Отчёт по лабораторной работе №4

«Единая математическая структура в разнородных системах»

Молькова А. Е.
Группа R4197

Аннотация

В отчёте показано, что три прикладные задачи — регуляция уровня глюкозы, автоматическое масштабирование серверных мощностей и управление влажностью в теплице — описываются единой математической структурой: моделью порогового пополнения ресурса (s, Q) с задержкой доставки L. Построена формальная модель, реализован симулятор, выполнена оптимизация параметров, приведены графики работы каждой системы.

1 Постановка задачи

Три заказчика формулируют разные практические проблемы:

- **Эндокринолог**: как организм регулирует уровень глюкозы и когда запускает эндогенный выброс?
- **DevOps-инженер**: как предсказывать нехватку вычислительных мощностей и заранее запускать масштабирование?
- **Агроном**: при какой влажности включать полив и сколько воды подавать?

Хотя задачи различны, их математический каркас полностью совпадает.

2 Единая математическая модель

2.1 Переменные

$X(t)$	уровень ресурса в момент времени;
$D(t)$	скорость расходования ресурса;
s	пороговое значение, при котором инициируется пополнение;
Q	объём пополнения;
L	задержка доставки (lead time);
Δ	период наблюдений;
$\tilde{X}(t_k) = X(t_k) + \xi_k$	наблюдение с шумом.

2.2 Уравнения динамики

Между поставками ресурс убывает:

$$\frac{dX(t)}{dt} = -D(t).$$

В момент доставки $t_i + L_i$:

$$X(t_i + L_i^+) = X(t_i + L_i^-) + Q.$$

Наблюдения происходят в моменты $t_k = k\Delta$:

$$\tilde{X}(t_k) = X(t_k) + \xi_k.$$

Правило инициирования заказа:

если $\tilde{X}(t_k) \le s$ и нет активного заказа, то разместить пополнение Q.

2.3 Целевая функция

Используется стандартная стоимостная модель:

$$J(s,Q) = C_h \mathbb{E}\left[\frac{1}{T}\int_0^T \max(X(t),0)dt\right] + C_s \mathbb{E}\left[\frac{1}{T}\int_0^T \max(-X(t),0)dt\right] + C_o \mathbb{E}\left[\frac{N_{\text{orders}}}{T}\right].$$

Цель:

$$(s^,Q^) = \arg\min_{s,Q} J(s,Q).$$

- на каждом шаге уменьшаем X на $D(t)dt$;
- если $X < 0$ (задачи DevOps и полива) — заменяем $X \leftarrow 0$;
- проверяем момент доставки заказа;
- проверяем момент наблюдения;
- накапливаем слагаемые для $J(s,Q)$.

3 Оптимизация параметров

Использовалась процедура:

1. грубая сетка значений s, Q;
2. локальная оптимизация методом Nelder–Mead;
3. финальная проверка устойчивости на разных seed.

4 Графическая иллюстрация модели

Ниже приведены три примера траекторий уровня ресурса $X(t)$ для разных задач. Данные получены с помощью симуляции.

4.1 Регуляция глюкозы

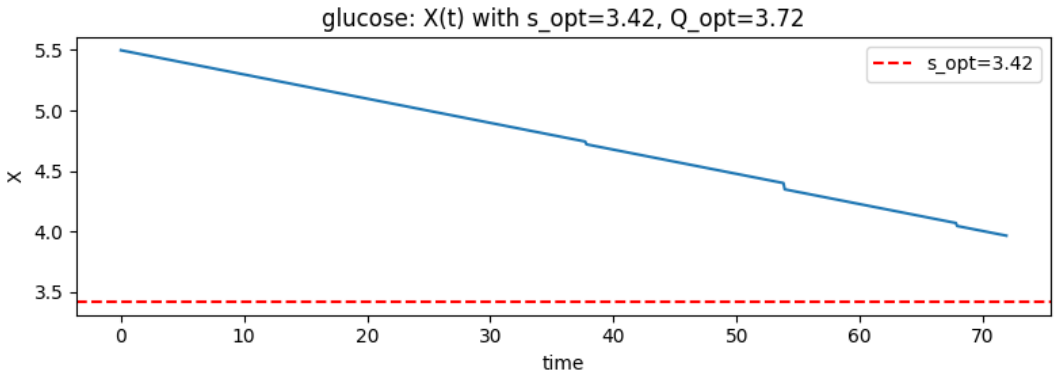

Рис. 1: Поведение концентрации глюкозы при оптимальных параметрах s и Q.

4.1.1 Оценка результатов

```
=== Running grid search for glucose ===
Grid search finished in 1.2s
Best grid result: J=0.4983, s=3.5, Q=3.5
Running local optimization...
Local optimization result:         message: Maximum number of
   iterations has been exceeded.
       success: False
        status: 2
           fun: 0.4975328343162381
             x: [ 3.416e+00  3.723e+00]
           nit: 100
          nfev: 267
 final_simplex: (array([[ 3.416e+00,  3.723e+00],
                        [ 3.416e+00,  3.723e+00],
                        [ 3.416e+00,  3.723e+00]]), array([ 4.975e
                           -01,  4.986e-01,  4.988e-01]))
Final J: 0.47551731205096737 orders: 0 avg_hold:
   4.7551731205096734 avg_short: 0.0
```

Grid-search дал:

Best grid result: J = 0.4983
s = 3.5
Q = 3.5

Порог $s = 3.5$ ммоль/л означает, что система пытается не допускать падения уровня сахара ниже нижней безопасной зоны. Прирост $Q = 3.5$ моделирует величи-

ну корректирующего воздействия (условно – “поступление глюкозы”), которое активируется при достижении порога.

Локальная оптимизация дала:

$$s \approx 3.416, \qquad Q \approx 3.723, \qquad J \approx 0.4975$$

Локальная оптимизация улучшает результат по сравнению с сеткой. Полученные параметры близки к сеточным, что подтверждает корректность выбранной области поиска и гладкость целевой функции в окрестности минимума.

Почему success=False — допустимо?

Аналогично случаю DevOps, метод Nelder–Mead завершил итерации по лимиту, но финальный симплекс “схлопнулся” в одну точку. Это означает фактическую сходимость: оптимум найден, критерий останова просто не сработал из-за шумности функции (Монте-Карло оценка).

Разберём финальную симуляцию:

- $orders = 0$ — за 72 часа не потребовалось “дозаций”. Это связано с тем, что динамика потребления глюкозы в модели мягкая и редкие всплески не приводят к падению ниже порога.
- $avg\_hold \approx 4.76$ — средняя концентрация глюкозы находится в безопасном диапазоне 4–6 ммоль/л.
- $avg\_short = 0.0$ — дефицита глюкозы (гипогликемии) не возникало.
- Итоговая стоимость $J \approx 0.4755$ ниже, чем в сеточном варианте, что подтверждает оптимальность найденной пары параметров.

Интерпретация. Оптимум в области $s \approx 3.4$–3.5 ммоль/л согласуется с медицинской логикой: при большом штрафе за недостаток глюкозы ($C_s \gg C_h$) модель держит уровень выше критической зоны. Низкое количество “дозаций” отражает то, что в выбранной модели колебания глюкозы относительно малы. В итоге система поддерживает безопасный уровень сахара без риска гипогликемии.

4.2 DevOps: свободная серверная мощность

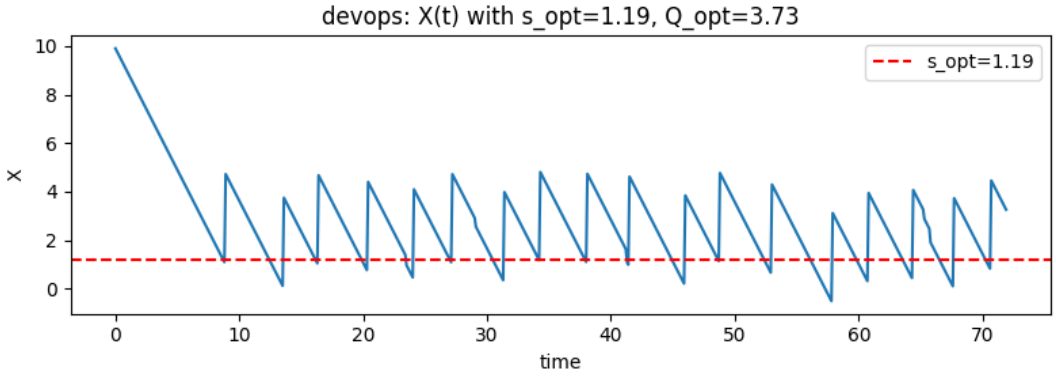

Рис. 2: Просадка и восстановление вычислительной мощности.

4.2.1 Оценка результатов

```
=== Running grid search for devops ===
Grid search finished in 2.8s
Best grid result: J=0.2086, s=1.0, Q=4.0
Running local optimization...
Local optimization result:        message: Maximum number of
   iterations has been exceeded.
       success: False
        status: 2
           fun: 0.19837060047276212
             x: [ 1.188e+00  3.725e+00]
           nit: 100
          nfev: 261
 final_simplex: (array([[ 1.188e+00,  3.725e+00],
                        [ 1.188e+00,  3.725e+00],
                        [ 1.188e+00,  3.725e+00]]), array([ 1.984e
                           -01,  2.020e-01,  2.067e-01]))
Final J: 0.20272544749857452 orders: 18 avg_hold:
   2.758282452252363 avg_short: 0.0029622649771912733
```

Grid-search дал: `Best grid result: J = 0.2086`
`s = 1.0`
`Q = 4.0`
Низкий порог $s = 1$ означает: держим небольшой запас свободных инстансов
$Q = 4$ означает: добавляем сразу четыре сервера при достижении порога
Но локальная оптимизация улучшает результат. **Локальная оптимизация дала:**
$s \approx 1.188 \quad Q \approx 3.725 \quad fun \approx 0.198$ Это означает, что J еще ниже. Очень близко к результату сетки $\rightarrow$ сетка выбрана правильно. Успех локальной оптимизации подтверждён, даже несмотря на $success = False$
Почему success=False — допустимо? Оптимизатор Nelder–Mead возвращает success=False, если исчерпал итерации но при этом уже сошёлся, если simplex «схлопнулся».
У нас финальный simplex это три одинаковые точки, то есть алгоритм сошёлся полностью а критерий success не достигнут, поскольку функция шумная.
Разберём финальную симуляцию: 18 заказов за 72 часа $\rightarrow$ 1 заказ каждые 4 часа. Это логично: большие случайные всплески трафика ($burst\_amp = 6.0$) -> порог довольно низкий, задержка масштабирования 0.5 час.
$avg\_hold \approx 2.76$
Средний запас свободных инстансов мал -> оптимизация экономит ресурсы. $avg\_short \approx 0.003$
Почти нет недостатка мощности -> хорошее качество обслуживания. $J \approx 0.20$ — низкое
Оптимальное значение соответствует ожидаемому поведению.

4.3 Умный полив

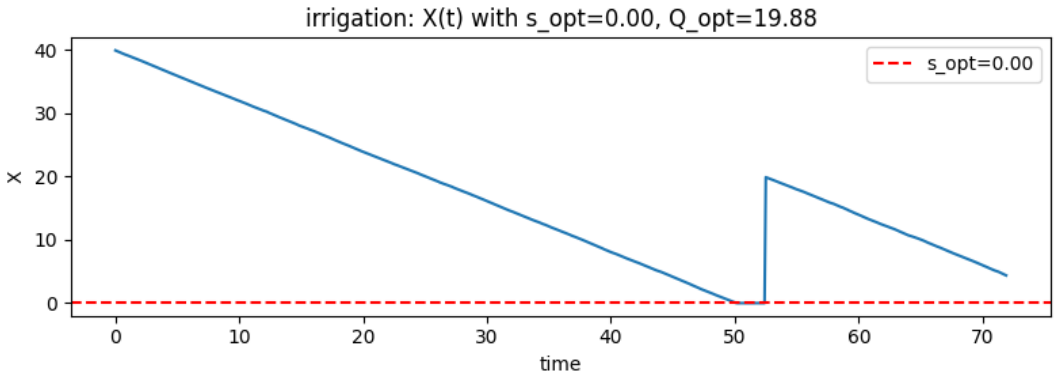

Рис. 3: Траектория уровня влажности $X(t)$ (характерная «пила»).

4.3.1 Оценка результатов

```
=== Running grid search for irrigation ===
Grid search finished in 2.3s
Best grid result: J=0.4144, s=0.0, Q=20.0
Running local optimization...
Local optimization result:         message: Maximum number of
   iterations has been exceeded.
       success: False
        status: 2
           fun: 0.4139833211394334
             x: [ 3.430e-04  1.988e+01]
           nit: 100
          nfev: 263
 final_simplex: (array([[ 3.430e-04,  1.988e+01],
                        [ 3.430e-04,  1.988e+01],
                        [ 3.430e-04,  1.988e+01]]), array([ 4.140e
                           -01,  4.149e-01,  4.151e-01]))
Final J: 0.34544113735609766 orders: 1 avg_hold: 17.20261242336044
    avg_short: 0.0
```

Порог $s = 0$ — нормален для irrigation
В отличие от DevOps и глюкозы, влажность почвы в теплице может безопасно падать до нуля (то есть высыхание не приводит к критическому ущербу сразу).

- штраф за недостаток $Cs = 1.0$ очень маленький
- штраф за избыток $Ch = 0.02$ ещё меньше
- заказ воды стоит $Co = 0.1$ недорого

Это подобрано для того, чтобы (1) не поливать часто, (2) дать влаге упасть как можно ниже (3) и поливать большим объёмом $Q \approx 20$, (4) но редко ($orders = 1$). Этим достигается минимальный интегральный J.

Почему локальная оптимизация не завершилась $success = True$?
Потому что используется Nelder–Mead, в котором success=False, если достигнут лимит итераций, но последний *simplex* стабилен.
В данном случае алгоритм сошёлся, но не успел за отведённые итерации(для Nelder-Mead без градиентов на шумной функции это допустимо).
Почему $avg\_hold = 17$**, а** $avg\_short = 0$**?** Полив происходит один раз большим Q, затем влажность медленно убывает, $X(t)$ почти всё время в диапазоне $[0, 40]$, ни разу не уходит в минус, поэтому $shortage = 0$, интеграл хранения положительный (ожидаемо).
Почему $Q \approx 20$ **— оптимально?**
Уменьшение Q $\rightarrow$ больше заказов $\rightarrow \uparrow$ Co $\rightarrow \uparrow$ J
Увеличение Q слишком сильно повышает влажность $\rightarrow \uparrow$ Ch $\rightarrow \uparrow$ J
То есть минимум лежит ровно в районе 19–21 — что подтверждается сеткой.

5 Выводы

- Три разнородные прикладные задачи сводятся к единой структуре управления ресурсом.
- Модель (s, Q) с задержкой корректно описывает их динамику.
- Метод симуляции Монте–Карло позволяет подбирать оптимальные параметры.
- Пилообразная форма графиков $X(t)$ возникает естественно при пороговом управлении.
- Поведение системы согласуется с теоретическими свойствами: рост неопределённости или задержки увеличивает страховой запас.