# Звіт до ЛР3 з дисципліни “Чисельні методи”

Роботу виконав студент групи К-27 Дергунов Микита Миколайович

## Необхідна інформація

Репозиторій з роботою: github.com/nndergunov/NMLab3

Мова програмування: Go v1.18.

За потреби можу надати бінарні файли для запуску на *майже* будь-якій комбінації ОС та архітектур (для побудови графіків апроксимації необхідно мати gnuplot в PATH).

**1. Побудувати поліном Лагранжа, що апроксимує значення функції.**

Код: /lagrange

Для початкових значень було прораховано значення функції 16 разів.

Далі було розроблено функцію, яка приймає ці значення й апроксимує значення функції в потрібній x за наступними формулами поліному Лагранжа:

$$\sum_{i=0}^{n} L_i(x) y_i$$

$$L_i(x) = \prod_{j=0;\, j\neq i}^{n} \frac{x - x_j}{x_i - x_j}$$

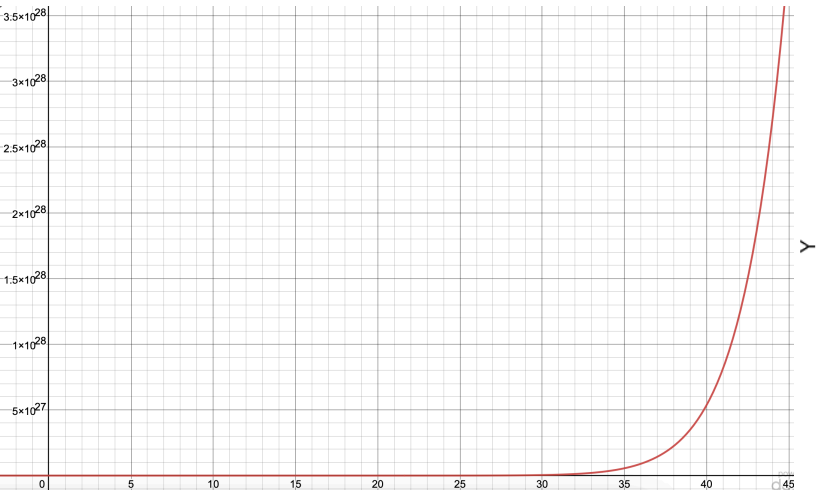
Повний отриманий поліном для n=16 записується в lagrange.txt:

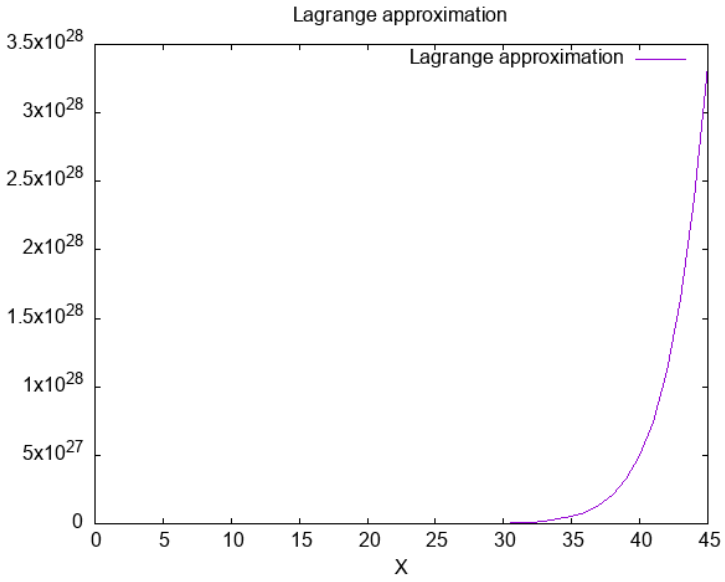
lagrange.txt

```
y = 8 * (x - 2) / (1 - 2) * (x - 3) / (1 - 3) * (x - 4) / (1 - 4) * (x - 5) / (1 - 5) * (x - 6) / (1 - 6) * (x - 7) / (1 - 7) * (x - 8) / (1 - 8) * (x
- 9) / (1 - 9) * (x - 10) / (1 - 10) * (x - 11) / (1 - 11) * (x - 12) / (1 - 12) * (x - 13) / (1 - 13) * (x - 14) / (1 - 14) * (x - 15) / (1 - 15) * (x
- 16) / (1 - 16) + 720896 * (x - 1) / (2 - 1) * (x - 3) / (2 - 3) * (x - 4) / (2 - 4) * (x - 5) / (2 - 5) * (x - 6) / (2 - 6) * (x - 7) / (2 - 7) * (x
- 8) / (2 - 8) * (x - 9) / (2 - 9) * (x - 10) / (2 - 10) * (x - 11) / (2 - 11) * (x - 12) / (2 - 12) * (x - 13) / (2 - 13) * (x - 14) / (2 - 14) * (x -
15) / (2 - 15) * (x - 16) / (2 - 16) + 602654094 * (x - 1) / (3 - 1) * (x - 2) / (3 - 2) * (x - 4) / (3 - 4) * (x - 5) / (3 - 5) * (x - 6) / (3 - 6) *
(x - 7) / (3 - 7) * (x - 8) / (3 - 8) * (x - 9) / (3 - 9) * (x - 10) / (3 - 10) * (x - 11) / (3 - 11) * (x - 12) / (3 - 12) * (x - 13) / (3 - 13) * (x
- 14) / (3 - 14) * (x - 15) / (3 - 15) * (x - 16) / (3 - 16) + 73014444032 * (x - 1) / (4 - 1) * (x - 2) / (4 - 2) * (x - 3) / (4 - 3) * (x - 5) / (4 -
5) * (x - 6) / (4 - 6) * (x - 7) / (4 - 7) * (x - 8) / (4 - 8) * (x - 9) / (4 - 9) * (x - 10) / (4 - 10) * (x - 11) / (4 - 11) * (x - 12) / (4 - 12) *
(x - 13) / (4 - 13) * (x - 14) / (4 - 14) * (x - 15) / (4 - 15) * (x - 16) / (4 - 16) + 3051757812500 * (x - 1) / (5 - 1) * (x - 2) / (5 - 2) * (x -
3) / (5 - 3) * (x - 4) / (5 - 4) * (x - 6) / (5 - 6) * (x - 7) / (5 - 7) * (x - 8) / (5 - 8) * (x - 9) / (5 - 9) * (x - 10) / (5 - 10) * (x - 11) / (5
- 11) * (x - 12) / (5 - 12) * (x - 13) / (5 - 13) * (x - 14) / (5 - 14) * (x - 15) / (5 - 15) * (x - 16) / (5 - 16) + 64885527871488 * (x - 1) / (6 -
1) * (x - 2) / (6 - 2) * (x - 3) / (6 - 3) * (x - 4) / (6 - 4) * (x - 5) / (6 - 5) * (x - 7) / (6 - 7) * (x - 8) / (6 - 8) * (x - 9) / (6 - 9) * (x -
10) / (6 - 10) * (x - 11) / (6 - 11) * (x - 12) / (6 - 12) * (x - 13) / (6 - 13) * (x - 14) / (6 - 14) * (x - 15) / (6 - 15) * (x - 16) / (6 - 16) +
864056194809626 * (x - 1) / (7 - 1) * (x - 2) / (7 - 2) * (x - 3) / (7 - 3) * (x - 4) / (7 - 4) * (x - 5) / (7 - 5) * (x - 6) / (7 - 6) * (x - 8) / (7
- 8) * (x - 9) / (7 - 9) * (x - 10) / (7 - 10) * (x - 11) / (7 - 11) * (x - 12) / (7 - 12) * (x - 13) / (7 - 13) * (x - 14) / (7 - 14) * (x - 15) / (7
- 15) * (x - 16) / (7 - 16) + 8162774324609024 * (x - 1) / (8 - 1) * (x - 2) / (8 - 2) * (x - 3) / (8 - 3) * (x - 4) / (8 - 4) * (x - 5) / (8 - 5) * (x
- 6) / (8 - 6) * (x - 7) / (8 - 7) * (x - 9) / (8 - 9) * (x - 10) / (8 - 10) * (x - 11) / (8 - 11) * (x - 12) / (8 - 12) * (x - 13) / (8 - 13) * (x -
14) / (8 - 14) * (x - 15) / (8 - 15) * (x - 16) / (8 - 16) + 59296646043258910 * (x - 1) / (9 - 1) * (x - 2) / (9 - 2) * (x - 3) / (9 - 3) * (x - 4) /
(9 - 4) * (x - 5) / (9 - 5) * (x - 6) / (9 - 6) * (x - 7) / (9 - 7) * (x - 8) / (9 - 8) * (x - 10) / (9 - 10) * (x - 11) / (9 - 11) * (x - 12) / (9 -
12) * (x - 13) / (9 - 13) * (x - 14) / (9 - 14) * (x - 15) / (9 - 15) * (x - 16) / (9 - 16) + 350000000000000000 * (x - 1) / (10 - 1) * (x - 2) / (10 -
2) * (x - 3) / (10 - 3) * (x - 4) / (10 - 4) * (x - 5) / (10 - 5) * (x - 6) / (10 - 6) * (x - 7) / (10 - 7) * (x - 8) / (10 - 8) * (x - 9) / (10 - 9) *
(x - 11) / (10 - 11) * (x - 12) / (10 - 12) * (x - 13) / (10 - 13) * (x - 14) / (10 - 14) * (x - 15) / (10 - 15) * (x - 16) / (10 - 16) +
1746089734815742200 * (x - 1) / (11 - 1) * (x - 2) / (11 - 2) * (x - 3) / (11 - 3) * (x - 4) / (11 - 4) * (x - 5) / (11 - 5) * (x - 6) / (11 - 6) * (x
- 7) / (11 - 7) * (x - 8) / (11 - 8) * (x - 9) / (11 - 9) * (x - 10) / (11 - 10) * (x - 12) / (11 - 12) * (x - 13) / (11 - 13) * (x - 14) / (11 - 14) *
(x - 15) / (11 - 15) * (x - 16) / (11 - 16) + 7580254614696493000 * (x - 1) / (12 - 1) * (x - 2) / (12 - 2) * (x - 3) / (12 - 3) * (x - 4) / (12 - 4) *
(x - 5) / (12 - 5) * (x - 6) / (12 - 6) * (x - 7) / (12 - 7) * (x - 8) / (12 - 8) * (x - 9) / (12 - 9) * (x - 10) / (12 - 10) * (x - 11) / (12 - 11) *
(x - 13) / (12 - 13) * (x - 14) / (12 - 14) * (x - 15) / (12 - 15) * (x - 16) / (12 - 16) + 29278330804059914000 * (x - 1) / (13 - 1) * (x - 2) / (13 -
2) * (x - 3) / (13 - 3) * (x - 4) / (13 - 4) * (x - 5) / (13 - 5) * (x - 6) / (13 - 6) * (x - 7) / (13 - 7) * (x - 8) / (13 - 8) * (x - 9) / (13 - 9) *
(x - 10) / (13 - 10) * (x - 11) / (13 - 11) * (x - 12) / (13 - 12) * (x - 14) / (13 - 14) * (x - 15) / (13 - 15) * (x - 16) / (13 - 16) +
102363806877040440000 * (x - 1) / (14 - 1) * (x - 2) / (14 - 2) * (x - 3) / (14 - 3) * (x - 4) / (14 - 4) * (x - 5) / (14 - 5) * (x - 6) / (14 - 6) *
(x - 7) / (14 - 7) * (x - 8) / (14 - 8) * (x - 9) / (14 - 9) * (x - 10) / (14 - 10) * (x - 11) / (14 - 11) * (x - 12) / (14 - 12) * (x - 13) / (14 -
13) * (x - 15) / (14 - 15) * (x - 16) / (14 - 16) + 328420417785644500000 * (x - 1) / (15 - 1) * (x - 2) / (15 - 2) * (x - 3) / (15 - 3) * (x - 4) /
(15 - 4) * (x - 5) / (15 - 5) * (x - 6) / (15 - 6) * (x - 7) / (15 - 7) * (x - 8) / (15 - 8) * (x - 9) / (15 - 9) * (x - 10) / (15 - 10) * (x - 11) /
(15 - 11) * (x - 12) / (15 - 12) * (x - 13) / (15 - 13) * (x - 14) / (15 - 14) * (x - 16) / (15 - 16) + 977677435906606200000 * (x - 1) / (16 - 1) * (x
- 2) / (16 - 2) * (x - 3) / (16 - 3) * (x - 4) / (16 - 4) * (x - 5) / (16 - 5) * (x - 6) / (16 - 6) * (x - 7) / (16 - 7) * (x - 8) / (16 - 8) * (x -
9) / (16 - 9) * (x - 10) / (16 - 10) * (x - 11) / (16 - 11) * (x - 12) / (16 - 12) * (x - 13) / (16 - 13) * (x - 14) / (16 - 14) * (x - 15) / (16 - 15)
```

Для демонстрації наближенності апроксимації до справжнього результату розрахунків програма будує графік апроксімації для x[1, 45] (результати функції від більших чисел виходять за рамки 64-бітних чисел з плаваючою точкою) й записує ці значення в файл lagrange.png . Прошу зауважити, що значення ближчі до x=1 прораховані, але не намальовані через низький розмір зображення, який я не знайшов як збільшити в gnuplot (я радий, що воно взагалі малює). Також програма записує значення в файл lagrange.txt, де можна переглянути розрахунки в текстовому вигляді.

Для порівняння зліва - файл onlineGraph.png (намальований за допомогою сторонніх ресурсів графік функції), зправа - графік апроксимації lagrange.png:



## 2. Побудувати лінійний інтерполяційний сплайн.

Код: /linearspline

Для початкових значень було прораховано значення функції на цілих значеннях проміжку x[1, 6] і потім побудовано лінійний інтерполяційний сплайн на цьому ж проміжку для інших 100004 x за формулою:

$$y = \frac{y_{i+1}-y_i}{x_{i+1}-x_i}(x - x_i) + y_i$$

Текстові результати обчислень знаходяться в linearSpline.txt, графік сплайну є в splineGraph.png (нижче - зправа) й побудований онлайн графік функції в builtGraph.png (нижче - зліва). Так само в графіку gnuplot, нажаль, не видно значень ближчіх до 1 але тепер порівняння із онлайн графіком більш чесне, бо на тому скріншоті не видно координатну вісь.