

## 7 Obliczanie całek (wskaźniki do funkcji)

### Całka a kwadratura.

Obliczanie całki zastępujemy obliczaniem kwadratury – numerycznego przybliżenia wartości całki Riemanna. Do najbardziej elementarnych metod obliczenia przybliżenia  $Q$  całki  $\int_a^b f(x)dx$  należą kwadratury (wzory):

1. prostokątów

$$Q_R = (b - a)f(c), \quad c \in [a, b],$$

w tym

- prostokątów w przód (lewostronna), gdy  $c = a$ ,
- prostokątów wstecz (prawostronna), gdy  $c = b$ ,
- prostokątów punktu środkowego (centralna), gdy  $c = (a + b)/2$ .

2. trapezów

$$Q_T = \frac{b - a}{2}[f(a) + f(b)],$$

3. Simpsona

$$Q_s = \frac{b - a}{6}[f(a) + 4f(c) + f(b)], \quad c = (a + b)/2.$$

W praktyce stosuje się je w wersjach złożonych (zadania 7.1.1 i 7.1.2) lub w algorytmach adaptacyjnych (7.1.3)

### 7.1 Całki jednokrotne – Kwadratury złożone

Złożona kwadratura wybranego typu polega na podziale przedziału całkowania  $[a, b]$  na  $n$  równych podprzedziałów o długości  $h = (b - a)/n$  i zsumowanie kwadratur prostych tego typu obliczonych dla każdego podprzedziału. Np. złożona kwadratura prostokątów w przód (leftpoint) jest sumą

$$C_{R_{left}} = h \sum_{i=0}^{n-1} f(a + ih).$$

#### 7.1.1 Funkcje z wskaźnikiem na funkcję podcałkową

Szablon programu należy uzupełnić o:

1. definicje funkcji (procedur) obliczających wartości przykładowych funkcji podcałkowych
  - `f_poly(double x)`, która zwraca wartość wielomianu  $2x^5 - 4x^4 + 3.5x^2 + 1.35x - 6.25$ ,
  - `f_rat(double x)`, która zwraca wartość funkcji  $f(x) = \frac{1}{(x-0.5)^2+0.01}$ ,

- `f_exp(double x)`, która zwraca wartość funkcji  $f(x) = 2xe^{-1.5x} - 1$ ,
  - `f_trig(double x)`, która zwraca wartość funkcji  $f(x) = x \operatorname{tg}(x) - 1$
2. definicję nazwy `typedef ...Func1vFp...`; – typu wskaźnikowego do funkcji z jednym parametrem typu `double` i zwracającą wartość typu `double`.
  3. definicje funkcji obliczających złożoną kwadraturę dla funkcji `f` z podziałem przedziału całkowania  $[a, b]$  na `n` podprzedziałów
    - prostokątów w przód (`leftpoint`) – `quad_rect_left(...)`,
    - prostokątów wstecz (`rightpoint`) – `quad_rect_right(...)`,
    - prostokątów punktu środkowego (`midpoint`) – `quad_rect_mid(...)`,
    - trapezów – `quad_trap(...)`,
    - Simpsona – `quad_simpson(...)`.

W ostatnich dwóch kwadraturach należy unikać dwukrotnego obliczania wartości funkcji podcałkowej dla tego samego argumentu.

Każda z tych funkcji ma 4 argumenty:

- (a) wskaźnik do funkcji obliczającej wartość funkcji podcałkowej `f`,
  - (b) dolną granicę całkowania `a`,
  - (c) górną granicę całkowania `b`,
  - (d) liczbę podprzedziałów `n`.
4. definicję nazwy `typedef ...QuadratureFp...` – typu wskaźnikowego do funkcji obliczających wartości kwadratury.

### Tablice wskaźników na funkcje

Szablon programu należy uzupełnić o definicję funkcji

`double quad_select(int quad_no, int fun_no, double a, double b, int n)`, która w przedziale  $[a, b]$  oblicza kwadraturę złożoną wskazaną indeksem `quad_no` (z podziałem na `n` podprzedziałów) dla funkcji podcałkowej wskazanej indeksem `fun_no`. Na zewnątrz funkcji `quad_select()` jest definiowana tablica wskaźników do funkcji typu `Func1vFp` oraz tablica typu `QuadratureFp`. Obie tablice są inicjowane wskaźnikami do funkcji zdefiniowanych w punkcie 7.1.1.

### Test 1

Wczytuje granice przedziału całkowania oraz liczbę podprzedziałów i 20 razy wywołuje funkcję `quad_select()`. Dwa pierwsze argumenty tej funkcji są parą iloczynu kartezjańskiego zbioru indeksów tablicy wskaźników do kwadratur `quad_tab` i zbioru indeksów tablicy wskaźników do funkcji podcałkowych `func_tab`.

### • Wejście

Nr testu

granice przedziału całkowania, liczba podprzedziałów

- **Wyjście**  
20 wartości każdej kwadratury dla każdej funkcji
- **Przykład:**  
Wejście:  
1  
0 0.75 25  
Wyjście:  
-3.97887 25.47947 0.14924 -0.48180  
-3.91316 25.77788 0.17020 -0.46719  
-3.94620 25.64102 0.15945 -0.47427  
-3.94602 25.62868 0.15972 -0.47450  
-3.94614 25.63690 0.15954 -0.47434

### 7.1.2 Algorytm adaptacyjny w wersji rekurencyjnej

W algorytmie jest stosowana jedna, elementarna kwadratura w wersji podstawowej (tzn. nie złożonej). Na każdym etapie obliczeń wyznaczamy przybliżoną wartość  $S$  całki z funkcji  $f$  w pewnym podprzedziale przedziału  $[a, b]$  wg podstawowego wzoru wybranej kwadratury.

Błąd przybliżenia wartości całki kwadraturą jest mały jeżeli długość  $h$  podprzedziału, w którym jest obliczana całka, jest mała. Algorytm adaptacyjny ma na celu osiągnięcie wyniku (przybliżonej wartości całki) z błędem bezwzględnym nie większym niż zadany  $= \Delta$  skracając długości podprzedziałów tylko tam, gdzie to jest konieczne.

Pierwsze przybliżenie całki jest obliczane dla całego przedziału  $[a, b]$ . Następnie ten przedział jest dzielony na 2 połowy. Wzór podstawowy wybranej kwadratury jest teraz stosowany osobno dla lewej połówki (od  $a$  do  $c = (a + b)/2$ ) i dla prawej (od  $c$  do  $b$ ). Otrzymujemy przybliżone wartości dwóch części obliczanej całki -  $S_1$  i  $S_2$ . Jeżeli suma  $S_1 + S_2$  różni się od  $S$  nie więcej niż o  $\Delta$ , to uznajemy, że otrzymany wynik jest dostatecznie dokładny i kończymy algorytm. W przeciwnym przypadku stajemy przed dwoma zadaniami - obliczyć całkę na dwóch połówkach (lewej i prawej), każdej z błędem nie większym niż  $\Delta/2$ . Zauważmy, że są to jakościowo dokładnie dwa takie same zadania, jak zadanie pierwotne (obliczyć całkę z błędem nie większym niż zadany).

Należy tak napisać program, aby liczba obliczeń wartości zadanej funkcji była jak najmniejsza - aby nie obliczać dwukrotnie funkcji dla tego samego argumentu. W tym celu, obliczona na danym poziomie rekurencji wartość kwadratury jest przekazywana przez parametry na kolejny poziom.

W funkcji rekurencyjnej powinna być kontrola poziomu rekursji. Załóżmy, że jeżeli maksymalny zadany poziom rekursji został osiągnięty, a błąd nadal przekracza dopuszczalną granicę `RECURS_LEVEL_MAX` (stała zdefiniowana w programie), to wynikiem obliczeń będzie symbol nieoznaczony NaN.

UWAGA: Ten algorytm nie gwarantuje wyniku w granicach zadanego błędu - możliwe jest przekroczenie zadanego dopuszczalnego błędu (nie trudno podać przykład takiej "złośliwej" funkcji). Dlatego w praktyce obliczeniowej stosuje

się dodatkowe zabezpieczenia, które tu - dla uproszczenia zadania - nie są proponowane.

Wartości startowe rekurencji (wartość  $S$  kwadratury obliczona na całym przedziale  $[a, b]$  i początkowy poziom rekurencji) są ustalane we wstępnej procedurze `init_recurs()`.

Szablon programu należy uzupełnić o definicję funkcji inicjującej

```
double init_recurs(Func1vFp f,double a,double b,double delta, QuadratureFp
quad)
oraz funkcji wywoływanej rekurencyjnie
double recurs(Func1vFp f,double a,double b,double S,double delta, QuadratureFp
quad,int level).
```

### Test 2

Wczytanie danych i wywołanie funkcji `double init_recurs`. Całkowana funkcja: wybrana z tablicy wskaźników do funkcji `func_tab`.

Kwadratura: wybrana z tablicy kwadratur `quad_tab`.

- **Wejście**  
Nr testu  
indeks funkcji podcałkowej, indeks kwadratury  
granice całkowania, dopuszczalny błąd bezwzględny
- **Wyjście**  
Wartość kwadratury
- **Przykład:**  
Wejście:  
2  
1 4  
0 3 0.01  
Wyjście:  
29.04248

## 7.2 Całka podwójna po powierzchni (*surface integral*)

W tym podrozdziale będą obliczane wartości funkcji dwóch zmiennych. Należy zdefiniować nazwę

```
typedef ...Func2vFp...;
```

typu wskaźnikowego do funkcji z dwoma parametrami typu `double` zwracającej wartość typu `double`.

### 7.2.1 Całka po obszarze prostokątnym

Obszar całkowania funkcji  $f(x, y)$  można zapisać w postaci

$$R = \{(x, y) : x_1 \leq x \leq x_2, y_1 \leq y \leq y_2\},$$

Szablon programu należy uzupełnić o definicję funkcji  
`double dbl_integr(Func2vFp f, double x1, double x2, int nx, double y1, double y2, int ny),`  
która złożoną metodą prostokątów w przód (leftpoint) oblicza przybliżoną wartość całki. Parametry `nx` i `ny` są liczbami podprzedziałów kwadratur złożonych.

$$V = \int_{y_1}^{y_2} \left( \int_{x_1}^{x_2} f(x, y) dx \right) dy \quad (1)$$

### Test 3

Wczytanie danych `x1`, `x2`, `nx`, `y1`, `y2`, `ny` i wywołanie funkcji.

`dbl_integr(f,x1,x2,nx,y1,y2,ny)`

Całkowana funkcja `f`: `func2v_2`.

- **Wejście**  
Nr testu  
`x1`, `x2`, `nx`, `y1`, `y2`, `ny`
- **Wyjście**  
Wartość kwadratury
- **Przykład:**  
Wejście:  
3  
0 1 100  
0 1 100  
Wyjście:  
1.42662

## 7.2.2 Całka po obszarze normalnym

Obszar postaci

$$D = \{(x, y) : x_1 \leq x \leq x_2, g(x) \leq y \leq h(x)\},$$

gdzie funkcje  $g(x)$  i  $h(x)$  są ciągłe na odcinku  $[x_1, x_2]$ , oraz  $g(x) < h(x)$  we wnętrzu tego odcinka, nazywamy obszarem normalnym względem osi  $Ox$ .

Użyteczny link:

[https://pl.wikipedia.org/wiki/CaĆka\\_podwjna](https://pl.wikipedia.org/wiki/Ca%C5%82ka_podwjna), część: Zamiana na całkę iterowaną.

Wtedy wzór (1) można zapisać w postaci

$$V_n = \int_{x_1}^{x_2} \left( \int_{g(x_1)}^{h(x_2)} f(x, y) dy \right) dx$$

Szablon programu należy uzupełnić o definicję funkcji  
`double dbl_integr_normal_1(Func2vFp f, double x1, double x2, int nx, double hy, Func1vFp fg, Func1vFp fh).`

Parametr `hy` jest przybliżoną długością podprzedziału kwadratury złożonej zastosowanej do całkowania wzdłuż zmiennej  $y$ . Służy do wyznaczenia liczby podprzedziałów  $n_y$  – najmniejszej liczby całkowitej, nie mniejszej od  $\frac{h(x_i) - g(x_i)}{h_y}$ .

#### Test 4

Wczytanie danych `x1`, `x2`, `nx`, `hy` i wywołanie funkcji

`dbl_integr_normal_1(f,x1,x2,nx,hy,fg,fh)`

Całkowana funkcja `f`: `func2v_2`.

Funkcje ograniczające obszar całkowania `fg` i `fh`: `lower_bound_2`, `upper_bound_2`.

- **Wejście**

Nr testu

`x1`, `x2`, `nx`, `y1`, `y2`, `ny`

- **Wyjście**

Wartość kwadratury

- **Przykład:**

Wejście:

4

0.7 0.9 200

1e-3

Wyjście:

0.14480

### 7.2.3 Całka po (wielu) obszarach normalnych wewnątrz prostokąta

Rozważmy przypadek obszaru całkowania bardziej ogólnego niż obszar normalny, gdy warunek  $g(x) \leq h(x)$  nie jest spełniony dla każdego  $x \in [a, b]$ . Rysunek 1 jest wykresem funkcji  $f(x, y)$  całkowanej we wszystkich (dwóch) podobszarach prostokąta

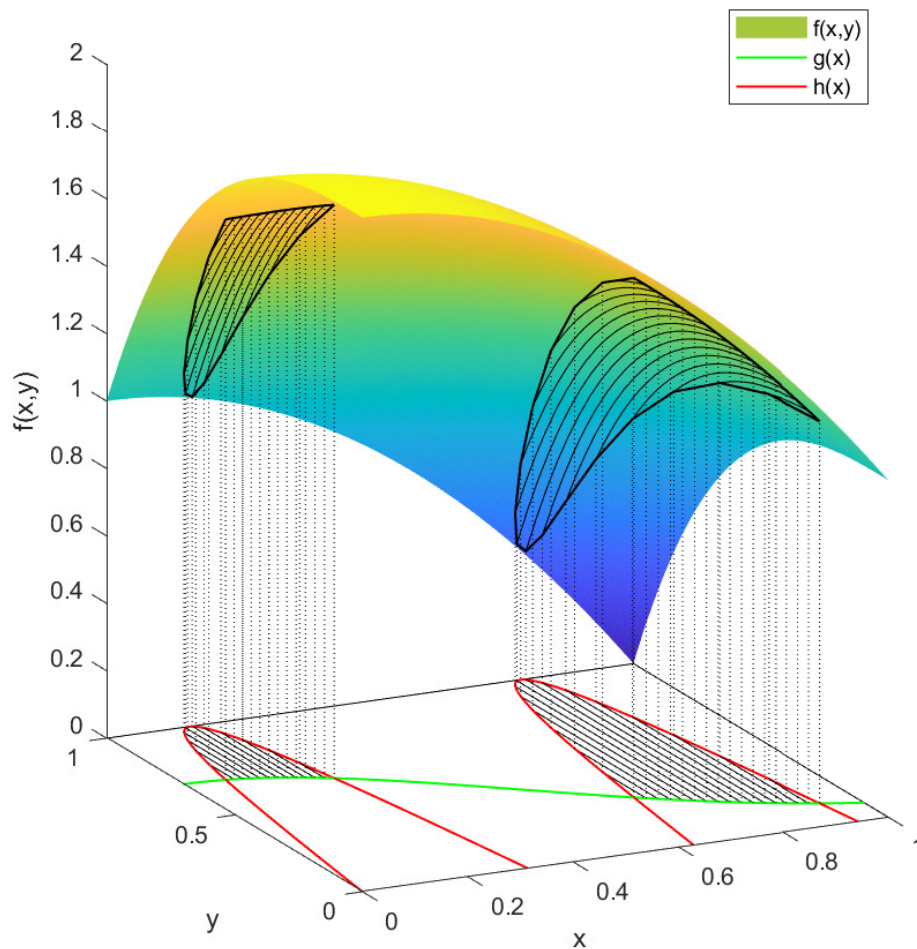
$$R = \{(x, y) : a \leq x \leq b, \quad c \leq y \leq d\},$$

w których  $g(x) < h(x)$  – obszary te są na rysunku oznaczone kreskowaniem kolorem czarnym. Każdy z nich jest obszarem normalnym względem osi  $Ox$ . Użycie algorytmu z punktu 7.2.2 wymagałoby wyznaczania wszystkich pierwiastków równania  $g(x) = h(x)$ .

W tym zadaniu należy zastosować prostszy algorytm całkowania – całkowania po obszarze prostokątnym  $R$  z predykatem orzekającym, czy dla danej wartości  $x_i$  zachodzi nierówność  $g(x_i) < y < h(x_i)$ . Jeżeli tak, to należy obliczyć całkę (a dokładniej – kwadraturę)

$$Q_i(x_i) \approx \int_{\max(y_1, g(x_i))}^{\min(y_2, h(x_i))} f(x_i, y) dy$$

stosując jedną z kwadratur złożonych.



Rysunek 1: Przykład zadania całkowania po dwóch obszarach normalnych względem osi  $0x$ .  $f(x,y) = 2 - x^2 - y^3$ ,  $g(x) = 0.7 \exp(-2x^2)$ ,  $h(x) = \sin(10x)$ .

Szablon programu należy uzupełnić o definicję funkcji

```
dbl.integr.normal.n(Func2vFp f, double x1, double x2, int nx, double
y1, double y2, int ny, Func1vFp fg, Func1vFp fh)),
```

która złożoną metodą prostokątów leftpoint oblicza przybliżoną objętość pod wykresem funkcji  $f$  nad obszarem normalnym ograniczonym wartościami  $x_1$ ,  $x_2$ , i funkcjami  $h(x)$  i  $g(x)$  oraz prostokątem  $x_1, y_1, x_2, y_2$ .

Liczbę podprzedziałów kwadratur  $Q_i(x_i)$  należy wyznaczyć podobnie jak w punkcie 7.2.2 przyjmując  $h_y = (y_2 - y_1)/n_y$ .

#### Test 5

Wczytanie danych `x1, x2, nx, y1, y2, ny` i wywołanie funkcji

`dbl_integr_normal_n(f, x1, x2, nx, y1, y2, ny, fg, fh)`

Całkowana funkcja `f`: `func2v_2`.

Funkcje ograniczające obszar całkowania `fg` i `fh`: `lower_bound_2, upper_bound_2`.

- **Wejście**

Nr testu

`x1, x2, nx, y1, y2, ny`

- **Wyjście**

Wartość kwadratury

- **Przykład:**

Wejście:

5

0 1 1000

0 1 1000

Wyjście:

0.21668

### 7.3 Całki potrójne – z predykatem wyłączającym część obszaru całkowania

**Uniwersalny typ funkcji (procedury) obliczającej wartość funkcji  $n$  zmiennych.**

Procedura obliczająca wartości funkcji  $n$  zmiennych wymaga przekazania do niej  $n$  wartości zmiennych niezależnych. Aby uniknąć konieczności definiowania kolejnych nazw typów dla kolejnych liczb zmiennych, wartości zmiennych niezależnych będziemy przekazywać poprzez  $n$ -elementową tablicę.

Przykładowa funkcja podcałkowa trzech zmiennych jest zdefiniowana w szablonie programu `double func3v(const double v[], int n)`. Nazwa typu wskaźnika do takiej procedury (funkcji) jest w szablonie programu zdefiniowana:

```
typedef double (*FuncNvFp)(const double*, int);
```

**Predykat wykluczający dany punkt z obszaru całkowania.** Zdefiniowana w szablonie funkcja `int bound3v(const double v[], int n)` jest predykatem zwracającym 1 gdy punkt o współrzędnych zapisanych w tablicy `v` leży wewnątrz obszaru całkowania.

Nazwa typu wskaźnika do procedury – predykatu – zwracającej wartość logiczną warunku określającego, czy zadany punkt w przestrzeni  $n$ -wymiarowej należy do zadanego obszaru całkowania, jest zdefiniowana w linii:

```
typedef int (*BoundNvFp)(const double*, int)
```



### Całka potrójna po prostopadłościanie z predykatem boundary akceptującym albo wykluczającym elementarną domenę kwadratury

Szablon programu należy uzupełnić o definicję funkcji

```
double trpl_quad_rect(FuncNvFp f, int variable_no, const double variable_lim[][2],
const int tn[], BoundNvFp boundary),
```

która oblicza kwadraturę prostokątów wstecz (rightpoint) jako przybliżenie całki potrójnej po prostopadłościanie. Dolne i górne granice przedziałów całkowania wzdłuż kolejnych zmiennych są przekazywane w tablicy `variable_lim`, a liczby podprzedziałów – w tablicy `tn`. Parametr `boundary` jest adresem predykatu. Wartość `NULL` tego parametru oznacza brak predykatu, czyli brak ograniczeń w obszarze całkowania.

#### Test 6

Wczytanie danych `x1`, `x2`, `nx`, `y1`, `y2`, `ny` i wywołanie funkcji `trpl_quad_rect()`.

Całkowana funkcja: `func3v`.

Predykat: Funkcje ograniczające obszar całkowania `bound3v`.

- **Wejście**

Nr testu

`x1`, `x2`, `nx`

`y1`, `y2`, `ny`

`z1`, `z2`, `nz`

0 – gdy nie ma ograniczeń obszaru całkowania,

1 – gdy istnieje predykat.

- **Wyjście**

Wartość kwadratury

- **Przykład:**

Wejście:

6

0 1 20

0 1 20

0 1 20

1

Wyjście:

0.39666

## 7.4 Całka n-krotna z predykatem

W szablonie programu są zdefiniowane przykładowe funkcje  $n$  zmiennych: procedura obliczająca wartość funkcji podcałkowej  $n$ -wymiarowej

```
funcNv(const double v[], int n)
```

```
oraz predykat int boundNv(const double v[], int n).
```

Liczba zmiennych jest ograniczona zdefiniowaną stałą `N_MAX`.

### Całka po hiperprostopadłościanie z predykatem boundary

Szablon programu należy uzupełnić o definicję rekurencyjnej funkcji

```
recur_quad_rect_mid(double *psum, FuncNvFp f, int variable_no, double
tvariable[], const double variable_lim[][2], const int tn[], int level,
BoundNvFp boundary),
```

która oblicza przybliżenie całki z `variable_no` wymiarowej funkcji `f` po hiperprostopadłościanie określonym ograniczeniami zapisanymi w `variable_lim`. Wzdłuż `i`-tej zmiennej jest obliczana złożona kwadratura prostokątów punktu środkowego (midpoint) z podziałem na `tn[i]` podprzedziałów.

Parametr `level` można zastąpić zmienną statyczną zdefiniowaną wewnątrz funkcji.

Wartość kwadratury jest zapisywana pod adresem przekazywanym do funkcji przez jej pierwszy parametr.

#### Test 7

Wczytanie danych `x1`, `x2`, `nx`, `y1`, `y2`, `ny` i wywołanie funkcji `recur_quad_rect()`.

Całkowana funkcja: `funcNv`.

Predykat: `boundNv`.

- **Wejście**

Nr testu

Krotność całki  $n$

$n$  linii z przedziałem całkowania i liczbą podprzedziałów

0 – gdy bez ograniczeń obszaru całkowania,

1 – gdy istnieje predykat.

- **Wyjście**

Wartość kwadratury

- **Przykład:**

Wejście:

7

4

0 1 10

0 1 10

0 1 10

0 1 10

1

Wyjście:

0.98941