

Obliczenia numeryczne.

W pracach naukowców, konstruktorów i technologów często występuje konieczność obliczania różnorodnych wielkości za pomocą skomplikowanych zależności. Mimo, że wyprowadzone są teoretyczne wzory, z których można coś obliczyć, to okazuje się w praktyce, że nie jest to takie łatwe (np. czy nowowyprowadzony most zawali się, gdy siła wiatru przekracza 200 km/h). Zamiast bawić się skomplikowanymi matematycznymi zależnościami, układami równań itp., można przecież wykorzystać komputery i zamiast rozwiązywać układy równań, po prostu je wyliczyć. **Obliczenia numeryczne** polegają na obliczaniu wielkości zdefiniowanych za pomocą zależności matematycznych przy pomocy komputera.

Dokładność obliczeń i kumulowanie błędów.

Nie ma problemu, jeśli posługujemy się liczbami całkowitymi, które w komputerze pamiętane są dokładnie – wyniki są zawsze dokładne. Z liczbami rzeczywistymi (zwłaszcza okresowymi lub niewymiernymi) lub bardzo dużymi całkowitymi komputery radzą sobie gorzej, zwłaszcza, jeśli mamy do czynienia z dużymi dokładnościami. Liczby rzeczywiste pamiętane są tylko na kilku bajtach (na 6 - REAL) i dlatego niedokładnie (jak np. zapamiętać liczbę π ?). Liczby rzeczywiste po prostu zaokrągla się i pamiętane są **z określoną dokładnością**, a to powoduje błędy przy dokładnych obliczeniach. Innym komputerowym problemem liczb rzeczywistych jest **kumulowanie błędów**. Przy wielokrotnym powtarzaniu tych samych operacji na liczbach zmiennoprzecinkowych (rzeczywistych), zwłaszcza przy dodawaniu i odejmowaniu, sumowanie błędów powoduje zniekształcanie wyników obliczeń.

PRZYKŁAD LEKCJA121 – kumulowanie błędów

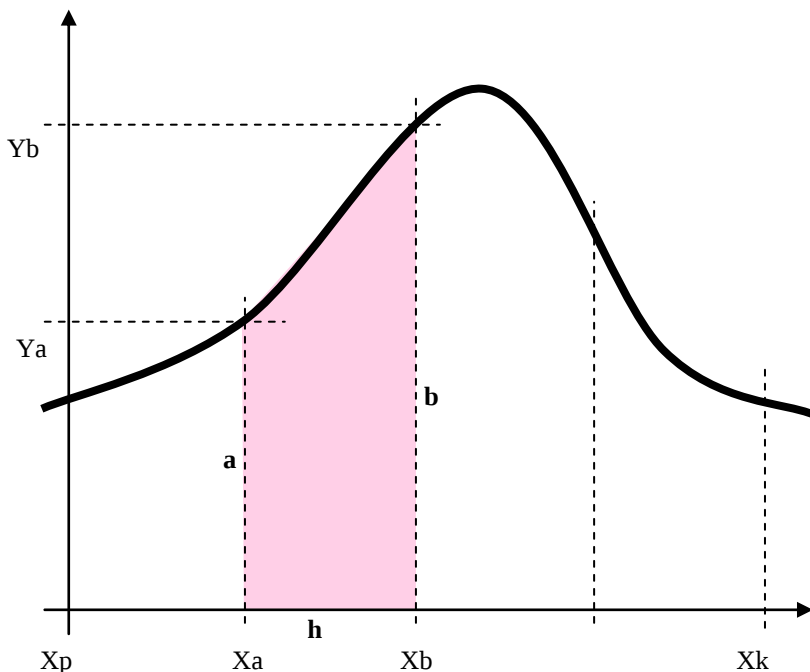
Zsumowanie dużą ilość razy tej samej liczby rzeczywistej (1,1) i podzielenie jej przez ilość sumowań powinno teoretycznie dać tą samą liczbę (czyli 1.1). Kumulowanie się błędów powstałych z zaokrąglania liczb rzeczywistych powoduje, że wyniki przy dużych ilościach są różne. Dla $r=1,1$ i $ile=10$ wynik wynosi 1,1 dla $ile=1000000$ wynik wynosi 1,0999999. Może te różnice nie są znaczne, ale przy obliczeniach wymagających bardzo dużej dokładności, taka dokładność jest niezbędna.

```
suma:=0;  
r:=1.1;  
for i:= 1 to ile do suma:=suma+r;  
writeln(suma/ile);
```

Całkowanie numeryczne (obliczanie pól powierzchni, objętości, długości linii ograniczonych krzywą).

Pole powierzchni

Obliczanie pola powierzchni polega na podzieleniu obszaru pod krzywą na niewielkie fragmenty (np. trapezy), dla których łatwo wyliczyć pole powierzchni. Im więcej trapezów tym dokładniejszy wynik. Jest to tzw. całkowanie numeryczne. Dla niektórych funkcji łatwo jest wyliczyć objętość, czy pole powierzchni za pomocą teoretycznych równań matematycznych (kwadrat, kula itp.), ale skomplikowane funkcje nie są możliwe do rozwiązania metodami teoretycznymi. Dlatego też całkowanie numeryczne ma ogromne znaczenie w wielu dziedzinach nauki i techniki.



Pole powierzchni

Chcemy obliczyć pole powierzchni pod wykresem, w przedziale $(X_p..X_k)$. Dzielimy ten przedział na jednakowej długości odcinki (im ich więcej, tym obliczenia będą dokładniejsze) w naszym przypadku na 4. Pole trapezu obliczamy z wzoru $P=h(a+b)/2$, na wykresie: $P=(X_b-X_a)(Y_b+Y_a)/2$ gdzie:

h - wysokość trapezu - skok na osi X (inaczej mówiąc, jaka jest odległość pomiędzy X_a , a X_b)

a - wartość Y obliczona dla X_a

b - wartość Y obliczona dla X_b

Jeśli zsumujemy pola wszystkich trapezów - otrzymamy (przybliżone) pole pod wykresem.

Jeśli wykres znajduje się pod osią OX - pola mają ujemną wartość, dlatego należy wyliczać je bez znaku (wartość bezwzględna, moduł itp).

Komputerowa realizacja – algorytm całkowania:

- wartość X_a jest równa X_p
- powtarzaj poniższe czynności
 - wylicz Y_a (wstawiając do wzoru na funkcję wartość X_a)
 - wylicz X_b (dodając do X_a SKOK na osi X)
 - wylicz Y_b (wstawiając do wzoru na funkcję wartość X_b)
 - oblicz pole trapezu (albo długość przeciwprostokątnej)
 - sumuj pola (długości)
 - wstaw do X_a wartość X_b
- aż X_b osiągnie wartość X_k
- zmniejsz pole o ostatnio obliczoną wartość

Aby program dawał poprawne wyniki nie należy brać pod uwagę ostatniej wartości pola, dlatego po zakończeniu pętli należy wykonać instrukcję: **pole:=pole-abs(p);**

Dla sprawdzenia obliczmy długość odcinka w przedziale -5..5 dla prostej $y=x$

Teoretyczne wyliczenia dają wynik: 25,00000

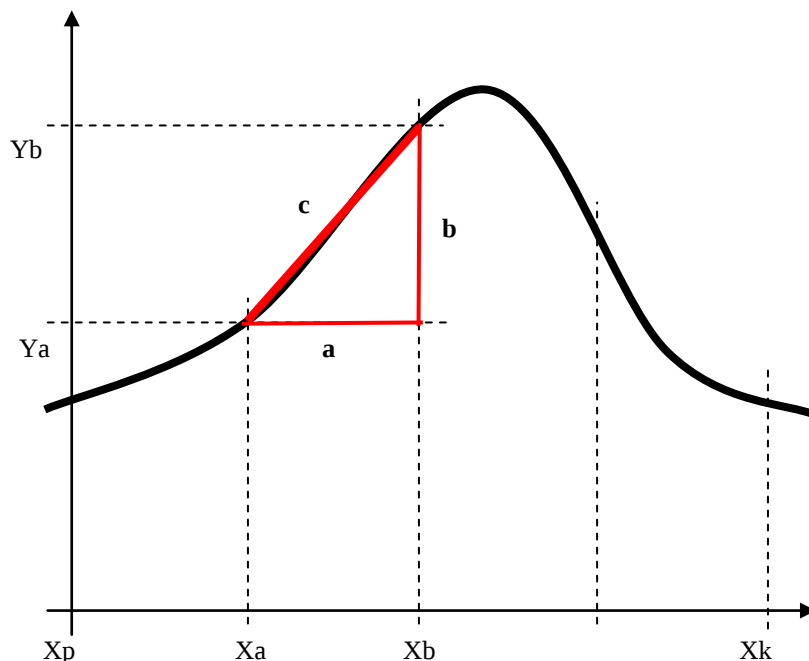
Program wylicza wartość równą 25,000000, gdy przedział wynosi 0.1

Długość krzywej

Idea obliczania długości jest podobna. Podział osi X na jednakowe przedziały, wyliczenie obu wartości, wyliczenie z twierdzenia Pitagorasa wartości przeciwprostokątnej, sumowanie przeciwprostokątnych.

Długość pojedynczego odcinka obliczymy z wzoru: $c = \sqrt{a^2 + b^2}$, co przekładając na oznaczenia na wykresie:

$$c = \sqrt{(X_b - X_a)^2 + (Y_b - Y_a)^2}$$



Dla sprawdzenia obliczmy długość odcinka w przedziale -5..5 dla prostej $y=x$

Teoretyczne wyliczenia dają wynik: 14,14213562

Program wylicza wartość równą 14,14213562, gdy przedział wynosi 0.1

ĆWICZENIA

- napisz funkcję P_CALKA oraz funkcję D_CALKA, obie z parametrami X_p , X_k , skok, które wyliczą pole powierzchni lub długość krzywej. Podobnie jak w poniższych przykładach, funkcje korzystają z zewnętrznej funkcji $Y(x)$
- napisz funkcję PR_CALKA, która liczy pole powierzchni, ale za pomocą prostokątów.
- porównaj wyniki działania obu funkcji, jakie są różnice w obliczeniach

W poniższych przykładach całkujemy funkcję $y=-3x^2-4x+5$ w przedziale $(-5..5)$ ze skokiem 0.1
Sposób dochodzenia do wyniku jest identyczny, zmienia się jedynie wzór: pole, długość

LEKCJA122 - POLE

```
var
  Xp,Xk,Xa,Xb,
  Ya,Yb,
  skok,
  pole,
  p:real;

function Y(x:real):real;
begin
  y:=-3*x*x-4*x+5;
end;

begin
  pole:=0;
  Xp:=-5;
  Xk:=5;
  skok:=0.1;

  Xa:=Xp;
  repeat
    Ya:=Y(Xa);
    Xb:=Xa+skok;
    Yb:=Y(Xb);
    p:=skok*(Ya+Yb)/2;
    pole:=pole+abs(p);
    Xa:=Xb;
  until Xa>=Xk;
  pole:=pole-abs(p);

  writeln(pole);
  readln;
end.
```

LEKCJA123- DŁUGOŚĆ

```
var
  Xp,Xk,Xa,Xb,
  Ya,Yb,
  skok,
  dlugosc,
  d:real;

function Y(x:real):real;
begin
  y:=-3*x*x-4*x+5;
end;

begin
  dlugosc:=0;
  Xp:=-5;
  Xk:=5;
  skok:=0.1;

  Xa:=Xp;
  repeat
    Ya:=Y(Xa);
    Xb:=Xa+skok;
    Yb:=Y(Xb);
    d:=sqrt(sqr(skok)+sqr(Yb-Ya));
    dlugosc:=dlugosc+d;
    Xa:=Xb;
  until Xa>=Xk;
  dlugosc:=dlugosc-d;

  writeln(dlugosc);
  readln;
end.
```