Химические реакции, стохастическое горение.

этап 4

Содержание

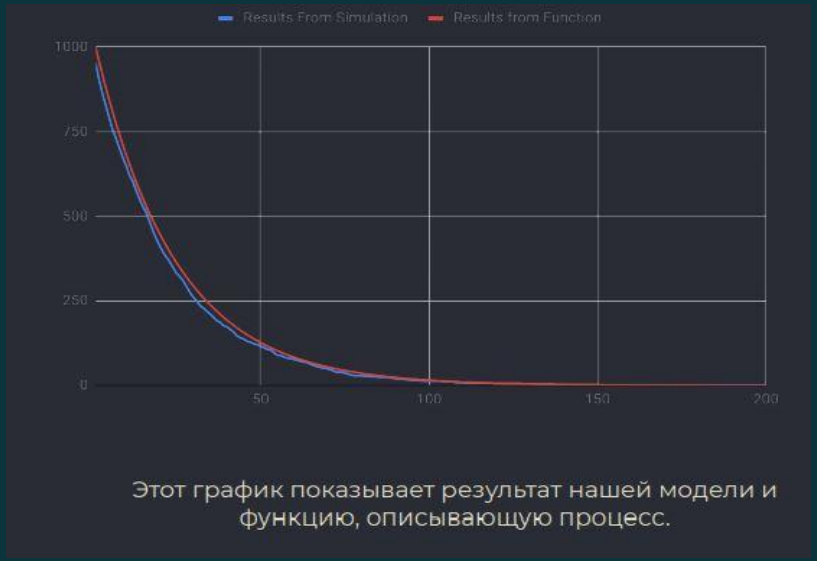
Теоретическое задание

в конечном счете, то, что мы должны помнить на теоретическом уровне для реализации нашего проекта химические реакции стохастическое горение.
действительно, для реализации мы должны знать определение Мономолекулярная реакция,Энергия активации
теоретическая формула, и это то, что мы сделали на первом этапе.

Химические реакции, стохастическое горение.

алгоритмы, используемые для нашего проекта

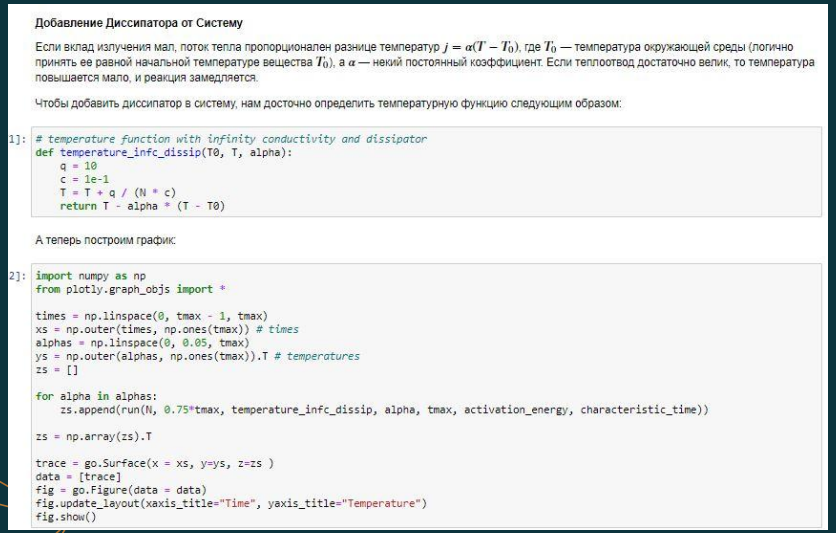
на втором этапе мы рассматривали некоторые алгоритмы : Алгоритм с разными типами теплопроводности и Функции Описывающие крайних случаев и также рассматривали решение системы дифференциальных уравнений ПРОГНОЗИРУЕМЫЕ РЕЗУЛЬТАТЫ
для построения графика системы без теплопроводности:
Изменяя константы, мы можем заметить, что при низких начальных температурах <60 Кельвинов реакция занимает слишком много времени для завершения во всех молекулах .
 также пришлось резко снизить уровень энергии активации (1e-20), чтобы реакция завершилась в разумные сроки. Это имеет смысл, потому что чем ниже энергия активации, тем легче системе завершить реакцию.



Этот график показывает результат нашей модели и функцию, описывающую процесс.

Программирование с помощью python

на третьем этапе мы использовали язык программирования python для реализации нашего проекта .
почему python, потому что облегчает кривую обучения и обеспечивает более высокий уровень успеха развертывания. Он позволяет быстро исправлять и обновлять приложения, сообщая об ошибках перед запуском.



Добавление Диссипатора от Систему

Если вклад излучения мал, поток тепла пропорционален разнице температур $j = \alpha(T - T_0)$, где T_0 — температура окружающей среды (логично принять ее равной начальной температуре вещества T_0), а α — некий постоянный коэффициент. Если теплоотвод достаточно велик, то температура повышается мало, и реакция замедляется.

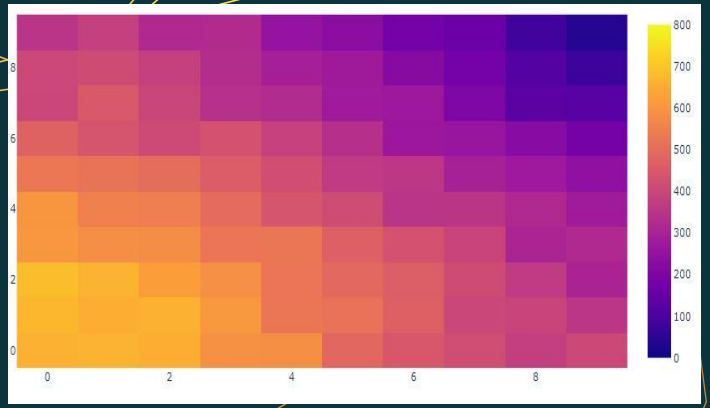
Чтобы добавить диссипатор в систему, нам досточно определить температурную функцию следующим образом:

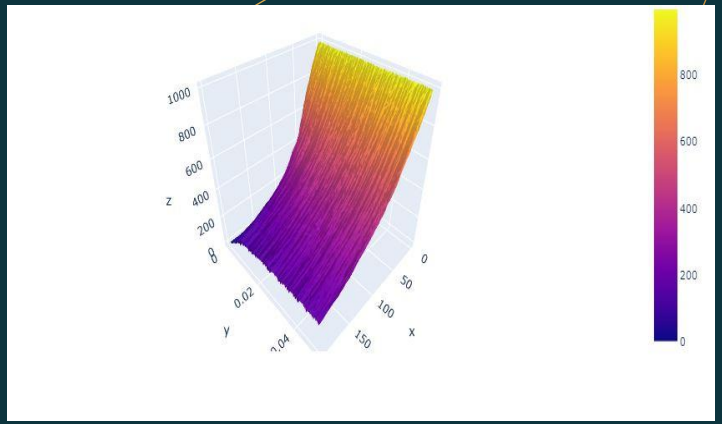
1]:
```
# temperature function with infinity conductivity and dissipator
def temperature_infc_dissip(T0, T, alpha):
    q = 10
    c = 1e-1
    T = T + q / (N * c)
    return T - alpha * (T - T0)
```

А теперь построим график:

2]:
```
import numpy as np
from plotly.graph_objs import *

times = np.linspace(0, tmax - 1, tmax)
xs = np.outer(times, np.ones(tmax)) # times
alphas = np.linspace(0, 0.05, tmax)
ys = np.outer(alphas, np.ones(tmax)).T # temperatures
zs = []

for alpha in alphas:
    zs.append(run(N, 0.75*tmax, temperature_infc_dissip, alpha, tmax, activation_energy, characteristic_time))

zs = np.array(zs).T

trace = go.Surface(x = xs, y=ys, z=zs )
data = [trace]
fig = go.Figure(data = data)
fig.update_layout(xaxis_title="Time", yaxis_title="Temperature")
fig.show()
```

результат, полученный в конце нашего проекта





В конце нашего проекта мы можем сказать, что достигли удовлетворительных результатов, как показано на нашем графике. для тех, кто заинтересован, вы можете проверить наш аккаунт на github или все шаги нашей реализации уже есть.

Eleobert/exo-reaction (github.com)