Применение генетического алгоритма для решения СЛУ и ОДУ

Баталов Семен

02.03.2021

1. Генетический алгоритм

Генетический алгоритм – это эвристический алгоритм поиска, используемый для решения задач оптимизации и моделирования путем случайного подбора, комбинирования и вариации искомых параметров с использованием механизмов, аналогичных естественному отбору в природе.

Задача формализуется таким образом, чтобы её решение могло быть закодировано в виде вектора («генотипа») генов, где каждый ген может быть битом, числом или неким другим объектом. Можно выделить следующие этапы генетического алгоритма:

- Инициализация
    1) Задать целевую функцию (приспособленности) для особей популяции
    2) Создать начальную популяцию
- Начало цикла
    1) Размножение (скрещивание)
    2) Мутирование
    3) Вычислить значение целевой функции (функции приспособленности) для всех особей
    4) Формирование нового поколения (селекция)
    5) Если выполняются условия остановки, то (конец цикла), иначе (начало цикла)

Далее считаем, что каждый ген является вещественным числом. Функция приспособленности зависит от задачи и будет определена позже. Условием остановки считаем достижение определенного количества итераций.

2. Инструменты

Языком разработки является «**Python**». Для работы с генетическим алгоритмом используется библиотека «**geneticalgorithm**». Подробнее о программе можно узнать в папке «**source**» проекта.

3. Решение ОДУ первого порядка

Мы рассматриваем задачу нахождения приближенного решения задачи Коши обыкновенного дифференциального уравнения первого порядка (1).

$$\dot{X} = F(t, X) \tag{1}$$

Естественно полагаем, что решение существует и единственно на промежутке $(a, b) \ni t_0$. Не умаляя общности, рассматриваем задачу Коши (2). Считаем, что решение выражается через элементарные функции.

$$t_0 = 0, \qquad X(t_0) = X_0 = 0 \tag{2}$$

Решение задачи (1), (2) приближаем полиномом p степени $deg(p) = 7$ (это число было подобрано экспериментально, с увеличением степени эффективность алгоритма падает). Генами при данной постановке вопроса являются коэффициенты многочлена при соответствующих степенях, таким образом длина «генотипа» равна 7.

Важно отметить, что количесво особей в каждом поколении постоянно и равно 100, другие параметры алгоритма можно посмотреть в коде программы. В качестве функции приспособленности используем среднее квадратичное отклонение (3).

$$Err(p) = \sqrt{\sum_{i=0}^{num-1} \Big(\dot{p}(t_i) - F(t_i, p(t_i))\Big)^2 + \Big(p(t_0) - X_0\Big)^2} \tag{3}$$

Здесь num - количество точек в дроблении промежутка (a, b). Далее рассмотрим примеры решений некоторых ДУ. На рисунках слева указано приближение полиномом, а справа изображено отклонение (разность значений) полинома от оригинальной функции.

3.1. $\dot{X} = cos(t)$

Решение задачи Коши (1), (2): $X(t) = sin(t)$. Промежуток задания: $(-1.5, 1.5)$. Дробление интервала осуществляется с шагом 0.1.

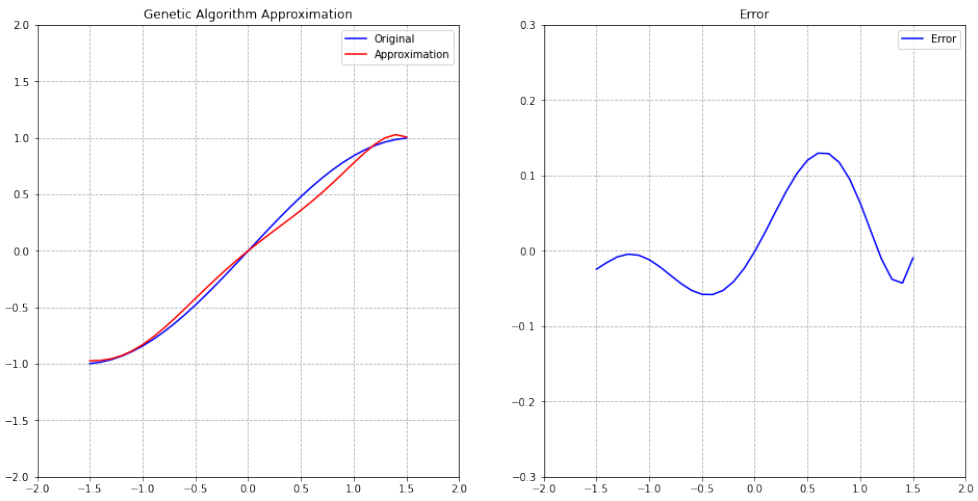

Рис. 1. Приближенное решение $\dot{X} = cos(t)$, 300 поколений.

На рис. 3.1.1 представлен результат работы программы для 300 поколений. Видно, что ошибка достаточно велика. Далее представлены результаты для 800 и 1000 поколений соответственно.

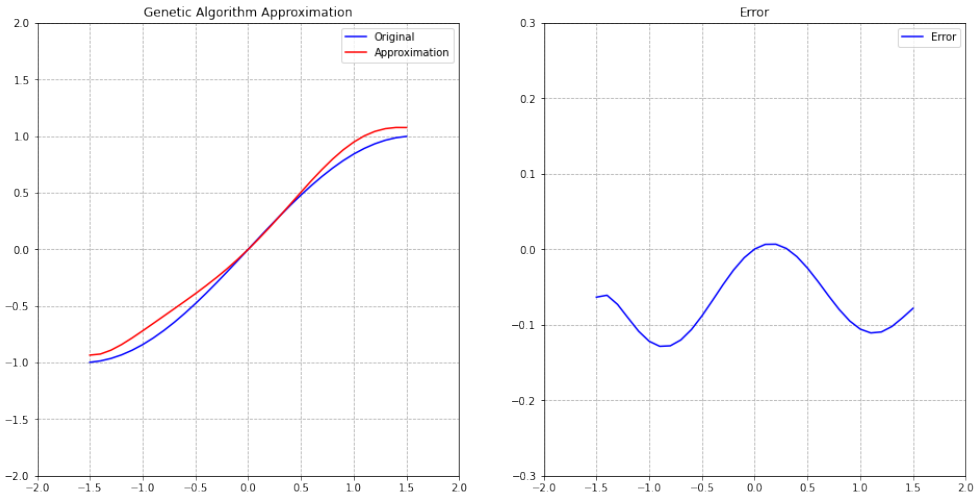

Рис. 2. Приближенное решение $\dot{X} = cos(t)$, 800 поколений.

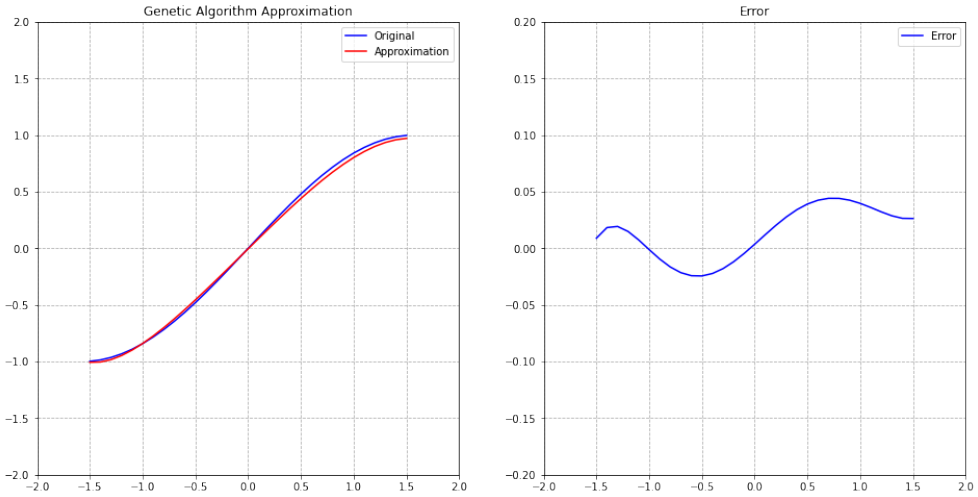

Рис. 3. Приближенное решение $\dot{X} = cos(t)$, 1000 поколений.

Нетрудно видеть, что на рис. 3.1.3 наименьшее отклонение значений полинома от значений решения ДУ.

3.2. $\dot{X} = X^2 + 1$

Решение задачи Коши (1), (2): $X(t) = tg(t)$. Промежуток задания: $(-1, 1)$. Дробление интервала осуществляется с шагом 0.1.

В отличие от предыдущего примера, здесь зависимость точности приближения от количества поколений прослеживается хуже. Лучший результат представлен на рис. 3.2.2, 800 поколений.

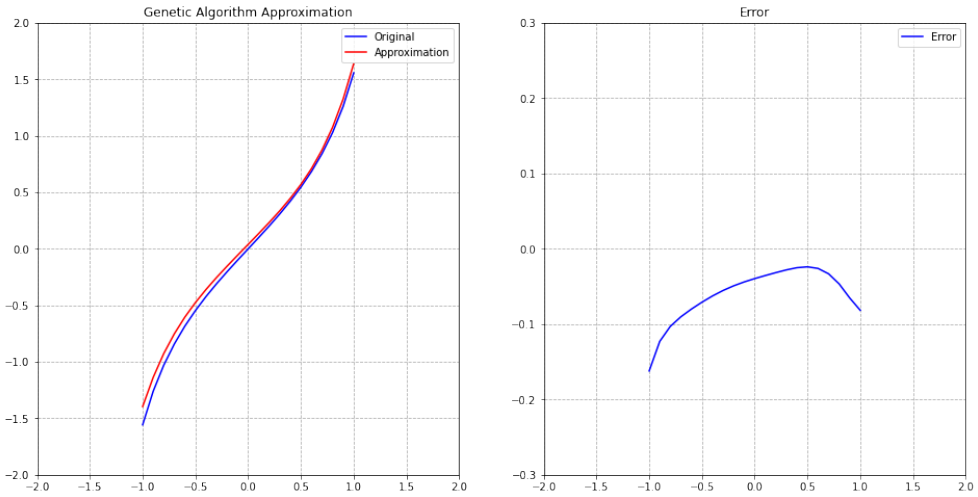

Рис. 4. Приближенное решение $\dot{X} = X^2 + 1$, 300 поколений.

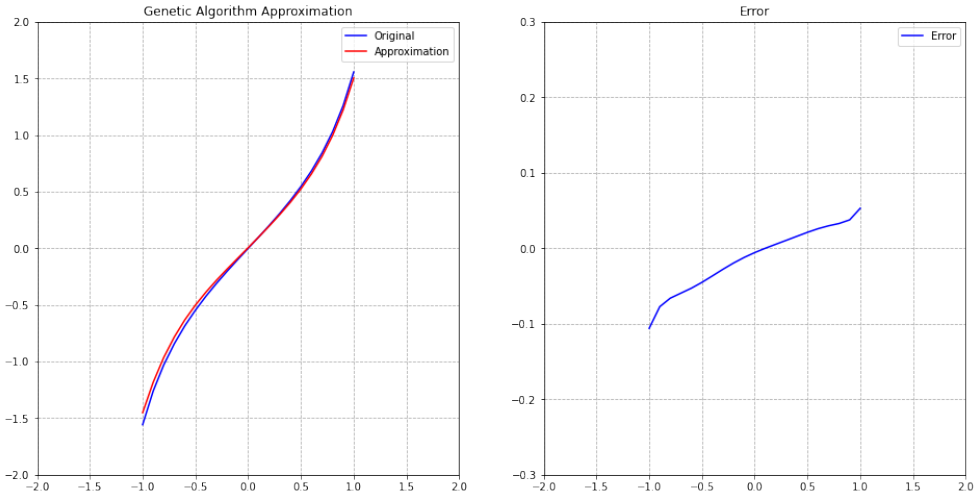

Рис. 5. Приближенное решение $\dot{X} = X^2 + 1$, 800 поколений.

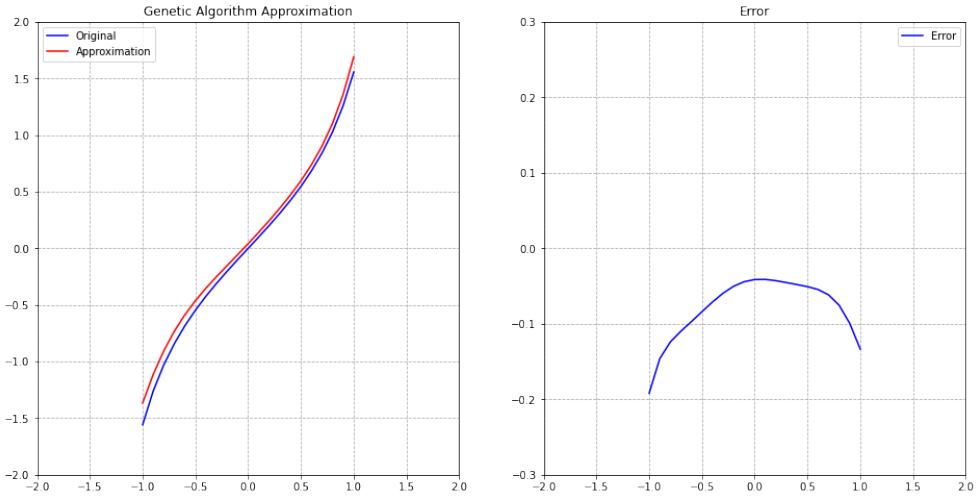

Рис. 6. Приближенное решение $\dot{X} = X^2 + 1$, 1000 поколений.

3.3. $\dot{X} = -X$

Решение задачи Коши (1), (2): $X(t) = -\frac{X^2}{2}$. Промежуток задания: $(-1.5, 1.5)$. Дробление интервала осуществляется с шагом 0.1.

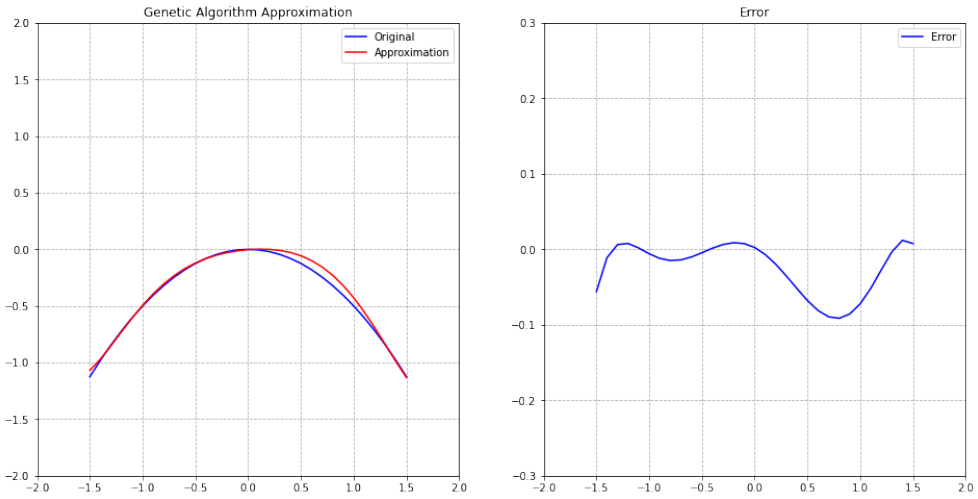

Рис. 7. Приближенное решение $\dot{X} = -X$, 300 поколений.

До этого мы рассматривали нечетные функции, четные алгоритм тоже приближает с приемлемой точностью. Стоит отметить, что результат работы программы может меняться при разных запусках, это объясняется вероятностной природой генетического алгоритма.

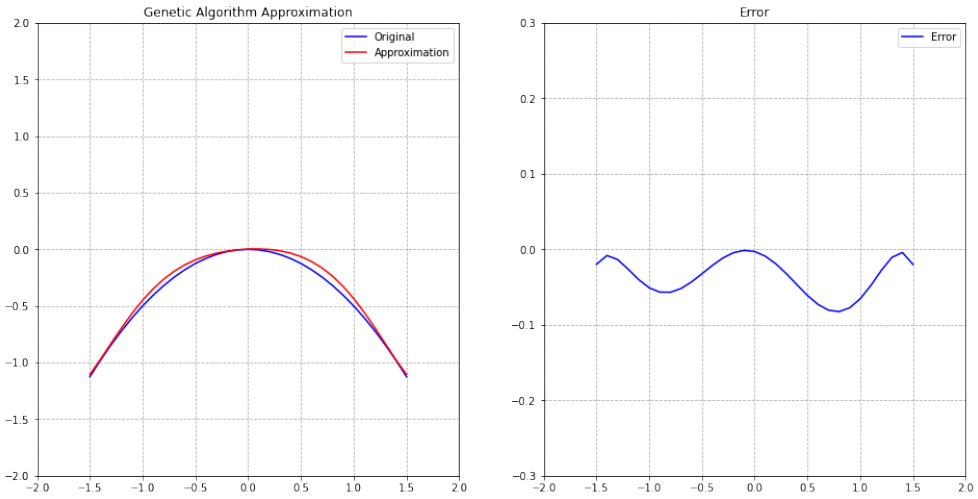

Рис. 8. Приближенное решение $\dot{X} = -X$, 800 поколений.

Видно, что результат работы алгоритма слабо зависит от количества итераций. Иногда при меньшем количестве поколений можно добиться большей точности. Далее рассмотрим приближение для 1000 итераций.

3.4. Выводы

В целом удалось решить проблему поиска решения задачи Коши ДУ первого порядка с помощью генетического алгоритма. Однако, возникает большое количество

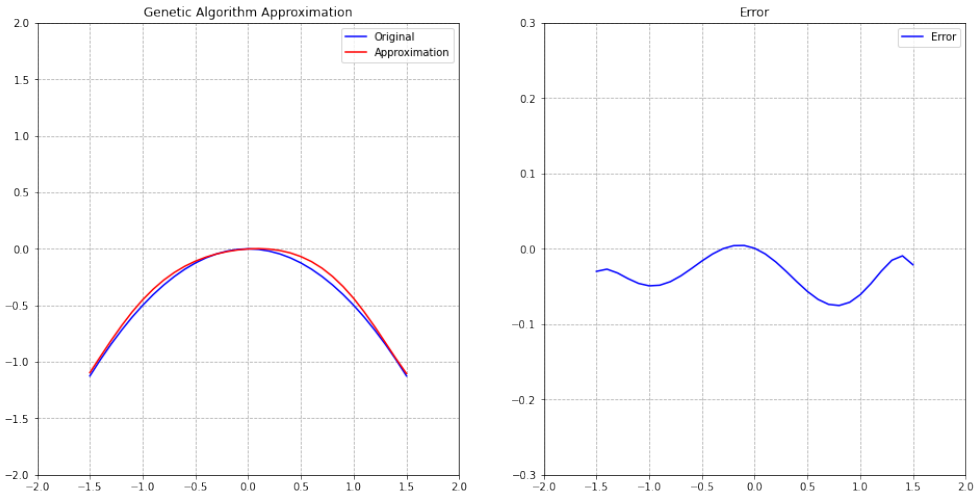

Рис. 9. Приближенное решение $\dot{X} = -X$, 1000 поколений.

проблем, связанных с вычислением приближенного решения. Мы аппроксимировали элементарные функции полиномом, это уже вносит неточность в ответ, более того нельзя без потерь увеличить степень этого многочлена, так как алгоритм сильно замедлится, если вообще сможет сойтись. Также недостатком метода является отсутствие информации (в общем случае) о диапазоне, в котором должен находится соответствующий коэффициент полинома. Если диапазон задать слишком широким, то алгоритм будет дольше сходиться, и точность полученного решения окажется меньшей.

Таким образом, генетический алгоритм способен находить приближенное решение задачи Коши ОДУ (первого порядка), выраженное полиномом, но данный метод имеет много недостатков и ограничений.

4. Решение СЛНУ

Теперь рассмотрим задачу нахождения приближенного решения системы линейных неоднородных уравнений (4). Будем рассматривать системы трех уравнений (для систем других порядков все результаты аналогичны).

$$\begin{pmatrix} a_{11} & a_{12} & a_{13} \\ a_{21} & a_{22} & a_{23} \\ a_{31} & a_{32} & a_{33} \end{pmatrix} \cdot \begin{pmatrix} x_1 \\ x_2 \\ x_3 \end{pmatrix} = \begin{pmatrix} b_1 \\ b_2 \\ b_3 \end{pmatrix}, \qquad X_0 = \begin{pmatrix} x_1^0 \\ x_2^0 \\ x_3^0 \end{pmatrix} \tag{4}$$

Будем считать, что решение системы (4) существует и единственно. В качестве генотипа будем брать вектор $X = (x_1, x_2, x_3)^T$. Дианазон значений для каждой координаты считаем известным заранее. Важно определить функцию приспособленности. Мы будем использовать сумму абсолютных значений отклонения каждой координаты (5).

$$Err(X) = \sum_{i=1}^{3} |a_{i1}x_1 + a_{i2}x_2 + a_{i3}x_3 - b_i| \tag{5}$$

Количество итераций алгоритма будет меняться, количество особей в рамках одного примера в каждом поколении постоянно. Дополнительные параметры алгорит-

ма можно увидеть в файле программы.

Далее потребуется анализировать величину отклонения приближенного решения системы от действительного, будем вычислять его по формуле (6).

$$Dev(X) = \sum_{i=1}^{3} |x_i - x_i^0| \tag{6}$$

4.1. Пример 1

Рассмотрим СЛНУ из трех уравнений (7). В соответствии с решением этой системы накладываем ограничения на диапазон возможных значений генов $x_1, x_2, x_3 \in [-5, 5]$.

$$\begin{pmatrix} 1 & 6 & 3 \\ 11 & 7 & 6 \\ 7 & 8 & 9 \end{pmatrix} \cdot \begin{pmatrix} x_1 \\ x_2 \\ x_3 \end{pmatrix} = \begin{pmatrix} 7 \\ 2 \\ 3 \end{pmatrix}, \qquad X_0 = \begin{pmatrix} -0.428 \\ 1.628 \\ -0.781 \end{pmatrix} \tag{7}$$

Запустим алгоритм для разного количества особей и числа поколений, составим график зависимости отклонения (6) приближенного решения от действительного от числа итераций для 100 и 200 особей (рис. 4.1.10). Значение ошибки в конкретном случае соответствует среднему арифметическому ошибок для 10 запусков программы.

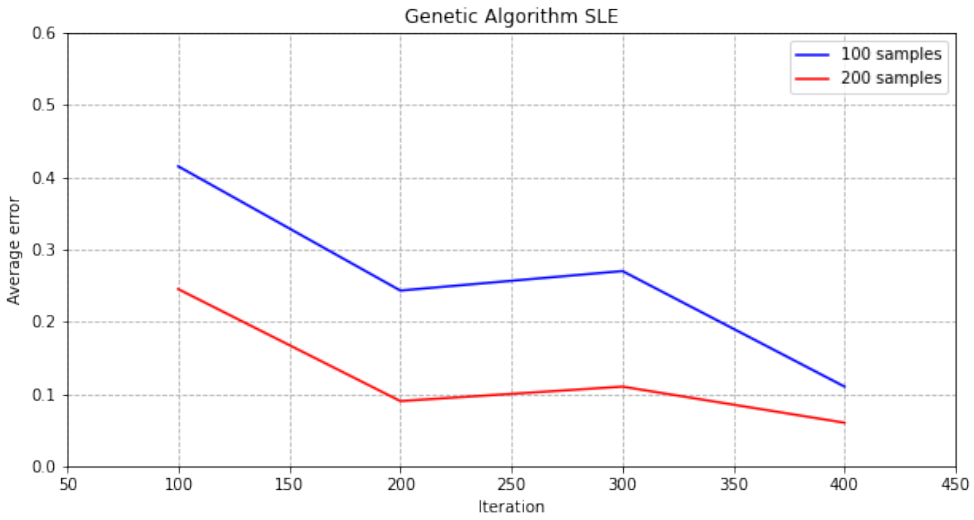

Рис. 10. Зависимость отклонения (6) приближенного решения системы (7) от числа итераций генетического алгоритма.

Заметим, что на точность решения оказывют влияние оба фактора: количество особей в каждом поколении, количество итераций алгоритма. При меньшем количестве особей потребуется больше поколений, чтобы получить требуемую точность.

4.2. Пример 2

Рассмотрим аналогичную СЛНУ из трех уравнений (8). В соответствии с решением этой системы накладываем ограничения на диапазон возможных значений генов

$x_1, x_2, x_3 \in [-10, 10]$. Диапазон значений расширился в сравнении с предыдущим примером.

$$\begin{pmatrix} 2 & 5 & 33 \\ 10 & 11 & 61 \\ 8 & 13 & 9 \end{pmatrix} \cdot \begin{pmatrix} x_1 \\ x_2 \\ x_3 \end{pmatrix} = \begin{pmatrix} 16 \\ 7 \\ 32 \end{pmatrix}, \qquad X_0 = \begin{pmatrix} -5.167 \\ 5.685 \\ -0.063 \end{pmatrix} \tag{8}$$

Запустим алгоритм для разного количества особей и числа поколений, составим график зависимости отклонения (6) приближенного решения от действительного от числа итераций для 100 и 200 особей (рис. 4.2.11). Значение ошибки в конкретном случае соответствует среднему арифметическому ошибок для 10 запусков программы.

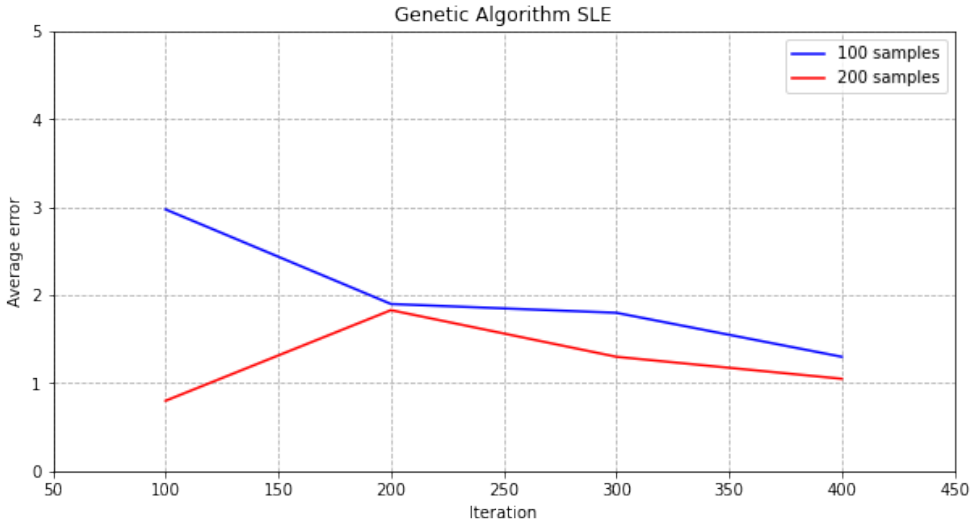

Рис. 11. Зависимость отклонения (6) приближенного решения системы (8) от числа итераций генетического алгоритма.

В целом результаты аналогичны результатам примера 1. Ярко выражена зависимость точности решения от числа особей в каждом поколении. Число итераций алгоритма также влияет на точность результата.

4.3. Вывод

При решении СЛНУ при помощи генетического алгоритма существенное влияние на точность решения оказывают число итераций (поколений) алгоритма и количество особей в каждом поколении. В соответствии с выбранной метрикой (5), (6) и постановкой вопроса можно заключить, что алгоритм находит решение задачи с достаточной точностью.

5. Общий вывод

В результате проведенной работы во многом стали очевидны положительные и отрицательные стороны применения генетического алгоритма. Стоит отметить его простоту, не требуется понимать всю специфику задачи, достаточно правильно подобрать метрику, составить генотип и установить вспомогательные параметры, чтобы

получить достаточно точное решение. Однако есть серьезный недостаток, связанный с вероятностной природой алгоритма. Нельзя абсолютно точно предсказать исход его работы (во многих экспериментах он работал нестабильно). Подобный недостаток сужает круг задач, в которых генетический алгоритм может быть применен.