do zdobycia: [30pkt]
czas realizacji: 3 tygodnie

Zestaw 3

Celem ćwiczenia jest napisanie programu równoległego służącego do numerycznego obliczania całki oznaczonej. Do tego celu ponownie zastosujemy dekompozycję trywialną. Problem ten pozwoli usystematyzować zdobytą już wiedzę.

Powstały program powinien obliczać wartość następującego wyrażenia:

$$I = \int_a^b f(x)\ \mathrm{d}x, \tag{1}$$

gdzie funkcja $f(x)$ jest wielomianem stopnia k, tj.

$$f(x) = \alpha_k x^k + \alpha_{k-1}x^{k-1} + \alpha_{k-2}x^{k-2} + \ldots + \alpha_2 x^2 + \alpha_1 x + \alpha_0. \tag{2}$$

Symbolami α_k oznaczono współczynniki wielomianu, które są liczbami rzeczywistymi.

Do przeprowadzenia całkowania numerycznego należy wykorzystać metodę trapezów. Zgodnie z nią, na wstępie dzieli się przedział całkowania $[a, b]$ na m-równych podprzedziałów, każdy o identycznej długości $h = (b - a)/m$. Długość h określa punkty podziału, które dane są jako:

$$x_i = a + ih, \tag{3}$$

gdzie $i = 0, 1, 2, \ldots, m$. Realizując całkowanie zakłada się, że idąc od punktu x_i do punktu x_{i+1} funkcja $f(x)$ zmienia się w sposób liniowy. Wobec tego wartość całki (1) oblicza się zgodnie ze wzorem:

$$I = \int_a^b f(x)\ \mathrm{d}x = \sum_{i=0}^{m-1} \frac{f(x_i) + f(x_{i+1})}{2} h. \tag{4}$$

Pole obszaru ograniczonego funkcją $f(x)$ przybliża się zatem trapezami (stąd nazwa metody).

Rozpatrywany problem można łatwo zrównoleglić stosując dekompozycję trywialną. W podejściu tym występującą po prawej stronie wyrażenia (4) sumę dzieli się na n-podsum, żądając by każdy z n-procesów obliczył wartość jednej podsumy. Można to zapisać w sposób następujący:

$$I = I_0 + I_1 + \ldots + I_{n-1}. \tag{5}$$

Symbolem I_r $(r = 0, 1, 2, \ldots, n - 1)$ oznaczono tutaj podsumę obliczaną przed proces r-ty. Jej postać można zapisać następująco:

$$I_r = \sum_{i=l(r)}^{u(r)} \frac{f(x_i) + f(x_{i+1})}{2} h. \tag{6}$$

Powyżej wprowadzono dwa nowe symbole: $l(r)$ (od ang. lower) oraz $u(r)$ (od ang. upper). Reprezentują one odpowiednio indeks pierwszego ($l(r)$) oraz przedostatniego ($u(r)$) punktu podziału przypisanego procesowi r-temu. Indeksy te spełniają związki:

1. $l(0) = 0$,
2. $u(n - 1) = m$,
3. $u(r - 1) = l(r) - 1$, dla $r > 0$.

Przy takim ujęciu wyznaczenie dekompozycji sprowadza się do określenia wartości liczb $l(r)$ i $u(r)$. Aby zagwarantować równomierny podział pracy, należy zadbać, by liczby punktów podziału przypisane poszczególnym procesom były zbliżone (w wariancie optymalnym dowolne dwa procesy powinny różnić się o nie więcej aniżeli jeden punkt podziału).

Powstały program powinien działać w sposób następujący.

1. Uruchamiając program użytkownik powinien przekazać jeden argument, będący nazwą pliku zawierającego parametry obliczeń. Plik ten zawiera informację o całkowanej funkcji (stopień wielomianu k, wartości współczynników α_i, dla $i = 0, 1, 2, \ldots, k$) oraz szczegóły dotyczące procesu całkowania (przedział całkowania $[a, b]$ oraz liczba podprzedziałów m). Do przekazania nazwy pliku należy posłużyć się wektorem argumentów (argumenty `int argc` oraz `char *argv[]` funkcji `main`). Wczytanie parametrów powinno zostać zrealizowane na masterze (zwykły odczyt szeregowy). Postać pliku wejściowego może wyglądać w sposób następujący:

```
degree      4
coeffs      0.123 2.1 8.1 -3.2 10.0
interval    -10.5 125.0
integration 1000
```

   W powyższym przykładzie stopień wielomianu wynosi $k = 4$, zaś współczynniki α_i wynoszą 0.123, 2.1, 8.1, -3.2 i 10.0, odpowiednio dla $i = 0, 1, 2, 3, 4$. Całkowanie realizowane jest na przedziale od $a = -10.5$ do $b = 125$. W ostatniej linii zawarta jest liczba podprzedziałów, wynosząca $m = 1000$.

2. Po wczytaniu parametry obliczeń (tj. k, α_i , a, b oraz m) należy przesłać do wszystkich procesów, wykorzystując do tego celu operację rozesłania (`MPI_Bcast`) oraz dane spakowane (`MPI_PACKED`). Koniecznie będzie dwukrotne wykorzystanie metody `MPI_Bcast` (oprócz samego przesłania danych wymagane jest uprzednie przesłanie informacji o ich rozmiarze).

3. W kroku kolejnym należy przeprowadzić dekompozycję problemu. Każdy z procesów (**włączając mastera**) powinien określić odpowiadający mu przedział całkowania, wyznaczając wartości parametrów $l(r)$ i $u(r)$.

4. W następnym etapie każdy z procesów (**włączając mastera**) powinien przeprowadzić całkowanie numeryczne, obliczając wartość wyrażenia (6).

5. Po przeprowadzeniu całkowania na masterze powinna nastąpić redukcja (`MPI_Reduce`) rezultatów częściowych (obliczonych podsum (6)). Uzyskany w ten sposób wynik sumowania (5) (obliczona numerycznie wartość całki (1)) powinien zostać wyświetlony na ekranie. Dodatkowo master powinien wyświetlić dokładny (uzyskany analitycznie) wynik całkowania (1). Można go wyznaczyć zgodnie ze wzorem:

$$I = \left(\frac{\alpha_k x^{k+1}}{k+1} + \frac{\alpha_{k-1} x^k}{k} + \frac{\alpha_{k-2} x^{k-1}}{k-1} + \ldots + \frac{\alpha_2 x^3}{3} + \frac{\alpha_1 x^2}{2} + \alpha_0 x \right) \Bigg|_a^b . \tag{7}$$

Pod adresem:

`http://www.mif.pg.gda.pl/homepages/swinczew/AR/integration`

znaleźć można przykładowe dane testowe (pliki wejściowe o nazwach `test1a.in`, `test1b.in`, ..., `test5d.in`). Znajduje się tam również plik o nazwie `results.txt`, który zawiera prawidłowe wyniki (wartość całki obliczoną numerycznie i analitycznie). Warto upewnić się, że napisany program zwraca identyczne rezultaty, niezależnie od wartości n (liczby procesów).

Istotne uwagi/wskazówki

1. Poniżej zamieszczono przykład funkcji realizującej całkowanie na przedziale $[a, b]$ przy m podprzedziałach. Korzysta się w niej z funkcji `double evaluate_f_of_x(double x)`, która oblicza wartość $f(x)$.

```
double integrate(double a, double b, int m)
{
  double x;
  double f_of_x;
  double h = ( b - a ) / m;
  double integral = 0.0;
  for (int i = 0; i <= m; i++)
  {
    x = a + i * h;
    f_of_x = evaluate_f_of_x(x);
    if ( ( i == 0 ) || ( i == m ) )
      integral += 0.5 * f_of_x;
    else
      integral += f_of_x;
  }
  integral *= h;
  return integral;
}
```

   Warto tutaj zwrócić uwagę na fakt, iż skrajne punkty podziału (tj. $x_0 = a$ oraz $x_m = b$) wnoszą do wyrażnia (4) inny wkład, aniżeli pozostałe punkty (czynnik 1/2 zamiast 1). Warto także zaznaczyć, że obecna pętla wykonywana jest począwszy od 0 do m (włącznie). Warto obecne w wyrażeniu (4) mnożenie przez długość h (czynnik stały) wyprowadzić poza pętle.

2. Przy określeniu dekompozycji warto posłużyć się wyrażeniami `p = m / n` oraz `r = m % n`, przypisując początkowym `r` procesom `p + 1` podprzedziałów, zaś pozostałym procesom `p` podprzedziałów. Indeks pierwszego punktu podziału oraz liczbę podprzedziałów odpowiadającą danemu procesowi można łatwo wyznaczyć korzystając z poniższego kodu.

```
int p = m / n;
int r = m % n;
int my_number_of_subintervals;
int my_first_midpoint;

if ( rank < r )
{
  my_number_of_subintervals = p + 1;
  my_first_midpoint = rank * (p + 1 );
}
else
{
  my_number_of_subintervals = p;
  my_first_midpoint = r * (p + 1 ) + ( rank - r ) * p;
}
```