Algorytmy numeryczne, część 2

dr Marcin Ziółkowski

Instytut Matematyki i Informatyki
Akademia im. Jana Długosza w Częstochowie

16 maja 2017 r.

CAŁKOWANIE NUMERYCZNE

W wielu praktycznych zagadnieniach mamy do czynienia z obliczaniem całek oznaczonych. Z matematycznego punktu widzenia wiadomo, że całkę oznaczoną można obliczać ze wzoru Newtona-Leibnitza:

$$\int_a^b f(x)dx = F(b) - F(a),$$

gdzie $F(x)$ jest dowolną funkcją pierwotną dla funkcji $f(x)$. Niestety istnieje duża grupa funkcji, dla których znalezienie funkcji pierwotnej nie jest możliwe i nie możemy tego wzoru wykorzystać do obliczania całek. Do takich całek należą między innymi:

$$\int_a^b \frac{\sin x}{x}dx, \int_a^b \frac{\cos x}{x}dx, \int_a^b \frac{\log x}{x}dx, \int_a^b e^{-x^2}dx$$

$$\int_a^b \cos(\frac{\pi x^2}{2})dx, \int_a^b \sin(\frac{\pi x^2}{2})dx, \int_a^b \sin(\sin x)dx, \int_a^b \cos(\cos x)dx$$

CAŁKOWANIE NUMERYCZNE

Do obliczania tego typu całek stosuje się metody numeryczne, które w ogólnym przypadku polegają na zastąpieniu funkcji podcałkowej inną funkcją (najlepiej funkcją będącą sklejeniem wielomianów), która z jednej strony w dobry sposób przybliża daną funkcję, a z drugiej strony jest kawałkami całkowalna z wykorzystaniem wzoru Newtona-Leibnitza. Najczęściej wykorzystujemy przybliżanie funkcją kawałkami liniową (metoda prostokątów oraz metoda trapezów) oraz funkcją kawałkami kwadratową (metoda Simpsona). Okazuje się że wybór wielomianów wyższego stopnia niekoniecznie daje lepsze wyniki. W związku z tym faktem te trzy metody są stosowane najczęściej.

METODA PROSTOKĄTÓW

Metoda prostokątów polega na zastąpieniu funkcji podcałkowej funkcją kawałkami stałą. Wówczas zamiast obliczać całkę $\int_a^b f(x)dx$ obliczamy sumę pól powstałych prostokątów. Dzielimy przedział $[a, b]$ na podprzedziały o długości h, a następnie obliczamy następującą sumę, która jest przybliżeniem danej całki:

$$\sum_{i=0}^{a+ih<b} f(a+ih) \cdot h \approx \int_a^b f(x)dx$$

METODA PROSTOKĄTÓW

Napiszemy teraz program w języku Python, który z wykorzystaniem metody prostokątów oblicza całkę $\int_0^{\pi} \sin x dx$.

PRZYKŁAD 1. METODA PROSTOKĄTÓW

```
import math
def fun(x):
    return math.sin(x)
def main():
    a=0
    b=math.pi
    i=0
    h=math.pi/30
    calka=0
    while a+i*h<b:
        calka=calka+fun(a+i*h)*h
        i=i+1
    print("Wartosc calki wynosi", calka)
    input()
main()
```

METODA TRAPEZÓW

Wartość obliczona według tego programu wynosi 1.9982. Rzeczywista wartość to 2. Oczywiście można jeszcze poprawić dokładność zmniejszając wartość h.
Metoda trapezów polega na zastąpieniu funkcji podcałkowej funkcją kawałkami liniową. Wówczas zamiast obliczać całkę $\int_a^b f(x)dx$ obliczamy sumę pól powstałych trapezów. Dzielimy przedział $[a, b]$ na podprzedziały o długości h, a następnie obliczamy następującą sumę, która jest przybliżeniem danej całki:

$$\sum_{i=0}^{a+ih<b} 0,5(f(a+ih)+f(a+(i+1)h))\cdot h \approx \int_a^b f(x)dx$$

METODA TRAPEZÓW

Napiszemy teraz program w języku Python, który z wykorzystaniem metody trapezów oblicza całkę $\int_0^{\pi} \sin x dx$.

PRZYKŁAD 2. METODA TRAPEZÓW

```
import math
def fun(x):
    return math.sin(x)
def main():
    a=0
    b=math.pi
    i=0
    h=math.pi/30
    calka=0
    while a+i*h<b:
        calka=calka+0.5*(fun(a+i*h)+fun(a+(i+1)*h))*h
        i=i+1
    print("Wartosc calki wynosi", calka)
    input()
main()
```

METODA SIMPSONA

Wartość obliczona według tego programu wynosi 1.9982. Rzeczywista wartość to 2. Oczywiście można jeszcze poprawić dokładność zmniejszając wartość h.
Metoda Simpsona (uważana za jedną z najlepszych metod całkowania numerycznego) polega na zastąpieniu funkcji podcałkowej funkcją będącą sklejeniem odcinków parabol. Wówczas zamiast obliczać całkę $\int_a^b f(x)dx$ obliczamy sumę pól powstałych trapezów krzywoliniowych opartych na odcinkach parabol. Dzielimy przedział $[a, b]$ na podprzedziały o długości h, a następnie obliczamy następującą sumę, która jest przybliżeniem danej całki:

$$\sum_{i=0}^{a+ih<b} \frac{1}{6}(f(a+ih)+4f(a+(i+0,5)h)+f(a+(i+1)h))\cdot h \approx \int_a^b f(x)dx$$

METODA SIMPSONA

Napiszemy teraz program w języku Python, który z wykorzystaniem metody Simpsona oblicza całkę $\int_0^{\pi} \sin x dx$.

PRZYKŁAD 3. METODA SIMPSONA

```
import math
def f(x):
    return math.sin(x)
def main():
    a=0
    b=math.pi
    i=0
    h=math.pi/30
    calka=0
    while a+i*h<b:
        calka=calka+1/6*(f(a+i*h)+4*f(a+(i+0.5)*h)+f(a+(i+1)*h))*h
        i=i+1
    print("Wartosc calki wynosi", calka)
    input()
main()
```

Wartość obliczona według tego programu wynosi 2.0000000835. Omówimy jeszcze jedną metodę całkowania numerycznego. Ideę tej metody zaproponował polski matematyk Stanisław Ulam. Omówimy metodę Monte Carlo w przypadku, gdy funkcja podcałkowa jest ściśle monotoniczna w przedziale $[a, b]$ (rosnąca lub malejąca, w innych sytuacjach przedział całkowania dzielimy na podprzedziały). Idea metody w takim przypadku jest następująca: Losujemy N punktów należących do prostokąta $[a, b] \times [0, max(f(a), f(b)]$. Niektóre n z tych punktów leżą poniżej krzywej będącej wykresem funkcji $y = f(x)$. Ułamek $\frac{n}{N}$ może być rozumiany jako estymator prawdopodobieństwa, że wylosowany punkt leży poniżej krzywej. Z teorii prawdopodobieństwa wynika, że mamy zależność:

$$\frac{n}{N} \approx \frac{\int_a^b f(x)dx}{(b-a) \cdot max(f(a), f(b))}$$

A stąd:

$$\int_a^b f(x)dx \approx \frac{n}{N}(b-a) \cdot max(f(a), f(b))$$

METODA MONTE CARLO

Pokażemy istotę tej metody na kolejnym przykładzie. Napiszemy program, który oblicza całkę $\int_2^3 x^3 dx$.

PRZYKŁAD 4. METODA MONTE CARLO

```
import random
def f(x):
    return x*x*x
def main():
    N=1000
    a=2
    b=3
    n=0
    for i in range (N):
        x=random.uniform(a,b)
        y=random.uniform(0,f(b))
        if y<f(x):
            n=n+1
    print("Wartosc calki to:", (n/N)*(b-a)*f(b))
    input()
main()
```

Metoda ta nie należy do najdokładniejszych. Wynik ostatnich obliczeń to: $16,632$ (rzeczywista wartość $16,25$). Oczywiście dokładność można zwiększyć zmieniając N, lecz pojawiają się problemy związane z okresowością generatorów losowych. Ma jednak tą zaletę że można za pomocą tej metody obliczać również skomplikowane całki wielowymiarowe.

NUMERYCZNE ROZWIĄZYWANIE UKŁADÓW RÓWNAŃ LINIOWYCH

Zajmiemy się teraz problemem rozwiązywania układów równań liniowych, a dokładniej układami kramerowskimi tzn. takimi, w których liczba równań i niewiadomych jest taka sama, a wyznacznik główny układu ($detA$) jest różny od zera. Takie układy równań posiadają tylko jedno rozwiązanie, które wyznaczamy ze znanych wzorów Cramera ($x_i = \frac{detX_i}{detA}$) lub stosując odwracanie macierzy (każdy taki układ może być zapisany w postaci macierzowej $AX = B$, a jego rozwiązanie jest następujące: $X = A^{-1}B$). Niestety obie metody są dość czasochłonne, gdy niewiadomych jest dużo, a liczba operacji silnie wzrasta wraz ze zwiększaniem się wymiaru macierzy A (złożoność rzędu $O(n!)$, czyli silnie wykładnicza !). Metodą stosowaną najczęściej i zdecydowanie szybszą jest METODA ELIMINACJI GAUSSA, którą przypomnimy na kolejnym przykładzie.

PRZYKŁAD 5.

Rozwiążemy poniższy układ: $\begin{cases} x_1 + x_2 + x_3 + x_4 = 10 \\ 2x_1 + 3x_2 + 3x_3 + 3x_4 = 29 \\ x_1 - x_2 - x_3 - 2x_4 = -12 \\ 3x_1 + 4x_2 + x_3 + 5x_4 = 34 \end{cases}$

Pierwsze równanie mnożymy odpowiednio przez -2, -1 oraz -3 i dodajemy do kolejnych równań otrzymując układ:

$$\begin{cases} x_1 + x_2 + x_3 + x_4 = 10 \\ \quad x_2 + x_3 + x_4 = 9 \\ \quad -2x_2 - 2x_3 - 3x_4 = -22 \\ \quad x_2 - 2x_3 + 2x_4 = 4 \end{cases}$$

PRZYKŁAD 5. c.d

Teraz drugie równanie mnożymy odpowiednio przez 2 oraz -1 i dodajemy do trzeciego i czwartego równania oraz zamieniamy trzecie i czwarte równanie, otrzymując układ:

$$\begin{cases} x_1 + x_2 + x_3 + x_4 = 10 \\ \quad x_2 + x_3 + x_4 = 9 \\ \quad -3x_3 + x_4 = -5 \\ \quad -x_4 = -4 \end{cases}$$

Teraz postępujemy w taki sposób, aby od końca wyznaczyć kolejno wartości wszystkich zmiennych (metoda eliminacji Gaussa nazywana jest czasem uogólnioną metodą podstawiania):

$$\begin{cases} x_1 = 10 - x_2 - x_3 - x_4 = 1 \\ \quad x_2 = 9 - x_3 - x_4 = 2 \\ \quad x_3 = (-5 - x_4) : (-3) = 3 \\ \quad x_4 = (-4) : (-1) = 4 \end{cases}$$

PROBLEMY I ZADANIA

Metoda eliminacji Gaussa ma mniejszą złożoność obliczeniową i jest bardzo wygodna do pisania programów rozwiązujących układy równań. Zajmiemy się tym zagadnieniem na ćwiczeniach.

PROBLEM 1.

Napisać w języku PYTHON program, który oblicza wartość całki:

$$\int_0^2 e^{-x^2} dx$$

w oparciu o metodę prostokątów, trapezów oraz Simpsona. Porównać otrzymane wyniki.

PROBLEM 2.

Napisać w języku PYTHON program, który wyznacza przybliżoną wartość liczby π w oparciu o metodę Monte Carlo.
Wskazówka: Rozpatrzeć funkcję $f(x) = \sqrt{1 - x^2}$ w przedziale $[-1, 1]$ (pole pod wykresem tej funkcji wynosi $\frac{\pi}{2}$).

PROBLEM 3.

Użytkownik wprowadza liczbę n będącą liczbą niewiadomych kramerowskiego układu równań, a następnie wszystkie jego współczynniki. Napisać program, który w oparciu o metodę eliminacji Gaussa rozwiązuje dany układ. Wykorzystać listy jedno lub wielowymiarowe.