27 grudnia 2022 r.

Jakub Grzywaczewski
Grupa nr 2

Rozwiązywanie układów równania macierzowego $XA = B$ przy użyciu rozkładu Cholesky'ego-Banachiewicza

Projekt nr 19

1 Opis metody

1.1 Układ równań macierzowych

Praca skupią się na znalezieniu efektywnego sposobu rozwiązywania układów równań macierzowych używając rozkładu Cholesky'ego-Banachiewicza. Układ równań macierzowych jest to uogólnienie macierzowej metody rozwiązywania równań typy:

$$XA = b$$

,gdzie A jest macierzą współczynników układu, B wektorem wyrazów wolnych.

Przykładowo układ równań:

$$\begin{cases} a_{1,1}x_1 + a_{1,2}x_1 + \cdots + a_{1,n}x_1 = b_1 \\ a_{2,1}x_2 + a_{2,2}x_2 + \cdots + a_{2,n}x_2 = b_2 \\ \vdots \\ a_{m,1}x_m + a_{m,2}x_m + \cdots + a_{m,n}x_m = b_m \end{cases}$$

możemy zapisać jako równanie macierzowe:

$$\begin{bmatrix} x_1 & x_2 & \cdots & x_m \end{bmatrix} \begin{bmatrix} a_{1,1} & a_{1,2} & \cdots & a_{1,n} \\ a_{2,1} & a_{1,2} & \cdots & a_{1,n} \\ \vdots & & & \\ a_{m,1} & a_{1,2} & \cdots & a_{1,n} \end{bmatrix} = \begin{bmatrix} b_1 & b_2 & \cdots & b_m \end{bmatrix}$$

Uogólnienie pozwala nam na obliczanie wartości x dla kilku wektorów b bez wykonywania dużej ilości nadmiernych obliczeń, poprzez zastąpienie wektora b macierzą B, która jest pionowo nałożonymi na siebie wektorami b dla każdego przypadku.

Zatem kiedy piszemy

$$XA = B$$

$$\begin{bmatrix} x_{1,1} & x_{1,2} & \cdots & x_{1,m} \\ \vdots & & & \\ x_{k,1} & x_{k,2} & \cdots & x_{k,m} \end{bmatrix} \begin{bmatrix} a_{1,1} & a_{1,2} & \cdots & a_{1,n} \\ a_{2,1} & a_{1,2} & \cdots & a_{1,n} \\ \vdots & & & \\ a_{m,1} & a_{1,2} & \cdots & a_{1,n} \end{bmatrix} = \begin{bmatrix} b_{1,1} & b_{1,2} & \cdots & b_{1,m} \\ \vdots & & & \\ b_{k,1} & b_{k,2} & \cdots & b_{k,m} \end{bmatrix}$$

,pokrywamy jednocześnie przypadek B będącego wektorem jak i macierzą.

1.2 Układy prostę do rozwiązania

W rozkładach macierzy na iloczyny często chcemy, aby owe macierze składowe były w formie macierz dolno (bądz górno) trójkątnych. Dzieje się to z powodów, iż macierz trójkątne (oznaczone tutaj L) mają istotne własności m.in.

1. Wyznacznik to iloczyn elementów na diagonali $\det L = \prod_{k=1}^{n} l_{kk}$
2. Układy równań $XL = B$ rozwiązywane w $O(n)$. Intuicyjny algorytm.
3. Łatwe obliczenie macierzy odwrotnej.

1.3 Rozkład Cholesky'ego-Banachiewicza:roz

Rozkład Cholesky'ego-Banachiewicza jest jednym z wielu stosowanych w praktyce rozkładów macierzy na czynniki. Rozkład ten jest stosowany praktycznie wyłącznie dla macierz symetrycznych dodatnio określonych, ponieważ wtedy jest on jednoznacznie określony. Rozkład polega na rozbiciu macierz wejściowej A na iloczyn macierzy dolno trójkątnej L oraz jej sprzężenia L^*.

$$A = LL^*$$

.

Ze względu na fakt, iż w praktyce często spotykamy się z macierzami symetrycznymi (bądź Hermanowskimi) oraz dodatnio określonymi rozkład ten jest niezwykle ważny w świecie matematyki obliczeniowej.

Implementacja algorytmu wyznaczania rozkładu Cholesky'ego-Banachiewicza w pseudokodzie:

Algorithm 1 Rozkład Cholesky'ego-Banachiewicza

Require: A - macierz nxn symetryczna dodatnio określona
  for $k = 1, 2, \cdots, n$ **do**
    $l_{kk} = \sqrt{a_{kk} - \sum_{j=1}^{k-1} l_{kj}^2}$ ▷ Element diagonali
    for $i = k+1, k+2, \cdots, n$ **do** ▷ Pozostałe elementy w kolumnie
      $l_{ik} = \left(a_{ik} - \sum_{j=1}^{k-1} l_{ij} l_{kj}\right) / l_{kk}$
    end for
  end for

Wykorzystując wektoryzację oraz syntaks Matlab'a jesteśmy w stanie pozbyć się sum oraz wyeliminować wewnętrzną pętlę.

Algorithm 2 Rozkład Cholesky'ego-Banachiewicza Zwektoryzowany

Require: A - macierz nxn symetryczna dodatnio określona
Ensure: $L \leftarrow zeros(n)$
  for $k = 1, 2, \cdots, n$ **do**
    $r \leftarrow L(k, 1:(k-1))$
    $L(k,k) \leftarrow \sqrt{A(k,k) - r * (r.')}$ ▷ Element diagonali
    $sektor \leftarrow L((k+1):n, 1:(k-1))$
    $L((k+1):n, k) \leftarrow (A((k+1):n, k) - sektor * (r.'))/L(k,k)$ ▷ Pozostałe elementy w kolumnie
  end for

1.4 Specjalny przypadek

W owej pracy skupimy się jednak nad specjalnym przypadkiem rozkładu Cholesky'ego-Banachewicza. Dokładniej spojrzymy jak można wykorzystać owy rozkład do rozwiązywania układów równań macierzowych $XA = B$, kiedy macierz A jest macierzą hermitowską dodatnio określoną oraz jest ona macierzą pięciodiagonalną.

Ograniczając sobie możliwości form wejściowych macierzy, jesteśmy w stanie znacząco przyśpieszyć wykonywanie rozkładu Cholesky'ego-Banachiewicza wykorzystując fakt,iż przy wystarczająco dużych wymiarach macierze 5 diagonalne składają się w większości z zer.

2 Opis funkcjonalności implementacji metody w Matlabie.

TODO Program obliczeniowy składa się z 9 funkcji oraz 2 skryptów. Teraz opiszę każdą z większymi detalami.

2.1 Główne funkcje / cholDecomp oraz cholDecompDiag

Owe funkcje są implementacją algorytmy rozkładu Cholesky'ego-Banachiewicza. Pierwsza z nich "cholDecomp"jest funkcją, ogólną działającą, dla każdej macierzy dodatnio określonej. Natomiast druga z nich "cholDecompDiag"jest funkcją napisaną specjalnie do rozkładu macierz m-diagonalnych. Funkcja ta wykorzystuje dodatkowe optymalizacje, aby znacząco przyśpieszyć wykonywanie obliczeń, dla tego rodzaju macierzy. Szczególnie widoczna jest różnica, przy dużych i rzadkich macierzach.

Dodatkowym faktem wartym zanotowania jest to, iż rozkładając macierz m-diagonalną ma iloczyn LL^* macierz L także, po części jest macierzą diagonalną tylko zachowana jest tylko dolna (górna) cześć grubej diagonali.

Ogólna: cholDecomp

```
function [L] = cholDecomp(A)
% Funckja obliczajaca rozklad Cholesky'ego-Banachewicza macierzy (A).
% Macierz A musi byc macierza kwadratowa pozytywnie polokreslona
% w przypadku podania macierzy o innej specyfikacji otrzymamy blad
% Funkcja zwraca macierz L dolno-trojkatna o tym samych wymiarach ...
    jak A
% spelniajaca rownanie A = LL*. (rozklad Cholesky'ego)

if ¬ismatrix(A)
    error("Nie podano arugmentu A, b d z argument nie jest macierza")
end

if ¬issymmetric(A)
    error("Macierz A nie jest symetryczna")
end

% Sprawdzamy czy macierz A jest dodatnio polokreslona z pewna ...
    niepewnoscia
eigenVals = eig(A);
tol = length(eigenVals) * eps(max(eigenVals));
if ¬all(eigenVals > -tol)
    error("Macierz A nie jest pozytywnie polokreslona")
end

n = length(A);
L = zeros(n);

```

```
% Specjalny przypadek pierwszej kolumny
L(1,1) = sqrt(A(1,1));
L(2:n, 1) = A(2:n, 1) ./ L(1,1);

% Pozostale kolumny
for k = 2:n
    r = L(k, 1:(k-1)); % Wektor poziomy
    L(k, k) = sqrt(A(k,k) - r * (r'));
    sektor = L( (k+1):n, 1:(k-1) );
    L( (k+1):n, k ) = (A( (k+1):n, k ) - sektor * (r')) / L(k,k);
end
```

Zoptymalizowana pod symetryczne m-diagonalne: cholDecompDiag

```
function [L] = cholDecompDiag(A, m)
% Funckja obliczajaca rozklad Cholesky'ego-Banachewicza macierzy ...
    m-diagonalnej (A).
% Macierz A musi byc macierza kwadratowa pozytywnie p okreslona
% w przypadku podania macierzy o innej specyfikacji otrzymamy blad
% Funkcja zwraca macierz L dolno-trojkatna o tym samych wymiarach ...
    jak A
% spelniajaca rownanie A = LL*. (rozk ad Cholesky'ego)

if ¬ismatrix(A)
    error("Nie podano arugmentu A, b dz argument nie jest macierza")
end

if ¬issymmetric(A)
    error("Macierz A nie jest symetyczna")
end

% Sprawdzamy czy macierz A jest dodatnio polokreslona z pewna ...
    niepewnoscia
eigenVals = eig(A);
tol = length(eigenVals) * eps(max(eigenVals));
if ¬all(eigenVals > -tol)
    error("Macierz A nie jest pozytywnie polokreslona")
end

n = length(A);
L = zeros(n);

% Liczba niezerowych elementow w dol pierwszej kolumnie od diagonali.
m_k = idivide(m, int32(2));

% Specjalny przypadek pierwszej kolumny
L(1,1) = sqrt(A(1,1));
L(2:(m_k+1), 1) = A(2:(m_k+1), 1) ./ L(1,1);

% Pozostale kolumny
for k = 2:n
    % Tworzymy wektor indeksow niezerowych elementow dla wierza
    slWinX = max(1, k-m_k):(k-1);

```

```
    r = L(k, slWinX);

    % Diagonalny element
    L(k, k) = sqrt(A(k,k) - r * (r'));

    % Tworzymy wektor indeksow niezerowych elementow dla kolumny
    slWinY = (k+1):min(k+m_k, n);

    % Reszta kolumny
    sektor = L( slWinY, slWinX );
    L( slWinY, k ) = (A( slWinY, k ) - sektor * (r')) / L(k,k);
end
```

2.2 Funkcje solve*

Funkcje solve* rozwiązują układ równań macierzowy $XA = B$, kiedy macierz A jest macierzą trójkątną.

1. solveLower - Rozwiązuje rówanie macierzowe $XL = B$, z macierzą L dolnotrójkątną.
2. solveLowerDiag - Zoptymalizowana wersjca solveLower pod macierze dolnotrójkąte z macierzy m-diagonalnych.
3. solveUpper - Rozwiązuje rówanie macierzowe $XL = B$, z macierzą L górnotrójkątną.
4. solveUpperDiag - Zoptymalizowana wersjca solveUpper pod macierze górnotrójkąte z macierzy m-diagonalnych.

```
function [X] = solveLower(L, B)
% Funkcja rozwizzujzca uklad rowan macierzowych XL = B,
% gdzie macierz L jest macierza dolno-trojkatna (odwracalna).
% Argument B moze byc albo macierza, badz wektorem poziomym

if ¬ismatrix(L)
    error("Argument L nie podany, badz nie jest macierza");
end

if ¬ismatrix(B)
    error("Argument B nie podany, badz nie jest macierza");
end

if ¬istril(L)
    error("Macierz L nie jest macierza dolno-trojkatna");
end


[nRowsB, nColsB] = size(B);
[nRowsL, nColsL] = size(L);
```

```
if nColsB ≠ nColsL
    error("Macierze nie sa odpowiednych wymiarow")
end

% Dla ulatwienie notacji
n = nRowsL; % Liczba wierszy
k = nRowsB; % Liczba kolumn
X = zeros(k, n);

X(1:k, n) = B(1:k, n) ./ L(n,n);
for i = (n-1):-1:1
    rest = X(1:k, (i+1):n) * L((i+1):n, i);
    X(1:k, i) = (B(1:k, i) - rest) ./ L(i,i);
end
```

```
function [X] = solveUpper(U, B)
% Funkcja rozwiazujaca uklad rowan macierzowych XU = B,
% gdzie macierz U jest macierza gorno-trojkatna (odwracalna).
% Argument B moze byc albo macierza, badz wektorem poziomym

if ¬ismatrix(U)
    error("Argument U nie podany, badz nie jest macierza");
end

if ¬ismatrix(B)
    error("Argument B nie podany, badz nie jest macierza");
end

if ¬istriu(U)
    error("Macierz U nie jest macierza gorno-trojkatna");
end

[nRowsB, nColsB] = size(B);
[nRowsU, nColsU] = size(U);

if nColsB ≠ nColsU
    error("Macierze nie sa odpowiednych wymiarow")
end

% Dla ulatwienie notacji
n = nRowsU; % Liczba wierszy
k = nRowsB; % Liczba kolumn
X = zeros(k, n);

X(1:k, 1) = B(1:k, 1) ./ U(1,1);
for i = 2:n % Index kolumny, ktora obliczamy
    rest = X(1:k, 1:(i-1)) * U(1:(i-1), i);
    X(1:k, i) = (B(1:k, i) - rest) ./ U(i,i);
end
```

2.3 Funkcje generujące macierze

Do obliczeń używane są macierze w specyficznej formie. Mianowicie wspomniane wcześniej macierze symetryczne dodatnio określone m-diagonalne. W celu generowania losowych macierz w tej formie powstały 3 funkcje:

1. randKdiag - Funkcja generuje losową macierz symetryczną dodatnio określona z elementami w liczbach rzeczywistych używając funkcji rand() jako podstawy.
2. randKdiagC - To samo co w funkcji randKdiag, lecz macierz posiada element z liczy zespolonych
3. onesKdiag - Generuje macierz m-diagonalna wypełnioną na tych diagonalach jedynkami (do wizualizacji)

```
function [A] = randKdiag(n, m)
% Funkcja tworzy losowa dodatnio okreslona macierz m-diagonalna o
% wymiarach nxn z elementami w liczbach rzeczywistych.

k = idivide(m, int32(2)) + 1;
if n < k
    error("Nie mozna utworzyc macierzy m-diagonalej o takich ...
        wymiarach");
end

M = rand(n);
% Upewniany sie, ze macierz bedzie macierza spanujaca cala ...
    przestrzen.
while det(M) == 0
    M = rand(n);
end

Z = zeros(n);

for i = 0:(k - 1)
    Z = Z + diag(diag(M, i), i);
end

A = 0.5 * (Z + Z') + eye(n);
end
```

```
function [Z] = randKdiagC(n, m)
% Funkcja tworzy losowa dodatnio okreslona macierz m-diagonalna o
% wymiarach nxn z elementami w liczbach zespolonych.

k = idivide(m, int32(2)) + 1;
if n < k
    error("Nie mozna utworzyc macierzy m-diagonalej o takich ...
        wymiarach");
end
```

```

M = complex(rand(n, n), rand(n,n));
% Upewniany sie, ze macierz bedzie macierza spanujaca cala ...
    przestrzen.
while det(M) == 0
    M = complex(rand(n, n), rand(n,n));
end

A = M*(M') + n*eye(n);
Z = zeros(n);

Z = Z + diag(diag(A, 0), 0);
for i = (-k+1):-1
    Z = Z + diag(diag(A, i), i);
end
for i = 1:(k-1)
    Z = Z + diag(diag(A, i)', i);
end
end
```

```
 function [Z] = onesKdiag(n, m)
% Funkcja tworzy macierz m-diagonalna wypelniona jedynkami.
% wymiarach nxn z elementami w liczbach rzeczywistych.

k = idivide(m, int32(2)) + 1;
if n < k
    error("Nie mozna utworzyc macierzy m-diagonalej o takich ...
       wymiarach");
end

M = ones(n);
Z = zeros(n);

for i = (-k + 1):(k - 1)
    Z = Z + diag(diag(M, i), i);
end

end
```

2.4 Aplikacja - kwadWiz

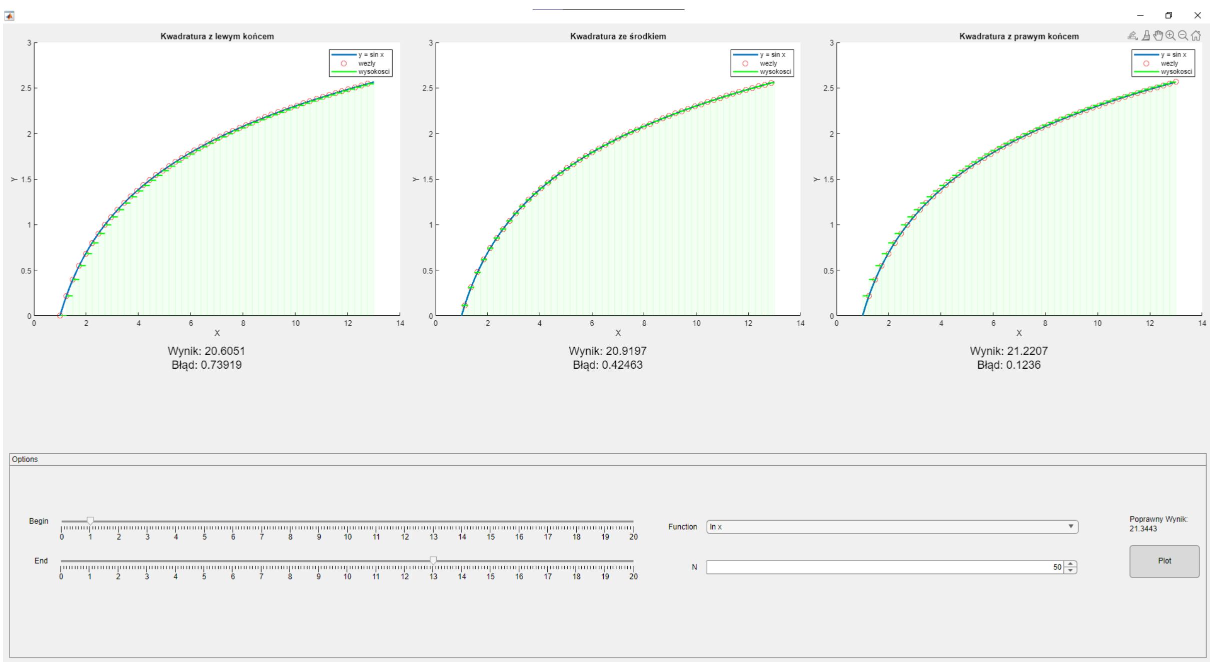
Jest to aplikacja GUI pozwalająca użytkownikowi szybkie sprawdzenie jak mają się błędy poszczególnych kwadratur do siebie dla wybranych przez niego przedziałów, funkcji oraz licz węzłów.

W aplikacji jesteśmy w stanie

- Wybrać oba końce przedziałów
- Wybrać funkcje podcałkową z listy dostępnych
- Zaznaczyć ile pod-przedziałów (prostokątów) chcemy użyć

Tabela 1: Czas wykonania fukcji w sekundach w zależności od wielkości macierzy

n	chol()	cholDecompDiag()	cholDecomp()
10	0.0000	0.0000	0.0000
20	0.0000	0.0001	0.0001
40	0.0000	0.0001	0.0002
80	0.0000	0.0003	0.0004
160	0.0001	0.0005	0.0012
500	0.0006	0.0019	0.0238
1000	0.0036	0.0044	0.3381
2000	0.0182	0.0084	2.9059
3000	0.0814	0.0158	
4000	0.1770	0.0183	
8000	1.5460	0.0385	



Rysunek 1: Interfejs aplikacji

- Wyświetlić wykresy dla każdej z 3 kwadratur
- Odczytać wyniki całkowania numerycznego oraz ich błędy bezwzględne

3 Przykłady obliczeniowe

Przyglądając w podstawowy sposób funkcje (używając aplikacji GUI) i jak dla nich zachowują się poszczególne kwadratury zauważyłem kilka ciekawych przypadków, które potem zadbałem dogłębniej używając skryptu "obliczenia.m".

Podczas obliczeń używałem następujących wartości n: 5 10 50 100 200 500 1000 10000.

3.1 Funkcja nieparzysta na symetrycznym przedziale

Weźmy pod uwagę funkcję, która jest nieparzystą np. $\sin x$, bądź $sin 2x$ oraz przedział symetryczny na przykład $(-\pi, \pi)$. Oczekujemy, aby wyniki były jak najbardziej zbliżone do 0.

Dla $\sin x$:

```
Wyniki
   1.0e-15 *

   -0.2790    0.1395    0.1539
   -0.0698   -0.0698    0.0072
    0.0244   -0.0366    0.0401
   -0.0044    0.0453    0.0033
    0.1107    0.0678    0.1146
   -0.1377   -0.0166   -0.1361
   -0.3210   -0.0764   -0.3202
    0.0671    0.0813    0.0671

Bledy
   1.0e-15 *

    0.2790    0.1395    0.1539
    0.0698    0.0698    0.0072
    0.0244    0.0366    0.0401
    0.0044    0.0453    0.0033
    0.1107    0.0678    0.1146
    0.1377    0.0166    0.1361
    0.3210    0.0764    0.3202
    0.0671    0.0813    0.0671
```

Każdy z tych błędów jest straszliwie mały ze względu na naturę sinusa.

Podobnie dla $\sin 2x$:

```
Wyniki
   1.0e-15 *

    0.3078         0   -0.3078
    0.1395         0   -0.1539
   -0.0279   -0.0070   -0.0593
   -0.0035   -0.0453   -0.0187
   -0.0113   -0.0100   -0.0191
    0.0298    0.1020    0.0267
   -0.0430    0.0211   -0.0446
   -0.0514   -0.0639   -0.0516

Bledy
   1.0e-15 *
```

```

    0.3078         0    0.3078
    0.1395         0    0.1539
    0.0279    0.0070    0.0593
    0.0035    0.0453    0.0187
    0.0113    0.0100    0.0191
    0.0298    0.1020    0.0267
    0.0430    0.0211    0.0446
    0.0514    0.0639    0.0516
```

3.2 Funkcja nieparzysta na niesymetrycznym przedziale

Dalej zostańmy przy funkcjach $\sin x$ oraz $\sin 2x$ tylko tym razem użyjmy przedziału $(0, \frac{\pi}{2})$.

Dla $sinx$:

```
Wyniki
    0.6326    0.8052    0.9468
    0.8192    0.9011    0.9763
    0.9642    0.9800    0.9956
    0.9821    0.9900    0.9978
    0.9911    0.9950    0.9989
    0.9964    0.9980    0.9996
    0.9982    0.9990    0.9998
    0.9998    0.9999    1.0000

Bledy
    0.3674    0.1948    0.0532
    0.1808    0.0989    0.0237
    0.0358    0.0200    0.0044
    0.0179    0.0100    0.0022
    0.0089    0.0050    0.0011
    0.0036    0.0020    0.0004
    0.0018    0.0010    0.0002
    0.0002    0.0001    0.0000
```

Oraz dla $\sin 2x$:

```
Wyniki
    0.7584    0.8209    0.7584
    0.8908    0.9046    0.8908
    0.9797    0.9802    0.9797
    0.9899    0.9900    0.9899
    0.9950    0.9950    0.9950
    0.9980    0.9980    0.9980
    0.9990    0.9990    0.9990
    0.9999    0.9999    0.9999

Bledy
    0.2416    0.1791    0.2416
    0.1092    0.0954    0.1092
```

```
     0.0203    0.0198    0.0203
     0.0101    0.0100    0.0101
     0.0050    0.0050    0.0050
     0.0020    0.0020    0.0020
     0.0010    0.0010    0.0010
     0.0001    0.0001    0.0001
```

Wyniki w przypadku *sinx* różnią się na tyle, iż warto wrzucić owe wartości do tabeli.

Tabela 2: Błędy względne kwadratur przy funkcji $\sin x$

n	Lewy koniec	Środek	Prawy koniec
5	0.3674	0.1948	0.0532
10	0.1808	0.0989	0.0237
50	0.0358	0.0200	0.0044
100	0.0179	0.0100	0.0022
200	0.0089	0.0050	0.0011
500	0.0036	0.0020	0.0004
1000	0.0018	0.0010	0.0002
10000	0.0002	0.0001	0.0000

3.3 Funkcja parzysta na symetrycznym przedziale

Widząc wyniki dla funkcji $\sin x$ możemy postarzeć na jej brata bliźniaka $\cos x$ oraz $\cos 2x$. Zmieńmy natomiast trochę przedział na $(-\frac{\pi}{2}, \frac{\pi}{2})$.

Dla $\cos x$:

```
Wyniki
     1.5169    1.6419    1.5169
     1.7817    1.8092    1.7817
     1.9593    1.9603    1.9593
     1.9798    1.9801    1.9798
     1.9900    1.9900    1.9900
     1.9960    1.9960    1.9960
     1.9980    1.9980    1.9980
     1.9998    1.9998    1.9998

Bledy
     0.4831    0.3581    0.4831
     0.2183    0.1908    0.2183
     0.0407    0.0397    0.0407
     0.0202    0.0199    0.0202
     0.0100    0.0100    0.0100
     0.0040    0.0040    0.0040
     0.0020    0.0020    0.0020
     0.0002    0.0002    0.0002
```

Dla $\cos 2x$:

```
Wyniki
   1.0e-15 *

    0.1082    0.1395         0
    0.2442    0.2442    0.2093
    0.2721    0.2930    0.2790
    0.0174   -0.0070    0.0419
    0.1413   -0.0436    0.1744
    0.2211    0.4046    0.2072
    0.5047    0.5204    0.5099
    0.2505   -0.3064    0.2510

Bledy
   1.0e-15 *

    0.0142    0.0171    0.1225
    0.1217    0.1217    0.0868
    0.1496    0.1705    0.1566
    0.1050    0.1294    0.0806
    0.0188    0.1661    0.0519
    0.0987    0.2821    0.0847
    0.3822    0.3979    0.3875
    0.1280    0.4289    0.1286
```

3.4 Funkcja parzysta na niesymetrycznym przedziale

Dalej $\cos x$ oraz $\cos 2x$, tylko na przedziale $(0, \frac{\pi}{2}$.

Dla $\cos x$:

```
Wyniki
    0.9468    0.8052    0.6326
    0.9763    0.9011    0.8192
    0.9956    0.9800    0.9642
    0.9978    0.9900    0.9821
    0.9989    0.9950    0.9911
    0.9996    0.9980    0.9964
    0.9998    0.9990    0.9982
    1.0000    0.9999    0.9998

Bledy
    0.0532    0.1948    0.3674
    0.0237    0.0989    0.1808
    0.0044    0.0200    0.0358
    0.0022    0.0100    0.0179
    0.0011    0.0050    0.0089
    0.0004    0.0020    0.0036
    0.0002    0.0010    0.0018
    0.0000    0.0001    0.0002
```

Tabela 3: Błędy względne kwadratur przy funkcji $\cos x$

n	Lewy koniec	Środek	Prawy koniec
5	0.0532	0.1948	0.3674
10	0.0237	0.0989	0.1808
50	0.0044	0.0200	0.0358
100	0.0022	0.0100	0.0179
200	0.0011	0.0050	0.0089
500	0.0004	0.0020	0.0036
1000	0.0002	0.0010	0.0018
10000	0.0000	0.0001	0.0002

Dla $\cos 2x$:

```
Wyniki
    0.3142    0.0000   -0.3142
    0.1571    0.0000   -0.1571
    0.0314    0.0000   -0.0314
    0.0157    0.0000   -0.0157
    0.0079    0.0000   -0.0079
    0.0031    0.0000   -0.0031
    0.0016    0.0000   -0.0016
    0.0002    0.0000   -0.0002

Bledy
    0.3142    0.0000    0.3142
    0.1571    0.0000    0.1571
    0.0314    0.0000    0.0314
    0.0157    0.0000    0.0157
    0.0079    0.0000    0.0079
    0.0031    0.0000    0.0031
    0.0016    0.0000    0.0016
    0.0002    0.0000    0.0002
```

Warto zauważyć,iż w tym przypadku użycie kwadratury z środkowym węzłem okazuje się najefektywniejsze.

Tabela 4: Błędy względne kwadratur przy funkcji $\cos 2x$

n	Lewy koniec	Środek	Prawy koniec
5	0.3142	0.0000	0.3142
10	0.1571	0.0000	0.1571
50	0.0314	0.0000	0.0314
100	0.0157	0.0000	0.0157
200	0.0079	0.0000	0.0079
500	0.0031	0.0000	0.0031
1000	0.0016	0.0000	0.0016
10000	0.0002	0.0000	0.0002

3.5 Funkcja wykładnicza

Najprościej rozważamy funkcję e^x dla przedziału $(-2, 5)$.

```
Wyniki
   43.6606  104.7364  251.2496
   88.2134  130.1447  192.0078
  135.1799  145.1888  155.9388
  141.6665  146.7645  152.0459
  144.9568  147.5288  150.1465
  146.9458  147.9801  149.0216
  147.6112  148.1292  148.6491
  148.2111  148.2630  148.3149
Bledy
  104.6172   43.5414  102.9718
   60.0645   18.1331   43.7300
   13.0980    3.0890    7.6609
    6.6113    1.5134    3.7681
    3.3210    0.7490    1.8687
    1.3321    0.2978    0.7438
    0.6666    0.1486    0.3713
    0.0667    0.0148    0.0371
```

Tabela 5: Błędy względne kwadratur przy funkcji e^x

n	Lewy koniec	Środek	Prawy koniec
5	104.6172	43.5414	102.9718
10	60.0645	18.1331	43.7300
50	13.0980	3.0890	7.6609
100	6.6113	1.5134	3.7681
200	3.3210	0.7490	1.8687
500	1.3321	0.2978	0.7438
1000	0.6666	0.1486	0.3713
10000	0.0667	0.0148	0.0371

3.6 Funkcja hiperboliczna

Rozważamy funkcję $\frac{1}{x}$ na przedziale $(0.001, 10)$:

```
Wyniki
   1.0e+03 *
    2.0013    0.0027    0.0017
    1.0023    0.0037    0.0025
    0.2043    0.0057    0.0044
    0.1051    0.0064    0.0051
    0.0558    0.0071    0.0058
    0.0267    0.0079    0.0067
    0.0173    0.0084    0.0073
    0.0098    0.0092    0.0088
Bledy
   1.0e+03 *
    1.9921    0.0065    0.0075
    0.9931    0.0055    0.0067
    0.1951    0.0035    0.0048
    0.0959    0.0028    0.0041
    0.0466    0.0021    0.0034
    0.0175    0.0013    0.0025
    0.0081    0.0008    0.0019
    0.0006    0.0000    0.0004
```

4 Analiza wyników

Po przejściu przez funkcje, których współgranie z metodą prostokątów widzimy powyżej oraz więcej, jednym z wniosków jaki jesteśmy w stanie zaobserwować jest fakt, iż błąd dla węzła środkowego jest prawie zawsze najmniejszy, a kiedy nie jest najmniejszy nie odbiega on w dużym stopniu od najlepszego.

Bywały funkcje dla których wybranie przykładowo lewego końca powodowało znaczący wzrost błędu pomiarowego przy stosunkowo małej liczbie węzłów.

Bezpiecznym zatem wyborem na węzeł staje się środek przedziału.

Literatura

[1] Notatki do Metod Numerycznych autorstwa dr. Iwony Wróbel