Министерство образования и науки РФ

Федеральное государственное бюджетное образовательное учреждение высшего профессионального образования

Рыбинский государственный авиационный технический университет имени П. А. Соловьева

Отчет по лабораторной работе №3

по дисциплине

Исследование операции

на тему

«Определение экстремумов функции одной переменной методом золотого сечения и методом Фибоначчи»

Студент группы ИПБ-18 Кондратенко М.М.

Преподаватель Задорина Н.А.

Рыбинск 2020

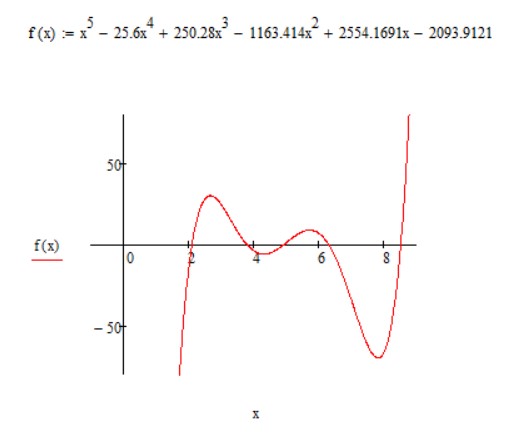
# **Формулировка задачи**

Найти локальные минимумы и максимумы функции f (x):

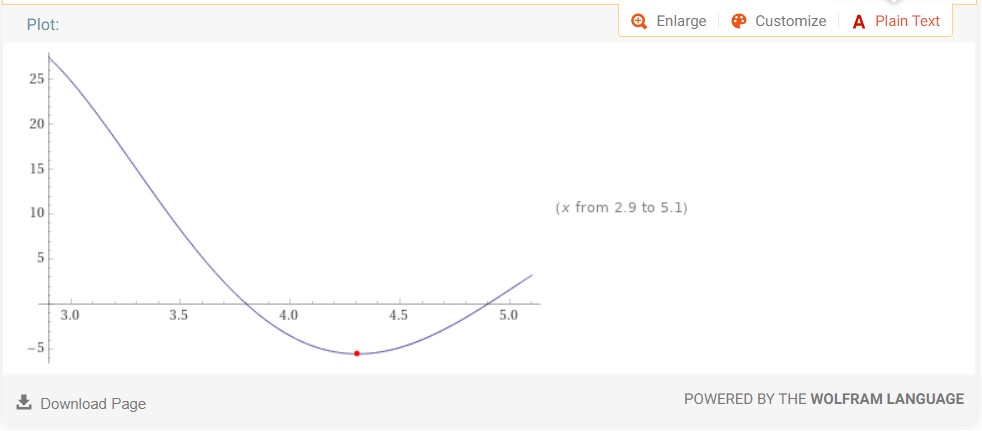
- методом золотого сечения;

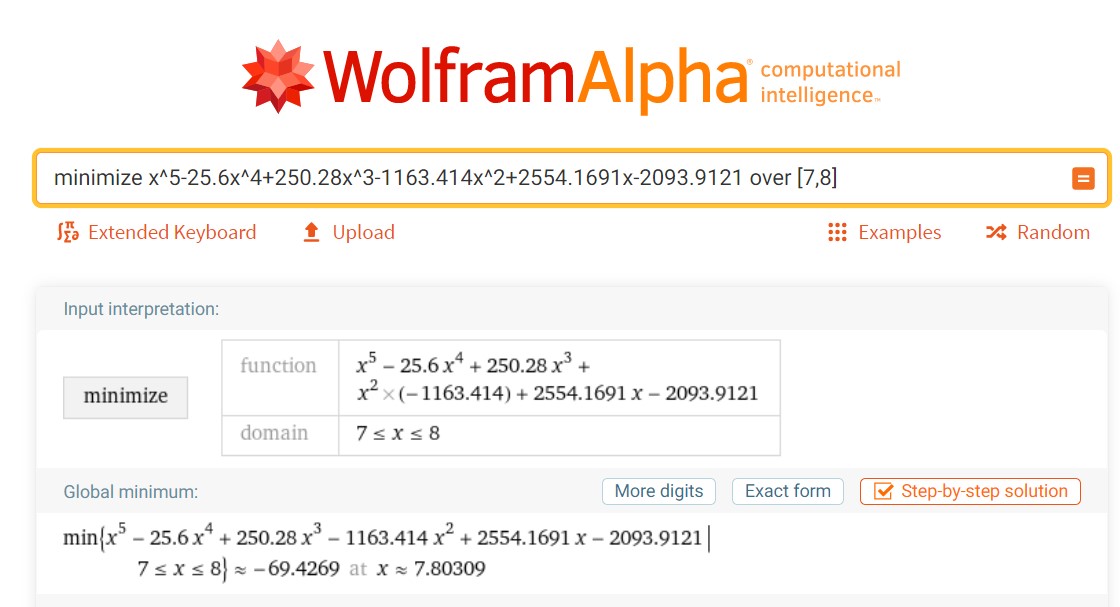
- методом Фибоначчи:

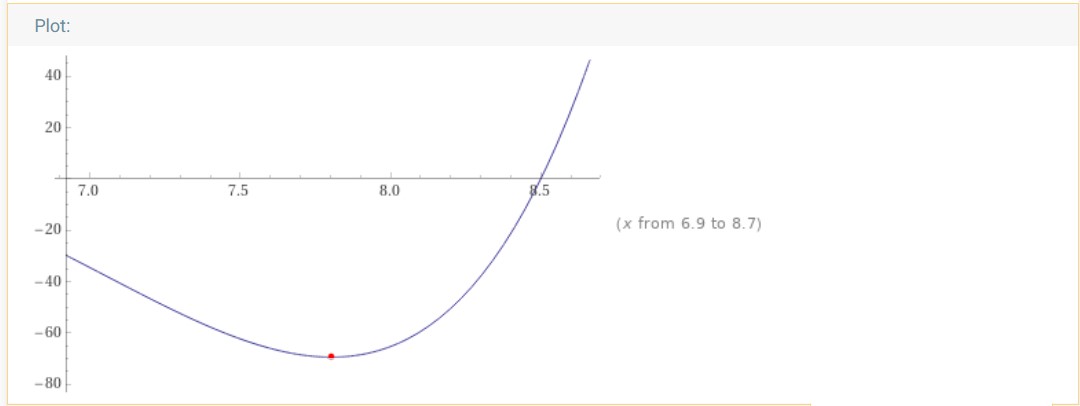
# **Результаты расчетов в Mathcad и WolframAlpha**

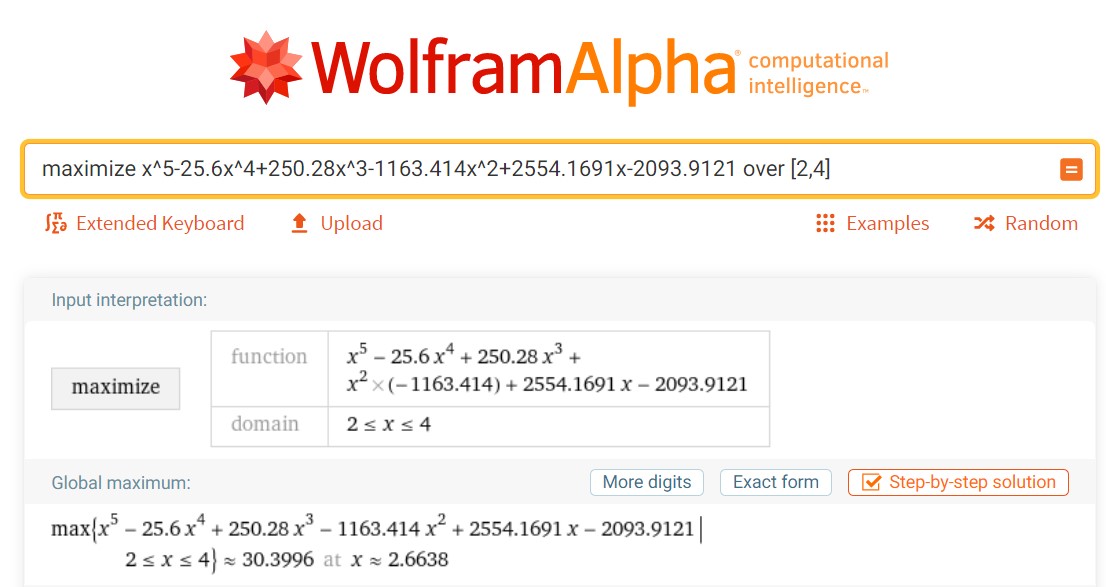


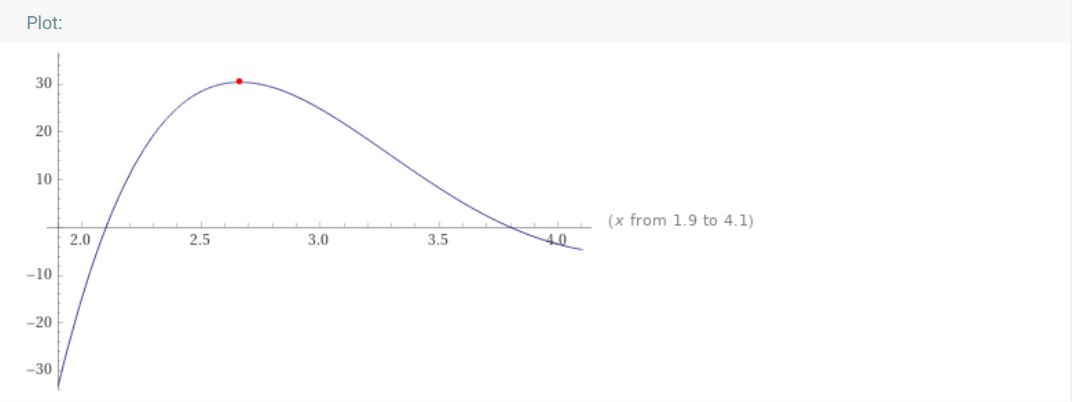


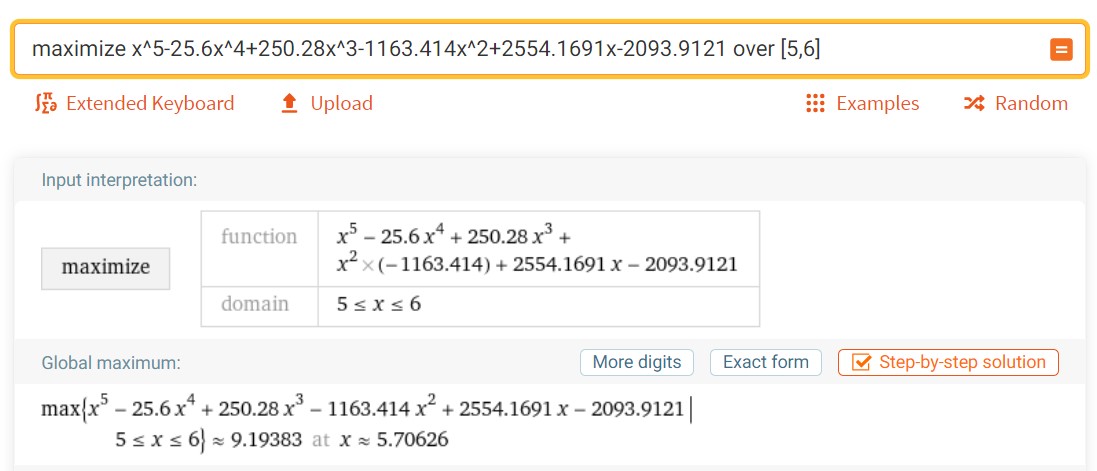


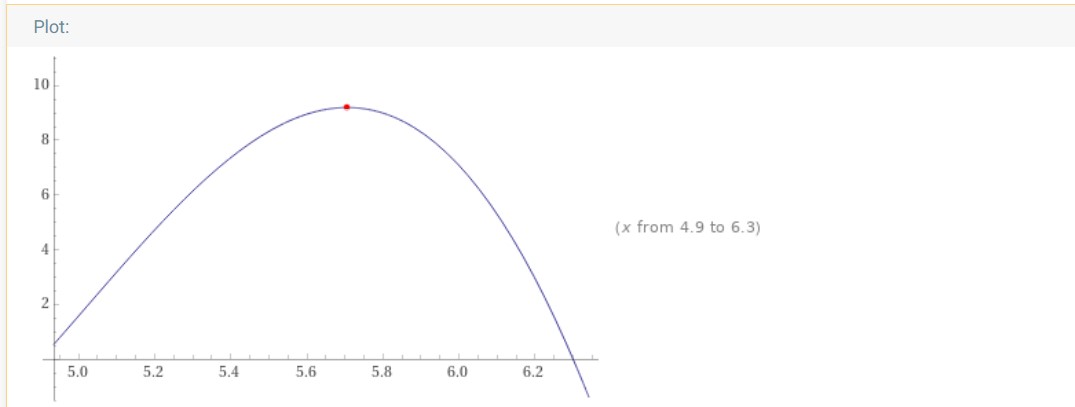












# **Код программы.**

from math import sqrt

# Функция f(x)

def f(x):

return x \*\* 5 - 25.6 \* x \*\* 4 + 250.28 \* x \*\* 3 - 1163.414 \* x \*\* 2 + 2554.1691 \* x - 2093.9121

# Метод золотого сечения

def golden(f, a, b, eps, min=True):

q = 1 if min else -1

while abs(f(a) - f(b)) >= eps:

x1 = b - (b - a) / ((1 + sqrt(5)) / 2)

x2 = a + (b - a) / ((1 + sqrt(5)) / 2)

if f(x1) \* q >= f(x2) \* q:

a = x1

else:

b = x2

return (a + b) / 2

fib = [1, 1]

# Вычислить числа Фибоначчи от 1 до n

def calcfib(n):

global fib

if len(fib) - 1 < n:

fib += [calcfib(n - 2) + calcfib(n - 1)]

return fib[n]

# Метод Фибоначчи

def fibmethod(f, a, b, eps, n, min=True):

calcfib(n)

q = 1 if min else -1

k = 1

while (abs(f(a) - f(b)) >= eps) and (k < n):

x1 = a + fib[n - k - 1] / fib[n - k + 1] \* (b - a)

x2 = a + fib[n - k] / fib[n - k + 1] \* (b - a)

if f(x1) \* q >= f(x2) \* q:

a = x1

else:

b = x2

k += 1

return (a + b) / 2

def main():

print('Золотое сечение:')

print('Минимумы:')

x = golden(f, 4, 5, 0.0001, True)

print('x={:.4f}'.format(x), 'f={:.4f}'.format(f(x)))

x = golden(f, 7, 8, 0.0001, True)

print('x={:.4f}'.format(x), 'f={:.4f}'.format(f(x)))

print('Максимумы:')

x = golden(f, 2, 4, 0.0001, False)

print('x={:.4f}'.format(x), 'f={:.4f}'.format(f(x)))

x = golden(f, 5, 6, 0.0001, False)

print('x={:.4f}'.format(x), 'f={:.4f}'.format(f(x)))

print('Метод Фибоначчи:')

print('Минимумы:')

x = fibmethod(f, 4, 5, 0.0001, 20, True)

print('x={:.4f}'.format(x), 'f={:.4f}'.format(f(x)))

x = fibmethod(f, 7, 8, 0.0001, 20, True)

print('x={:.4f}'.format(x), 'f={:.4f}'.format(f(x)))

print('Максимумы:')

x = fibmethod(f, 2, 4, 0.0001, 20, False)

print('x={:.4f}'.format(x), 'f={:.4f}'.format(f(x)))

x = fibmethod(f, 5, 68, 0.0001, 20, False)

print('x={:.4f}'.format(x), 'f={:.4f}'.format(f(x)))

if \_\_name\_\_ == '\_\_main\_\_':

main()

# **Результаты расчетов программы.**

Золотое сечение:

Минимумы:

x=4.3065 f=-5.5449

x=7.8030 f=-69.4269

Максимумы:

x=2.6637 f=30.3996

x=5.7067 f=9.1938

Метод Фибоначчи:

Минимумы:

x=4.3065 f=-5.5449

x=7.8030 f=-69.4269

Максимумы:

x=2.6637 f=30.3996

x=67.9971 f=979833828.9383

# **Вывод.**

В ходе работы определено, что метод дихотомии не эффективен в том смысле, что для конечного фиксированного числа n вычислений значений функции, он не приводит к наименьшему возможному интервалу. Эффективным методом в этом смысле является метод Фибоначчи.  
Важнейшая особенность этого метода состоит в том, что он позволяет для заранее заданного числа вычислений функции построить оптимальную процедуру поиска минимума функции.

Не всегда можно определить заранее, сколько раз придется вычислять функцию. Метод золотого сечения почти столь же эффективен при n-2, что и [метод Фибоначчи](https://math.semestr.ru/optim/fibonacci.php), однако при этом не требуется знать n – количество вычислений функции.