = 3.0; \mu_0 = 0.3; \mu_{A\,max} = 0.1; \mu_{B\,max} = 0.7$
$K_A = 1.0; K_B = 0.1; I_A = 100.0; I_B = 1.0$

Эти значения соответствуют сообществу в котором основная культура $X$ потребляет субстрат $S$ и выделяет продукты метаболизма $A$ и $B$. Оба метаболита ингибируют рост $X$. Метаболит $B$ ингибирует рост $X$ много больше чем $A$. Усредненный спутник $Y$ “подъедает” $A$ и $B$, но быстрее растет на $B$.

Рассчитаем количество уравнений определяющее однозначность решения. Если мы требуем в контрольных точках:
1. равенство расчетных и заданных $X$, $S$
2. равенство $Y$, $A$ , $B$
, то кол-во уравнений: $5N - 6$, где $N$ - число контрольных точек. И мы можем рассчитывать на однозначное решение если кол-во параметров которые могут меняться от точки к точке: $n < \frac{5N-6}{N-1}$ при условии что все другие параметры известны.
Либо $n < \frac{5N-6-n_C}{N-1}$ , где $n_C$- кол-во неизвестных параметров-констант.

# Описание алгоритма RL SAC

По результатам статьи [1] для подбора параметров системы (1) выбран алгоритм Reinforcement Learning Soft Actor Critic (RL SAC). Задачи обучения с подкреплением часто формулируются как марковские процессы принятия решений (MDP), которые можно определить кортежем $(S, A, P, r)$ [3]. В нашем случае пространство действий $A$ и пространство состояний $S$ непрерывны и могут быть ограничены. На временном шаге t среда находится в состоянии $s_t \in S$ и агент выполняет действие $a_t \in A$, чтобы среда перешла в новое состояние $s_{t+1} \in S$. Переход между состояниями при выполнении действия в общем случае является случайным с вероятностью перехода $P(s_t, a_t, s_{t+1})$. Агент получает вознаграждение $r_t$ из распределения вознаграждения $r$ на каждом временном шаге $t$. Обычно цель состоит в том, чтобы найти политику (правило) $\pi: S \to A$: которая максимизирует ожидаемое дисконтированное вознаграждение $Q(s_t, a_t) = E\left[\sum_{t'=t}^{\infty} \gamma r_{t'}(s_{t'}, a_{t'})\right]$, где $0 < \gamma \leq 1$ - коэффициент дисконтирования, $E$ - усреднение.
В нашем случае каждое измерение пространства действий $A$ соответствует одному параметру системы (1), т.е. точка в $A$ это вектор параметров в (1). Пространство состояний $S$ также содержит вектор параметров и одно дополнительное измерение: шаг $t$ (время контрольной точки/номер шага). $r_t$ определяется ошибкой между расчетными значениями и значениями заданными в контрольной точке. При заданном

$a_t$ среда (1) строго переходит в единственное возможное состояние $s_{t+1}$, однозначно заданное координатами вектора $a_t$.(набором параметров). В нашем случае кол-во шагов ограничено. Есть формальное начальное состояние $s_0$ и траектория $s_1$,..., $s_N$, где $N$ - число контрольных точек. $s_0$ единственное общее состояние для всех траекторий.
Для ограниченного и разумного $N$ дисконтирование для расчета совокупного вознаграждения не имеет смысла, т.е. $\gamma = 1$
SAC - популярный алгоритм обучения с подкреплением типа Q-learning [2]. Алгоритм использует две нейросети. Одна нейросеть аппроксимирует зависимость совокупного вознаграждения $Q(s_t, a_t)$, вторая формирует $\mu(s_t)$ - среднее значение - и $\sigma(s_t)$- дисперсию - нормального распределения которое и является политикой $\pi(s_t)$ оптимальную политику $\pi(s_t)$. Первая сетка тренируется так чтобы сократить ошибку $E(Q(s_t, a_t) - [r_t + \gamma E(Q(s_{t+1}, a_{t+1}) + \alpha H(\pi))])^2$, где $H(\pi)$ - энтропия политики, $\alpha$ - константа. Ошибка второй пропорциональна: $-E(Q(s_t, a_t) - \alpha log(\pi(s_t)))$. Если предположить что $\alpha = 0.$, то ошибка пропорциональна среднему значению $Q$ со знаком минус. Т. е вторая сетка тренируется так чтобы получить максимальный средний $Q$ за минусом среднего значения $h = \alpha log(\pi(s_t))$. Слагаемое $h$ “снижает” фокус на максимизацию $Q$ и позволяет агенту - $\pi(s_t)$ - исследовать не самые оптимальные на текущий момент варианты.

## Гипер-параметры

Помимо стандартных гиперпараметров таких как learning rate и т.д. алгоритм обладает специфическими:
target smoothing coefficient ($\tau$ ) - коэффициент сглаживания экспоненциального усреднения параметров сетки $Q(s_t, a_t)$. Авторы алгоритма (см. [2]) рассчитывают ошибку не для непосредственной сетки $Q(s_t, a_t)$, а для вспомогательной сетки $Q_{target}$. Параметры $Q_{target}$ рассчитываются как экспоненциальное среднее параметров $Q$ после каждой тренировки $Q$ .
target update frequency - число действий ($a_t$)/шагов после которого выполняется очередная тренировка сеток
В выложенной на github версии RL SAC я использовал параметры, предложенные создателями алгоритма.

### Задача 1

Из множества параметров $D,\ S_0, \gamma,\ v_A,\ v_B, g_A,\ g_B, \mu_0,\ \mu_{A\,max}, \mu_{B\,max}, K_S, K_A, K_B, I_A, I_B$ и начальных условий $Y_0, A_0,\ B_0$ необходимо найти 2 параметра: $g_B, I_A$ , два связанных параметра: $v_B = 0.003 g_B,\ I_A = 100 I_B$ и 3 начальных условия: $Y_0$ , $A_0$ , $B_0$ . Известно, что искомые значения лежат в интервалах:
$g_B \in [1, 10]$ , $I_A \in [0.0001, 1000],\ Y_0 \in [0., 2.0],\ A_0 \in [0., 0.5],\ B_0 \in [0., 0.5]$
Точные значения: $g_B = 3.0, I_A = 100$ .

Были опробованы два варианта реализации. В обоих случая $a_{t'} = (g_B, I_A, Y_0, A_0, B_0)$
Но в первом варианте $Y_0, A_0, B_0$ используются в расчете $Q$ только на первом шаге. Для последующих шагов значения $Y_0, A_0, B_0$, сгенерированные агентом, замещаются финальными значениями предыдущего шага.
Во втором варианте с помощью замены переменных: $Y = Y' + Y_0$ ,$A = A' + A_0$ , $B = B' + B_0$ . Тогда начальные условия для $Y'$,$A'$,$B'$, будут нулевыми, а $Y_0$ ,$A_0$ ,$B_0$ в (1) становятся параметрами, которые могут меняться в контрольных точках. Второй вариант более четко реализует идею RL SAC однако имеет недостаток - при формировании $a_{t'}$на любом шаге кроме первого значения $Y, A, B$ могут получить отрицательные значения, что ведет к сбою расчета (1). Финальные значения $Y'$,$A'$,$B'$ не формируются и алгоритм вынужден завершать траекторию/эпизод. Обучение RL SAC на коротких траекториях занимает значимо больше времени чем в первом варианте.

Рассчитанные значения первого варианта: $g_B = 3.261, I_A = 383.8$ . Значения получены после работы алгоритма на ноутбуке в течении суток.
На Рис 1 показаны точное решение (1) и результаты RL SAC

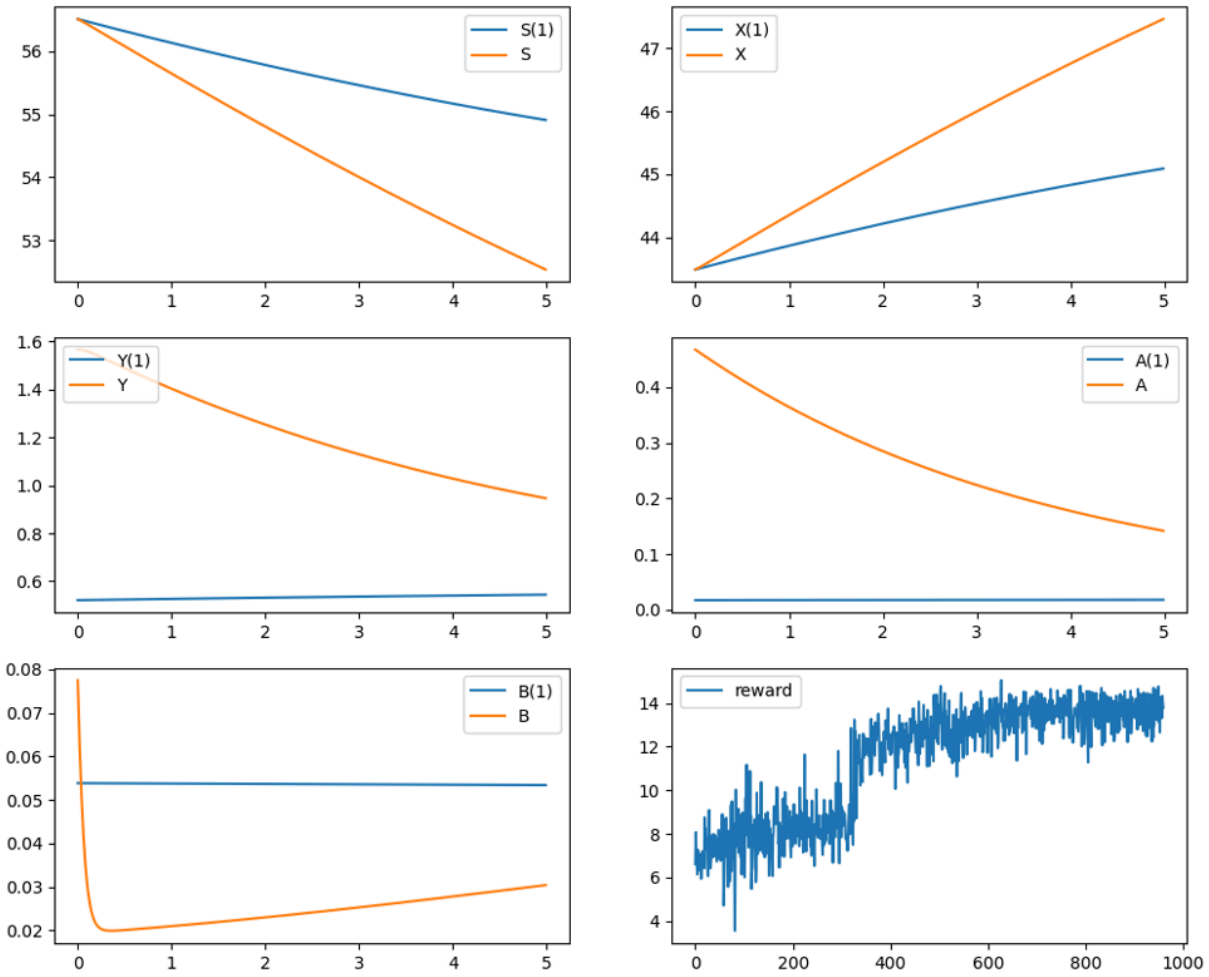


Рис. 1 Точное решение и результаты RL SAC.

Контрольные точки отобраны через каждые 15 мин. (0.25), период серии измерений 5ч (5.0), количество измерений - 19. Кол-во контрольных точек - 20. Начальная контрольная точка $s_0$ - абстрактная, общая для всех расчетных траекторий.

График внизу справа показывает рост Q с увеличением количества расчетных траекторий. Траектория это одно прохождение алгоритма по всем контрольным точкам. Максимально возможный Q ~ 20.

# Проверка других алгоритмов

Если упростить постановку и рассматривать задачу как одношаговую (параметры не меняются, либо их изменение можно аппроксимировать подбором параметров заданной функции и расчет выполняется сразу для всего интервала), то можно использовать :

1. градиентный подъем к максимуму $Q$
2. политику с многомерным нормальным распределением (MND)
3. комбинацию первого и второго подхода

В проведенных экспериментах градиентный спуск/подъем работал только на начальных этапах поиска оптимума при малых значениях $Q$. С ростом $Q$ скорость спуска снижается до полной остановки. Безусловно все надо перепроверять, тем не менее сейчас я думаю что градиент "слишком локален". Т.е. траектория оптимального спуска  далека от прямой. Точно "попасть"  на эту траекторию маловероятно. Градиент в окрестности этой траектории ведет к самой траектории (желобу), а при приближении изменение искомых параметров либо приводит к переходу на противоположный склон, либо к остановке в седловой точке.

Основные идеи MND алгоритма:

- Политика это многомерное нормальное распределение по параметрам.
- Текущее среднее значение политики находится в точке с максимальным $Q$, известной на текущий момент
- Ковариационная матрица политики рассчитывается по N точкам с наибольшим $Q$, выбранным случайным образом в окрестности среднего.

Данный алгоритм показал высокую точность и скорость работы как для задач поиска значений неизвестных постоянных параметров так и для задачи поиска функции изменения протока.
Как выяснилось по ходу проведения расчетов поставленная задача плохо обусловлена, т.е. высокую точность соответствия экспериментальным точкам можно получить для большого числа различных значений параметров. Может быть точное решение одно, однако, как показано ниже, "альтернативные" варианты обеспечивают reward 99.8+ и с практической точки зрения являются самостоятельными решениями. Большое множество "альтернатив" имеют некорректные начальные значения величин $Y, A, B$, которые в силу высокой (относительно выбранного временного шага) скорости роста $Y$ быстро корректируются. Такие варианты решения можно отсечь, без потери общности, штрафом на высокие значения вторых производных величин $A, B$

## Задача 2

На Рис.3 приведены графики системы (1) и графики по параметрам, полученным MND, в Задаче 1 с увеличенным периодом - 40ч. Вычислительная эффективность алгоритма позволила увеличить период.

Точные значения: $g_B = 3.0, I_A = 100$ .

Рассчитанные значения:

$g_B = 3.39,\ I_A = 100.02$ после 2.5 часов работы (параметр SAMPLES = 1000, см.код)

$g_B = 2.985, I_A = 100.002$ после 12 часов работы (параметр SAMPLES = 10000)

Графики $S/S(1)$ и $X/X(1)$ не различимы в данном масштабе поэтому на рисунке видны только $S$ и $X$

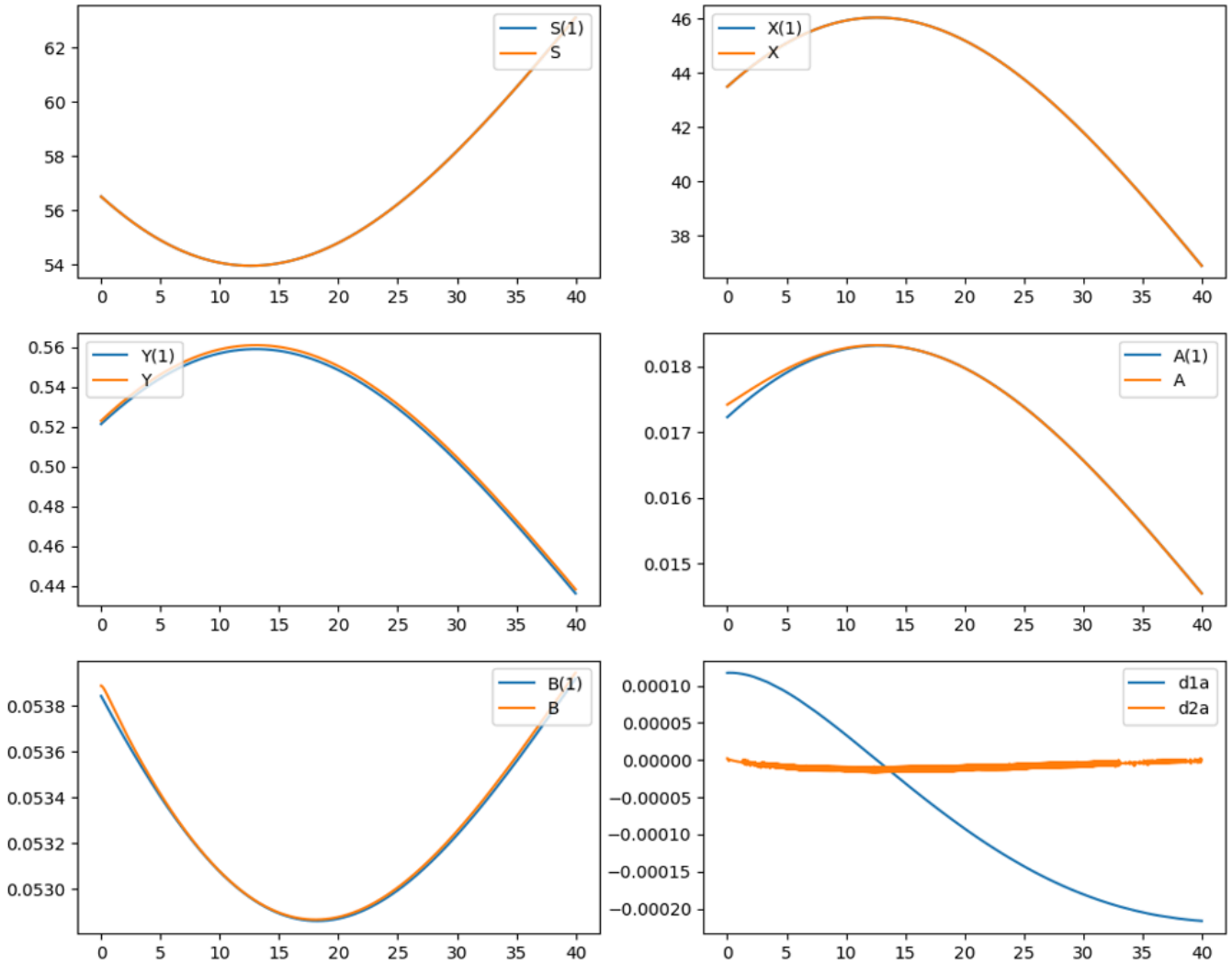


Рис.3. Точное решение и результаты MND.

В нижнем правом углу график первой и второй производных переменной A

## Задача 3

Задача 3 представлена как пример плохой обусловленности. На Рис.4 приведены графики системы (1) и графики по параметрам, полученным MND алгоритмом.

Алгоритм искал 2 параметра: $g_B, I_A$ (+ 1 связанный $I_A = 100I_B$) и 3 начальных условия:

$Y_0$ , $A_0$ , $B_0$ , в диапазонах:

$g_B \in [1, 10]$ , $I_A \in [0.0001, 1000]$, $Y_0 \in [0., 2.0]$, $A_0 \in [0., 0.5]$, $B_0 \in [0., 0.5]$

Точные значения: $g_B = 3.0, I_A = 100$ . Рассчитанные: $g_B = 5.18, I_A = 100.02$

При практически полном совпадении $S/S(1)$, $X/X(1)$, $B/B(1)$ алгоритм занижает $Y$ и завышает $A$. Т.е. в полученном альтернативном варианте $Y$ лучше кушает $B$ поэтому при том же $B$ требуется меньше $Y$. Меньший $Y$ съедает меньше $A$, т.е. $A$ выше. Графики $S/S(1)$ и $X/X(1)$ не различимы в данном масштабе поэтому на рисунке видны только $S$ и $X$.

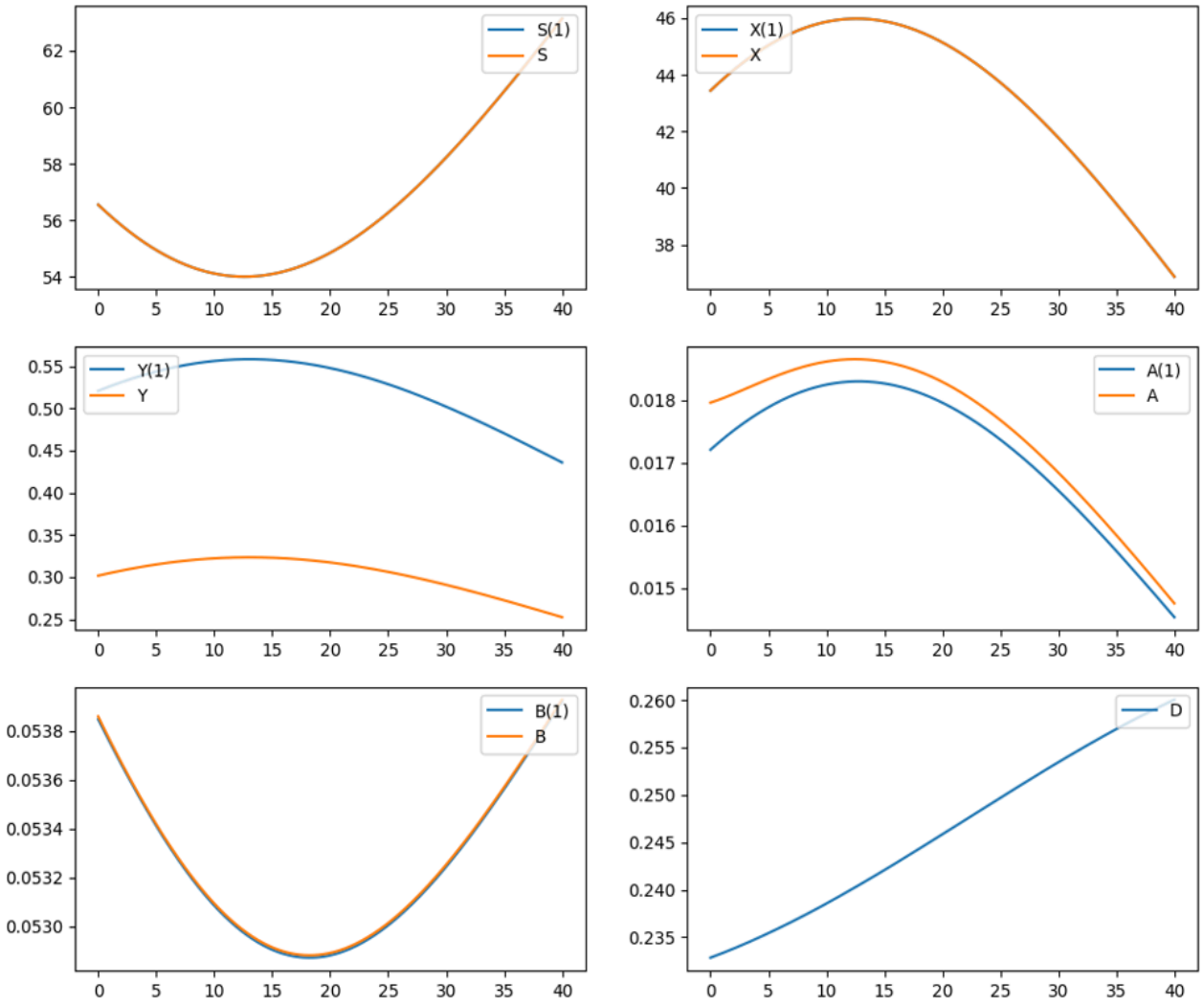


Рис.4

## Задача 4

Необходимо найти как менялся проток при заданных параметрах и неизвестных начальных условиях $Y_0$ , $A_0$ , $B_0$ . Диапазон поиска протока $D \in [0.1., 0.4]$, начальных условий $Y_0 \in [0., 2.0]$, $A_0 \in [0., 0.5]$, $B_0 \in [0., 0.5]$

Неизвестная функция изменения протока аппроксимирована 2-мя гиперболами - по гиперболе на каждую половину интервала с условием равенства протока и первой производной в середине интервала.
Практически все точные графики неотличимы от расчетных.

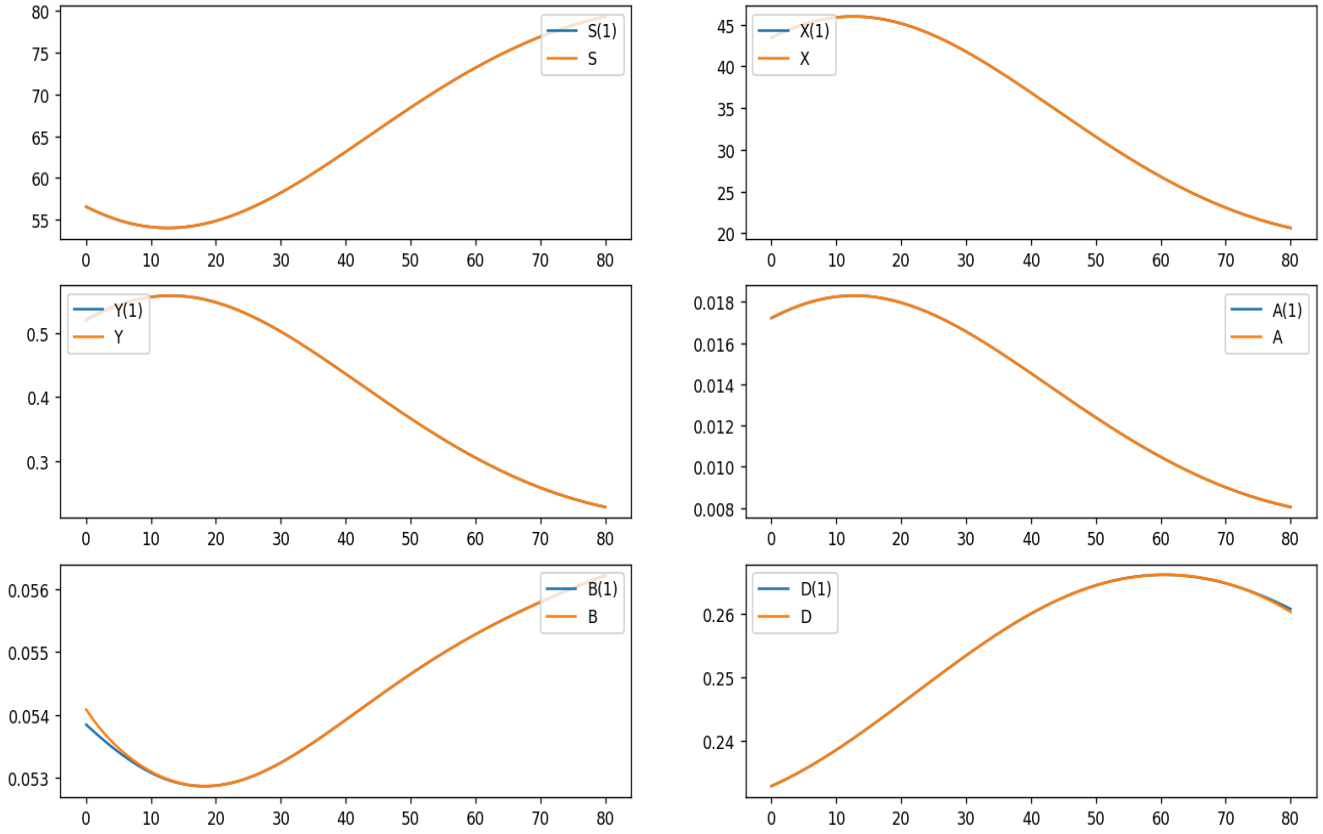


Рис.5

## Задача 5

Необходимо найти как менялся проток при заданных параметрах и неизвестных начальных условиях $Y_0$ , $A_0$ , $B_0$ . Диапазон поиска протока $D \in [0.1., 0.4]$, начальных условий $Y_0 \in [0., 2.0]$, $A_0 \in [0., 0.5]$, $B_0 \in [0., 0.5]$

Неизвестная функция изменения протока аппроксимирована параболой.

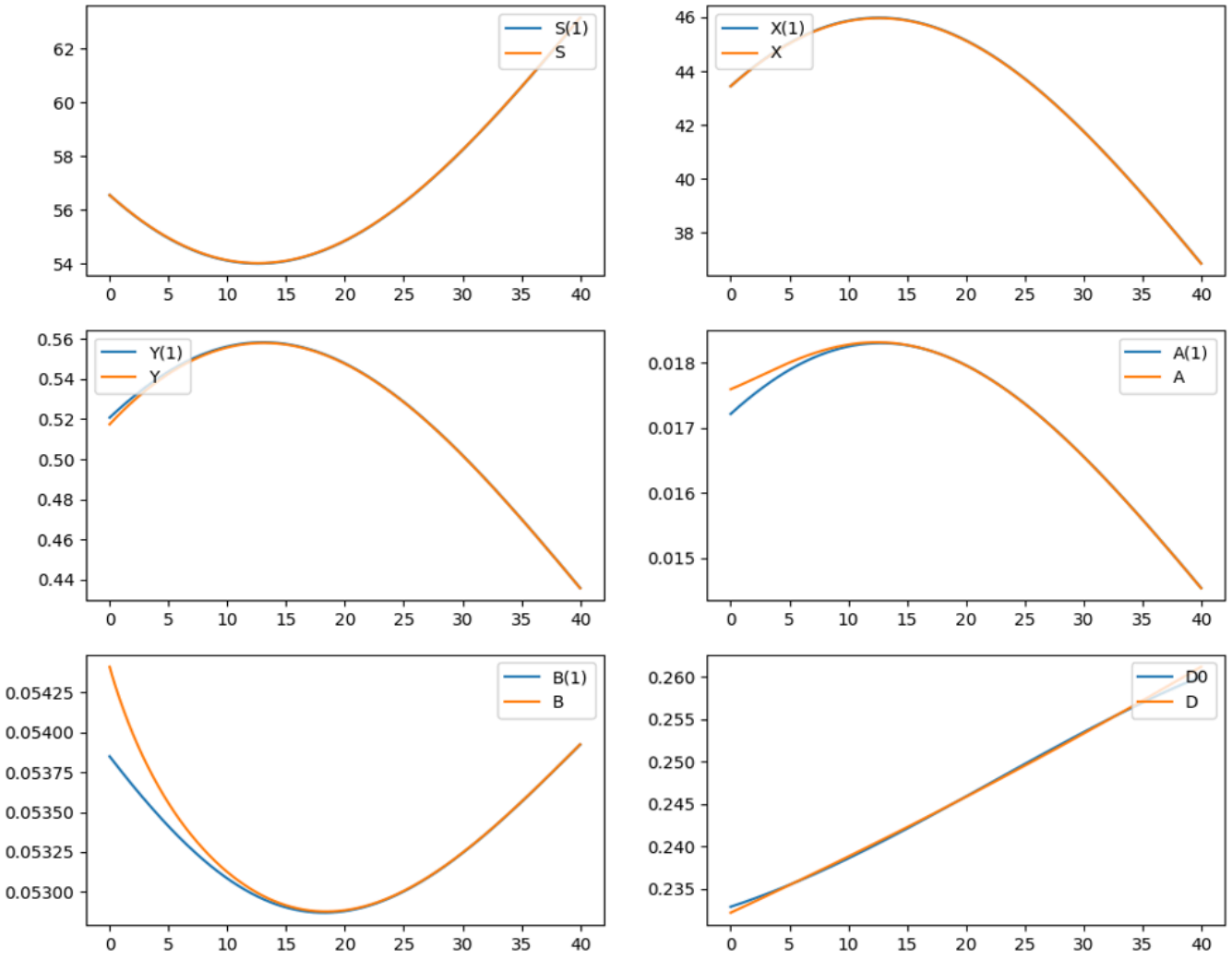


Рис.6

## Полное описание MND алгоритма

1. Выполняется поиск начальной точки с ненулевым $Q$ (reward'ом). Пробные точки генерируются случайным образом с равномерным распределением по пространству параметров
2. В окрестности точки с максимальным $Q$ выполняется поиск с большим $Q$. PDF на этом этапе равномерная.
3. В окрестности точки с максимальным $Q$ выполняется поиск точки с большим $Q$. PDF поиска - многомерное нормальное распределение по искомым параметрам. PDF строится следующим образом:
    - среднее значение помещается в точку с максимальным $\text{текущим } Q$
    - в окрестности среднего рассчитывается $Q$ для $N$ точек. Из рассчитанных $N$ выбирается $M$ с наибольшими $Q$. По выбранным точкам строится ковариационная матрица и генерируется $N$ новых точек. Если ни одна из точек не имеет $Q$ больше максимального, из $N$ новых и $M$ выбранных на предыдущем шаге отбираются $M$ с наибольшими $Q$. Снова строится

ковариационная матрица и отбираются $N$ точек. Так повторяем до тех пор пока не получим хотя бы одну точку с $Q$ больше текущего максимума.

Замечание 1: ковариационная матрица масштабируется в соответствии с выбранным размером окрестности.

Замечание 2: по ходу цикла можно менять радиус окрестности и $M$. В представленных экспериментах radius менялся вручную в диапазоне от 0.03 до 0.000006, $M$ от 10 до 1000. Вопрос подбора $M$ и радиуса остается открытым. $M = 10$ обеспечивает более сильные ограничения на допустимые изменения параметров и более сильные зависимости параметров друг от друга. И это хорошо поскольку сужает диапазон поиска. Но малое $M$ может привести к некорректному PDF которая не сможет обеспечить $Q$ выше чем использованные в расчете ковариационной матрицы и алгоритм зайдет в тупик. Большое $M$ снижает точность "прицеливания" поскольку в расчеты ковариационной матрицы попадают "максимумы" второго эшелона и долго замещаются большими значениями.

Что касается радиуса, в случае "желоба" с "треугольным профилем", уменьшение радиуса не позволит упростить поиск спуска, а лишь увеличит время поиска.

4. Если мы нашли точку с $Q$ больше предыдущего максимума, то строим вектор от предыдущего максимума к текущему и проверяем точки в ближайшей окрестности от нового максимума в направлении вектора. В текущей версии проверяются точки расположенные на 1/20, 2/20, 3/20, ... длины вектора.
5.

[1] Max R. Mowbray, Chufan Wu, Alexander W. Rogers. A reinforcement learning-based hybrid modeling framework for bioprocess kinetics identification, 2022 г., Biotechnology and Bioengineering, Wiley Periodicals LLC.

[2] T. Haarnoja, A. Zhou, P. Abbeel, and S. Levine. Soft actor-critic: Off-policy maximum entropy deep reinforcement learning with a stochastic actor. arXiv preprint arXiv:1801.01290, 2018.

[3] R. S. Sutton and A. G. Barto. Reinforcement learning: An introduction. MIT press, 2018.

# Приложение 1. Описание файлов на Github

| | |
|---|---|
| main_vF1.0.py | main Задачи 2 (MND алгоритм для определения 2-х параметров $g_B, I_A$ , 2-х связанных параметров $v_B = 0.003 g_B, I_A = 100 I_B$ и 3-х начальных условий $Y_0$ , $A_0$ , $B_0$ .) |
| environment_vF01.py | модуль содержит класс среды (система ODE, параметры, контрольные точки, расчет reward'а) для Задачи 2 |
| main_vF1.1.py | main Задачи 4 (MND алгоритм для определения функции изменения протока при заданных параметрах и |

| | |
|---|---|
| | неизвестных начальных условиях $Y_0$ , $A_0$ , $B_0$) |
| environment_vF11.py | модуль содержит класс среды (система ODE, параметры, контрольные точки, расчет Q) для Задачи 4 |
| replay_buffer_vF00.py | модуль содержит класс репозитория векторов параметров и reward'ов для Задач 2 и 4 |
| main_v.sac.py | main Задачи 1 (RL SAC алгоритм для определения 2-х параметров $g_B$, $I_A$ , 2-х связанных параметров $v_B = 0.003 g_B$, $I_A = 100 I_B$ и 3-х начальных условий $Y_0$ , $A_0$ , $B_0$ .) |
| environment_vsac1.py | модуль содержит класс среды первого варианта RL SAC Задачи 1 |
| environment_vsac2.py | модуль содержит класс среды второго варианта RL SAC Задачи 1 |
| sac_v1.py | модуль содержит классы первого варианта RL SAC алгоритма Задачи 1 |
| sac_v2.py | модуль содержит классы второго варианта RL SAC алгоритма Задачи 1 |
| replay_buffer_vsac | модуль содержит класс репозитория алгоритма RL SAC для Задач 1 |