Sprawozdanie

z Laboratorium Metod numerycznych

Ćwiczenie 1:

Błędy w obliczeniach numerycznych

Autor: Dominik Gajda [24066]

Maksym Dmytruk [240353]

Data wykonania ćwiczenia: 19.10.2022

Data sporządzenia i oddania sprawozdania: 25.10.2022

Część 1. Obliczanie wielomianu różnymi

Napisać funkcje obliczające wartość wielomianu jako ważonej sumy potęg zmiennej niezależnej (funkcja `pval1` i `pval2` na końcu sprawozdania) oraz przy pomocy schematu Hornera (funkcja `horner`). Dla wielomianu

$$W(x) = (x - x_0)^9$$

gdzie $x_0 = \frac{3}{2}$ przetestowano te funkcje dla argumentów pojedynczej i podwójnej precyzji.

```
format long
format compact
```

Pierwszy wariant funkcji obliczającej wartość wielomianu ma nosić nazwę `pval1` i dodawać wyrazy wielomianu w porządku malejących potęg czyli $p_1 x^n + p_2 x^{n-1} + \ldots + p_n x + p_{n+1}$ od lewej do prawej. Na początek tworzymy dane testowe w postaci współczynników wielomianu stopnia 3

$P(x) = (x + 1)(x - 2)(2x - 3)$

oraz wektor wartości zmiennej niezależnej

```
p=[2,-5,-1,6]
```

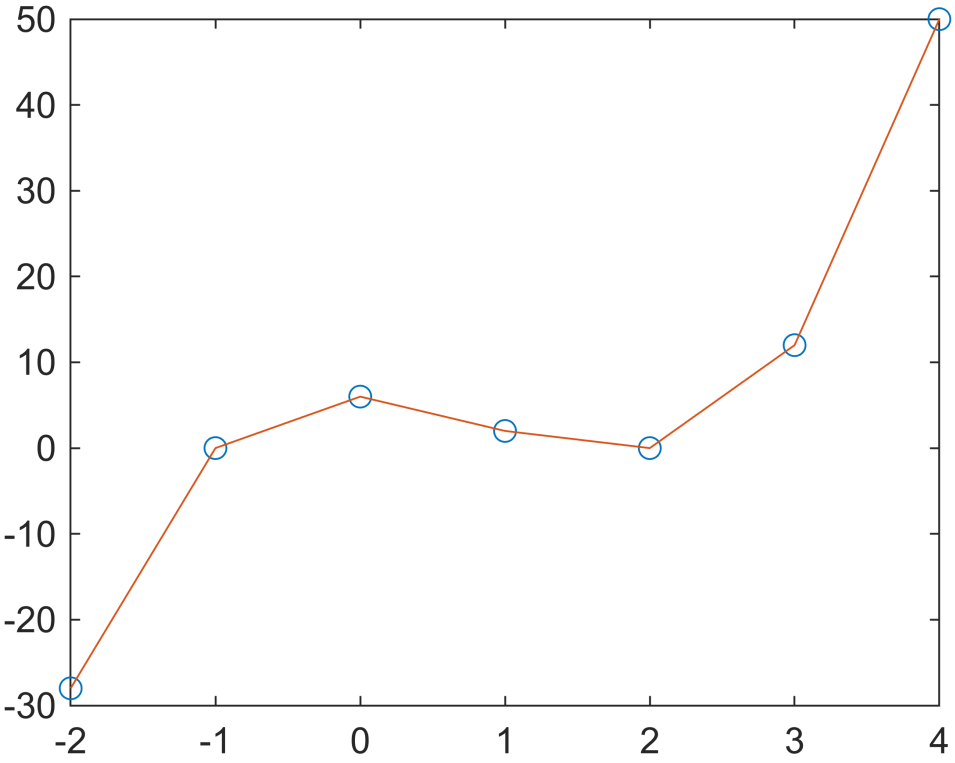
```
p = 1×4
     2    -5    -1     6
```

```
x=[-2,-1,0,1,2,3,4]
```

```
x = 1×7
    -2    -1     0     1     2     3     4
```

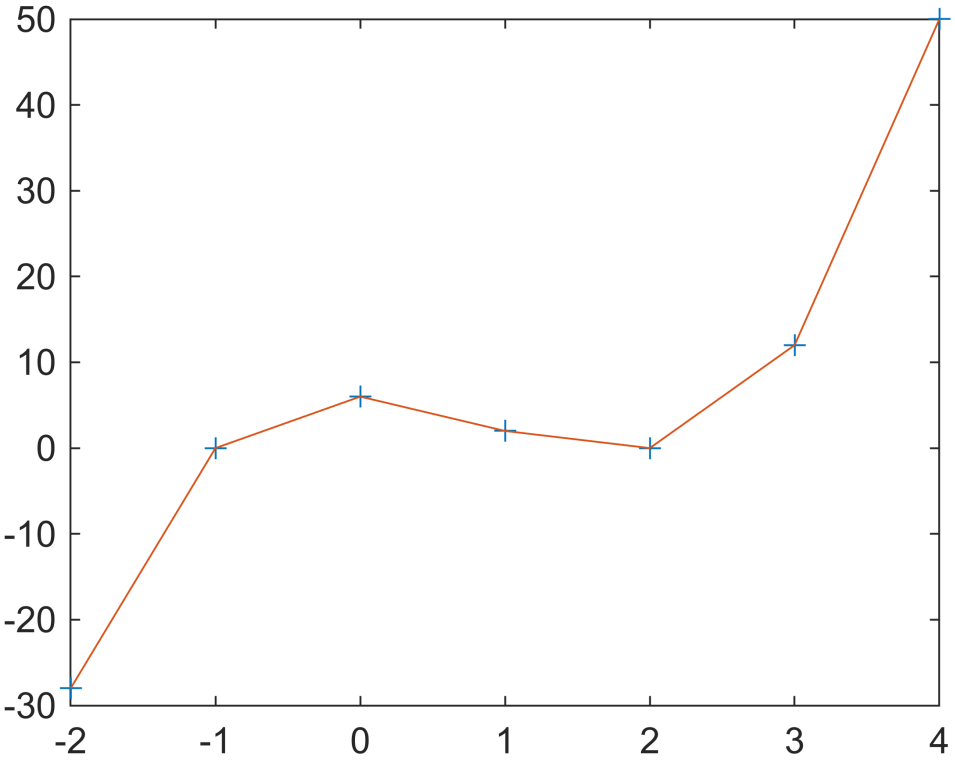
Uzupełniono poniższy kod i usunięto znaki komentarza, które "ukrywają" obecnie nieprawidłowe syntaktycznie linie przed analizatorem kodu. Po sprawdzeniu kodu (przy użyciu "Run Section" skopiowano kod do nowej sekcji, zaznaczono część między komentarzami z podwójną linią ze znaków równości i wybrano z menu kontekstowego "*Convert to local function*", nazywając ją `pval1`).

```
y = pval1(p, x);
plot(x,y,'o',x,polyval(p,x),'-')
```



Drugi wariant dodaje wyrazy wielomianu w porządku rosnących potęg (po sprawdzeniu, skopiowano kod do nowej sekcji i przekształcono kod na funkcję lokalną, nazywając ją `pval2`):

```
y = pval2(p, x);
plot(x,y,'+',x, polyval(p,x))
```

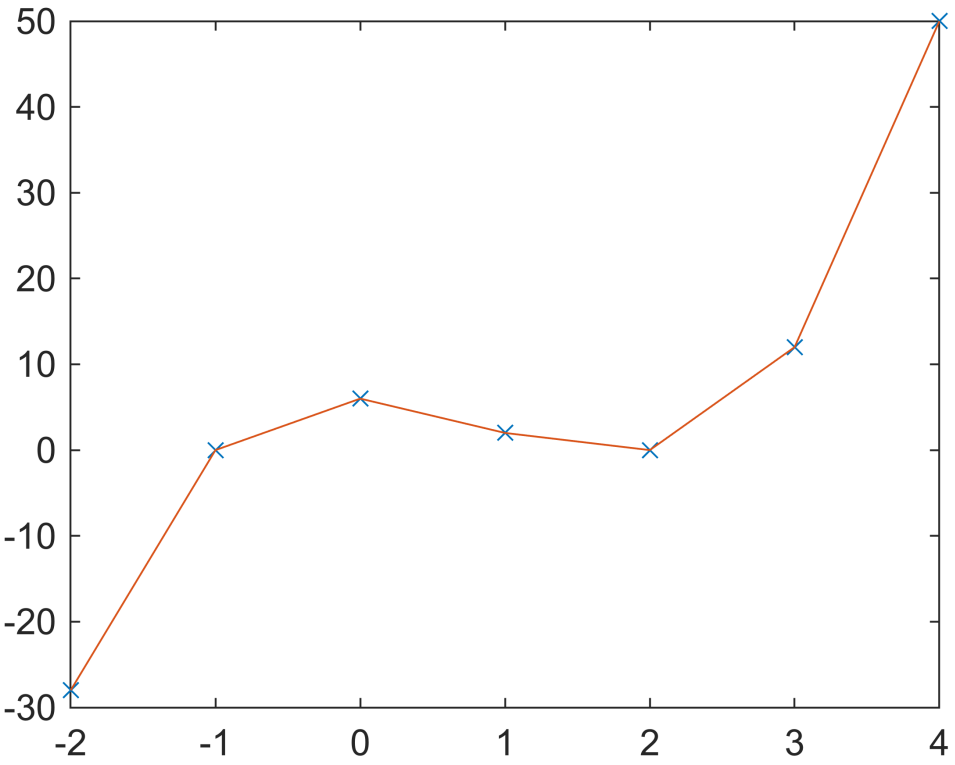


Wariant trzeci realizuje schemat Hornera (nazwa `horner`), to znaczy według schematu po prawej stronie poniższego równania:

$$p_1 x^n + p_2 x^{n-1} + \ldots + p_n x + p_{n+1} = ((\ldots((p_1 x + p_2)x + p_3)x + \ldots\ )x + p_n)x + p_{n+1}$$

Wyrażenie jest obliczane ściśle w porządku określonym przez nawiasy, więc należało zastosować pętlę w której zmiennej pomocniczej będą nadawane wartości kolejnych, coraz bardziej zewnętrznych nawiasów.

```
y = horner(p, x);
plot(x,y,'x',x,polyval(p,x),'-')
```



Obserwacja działania napisanych funkcji dla podanego na wstępie wielomianu posiadającego zero wielokrotne

```
w=3*ones(1,9)/2
```

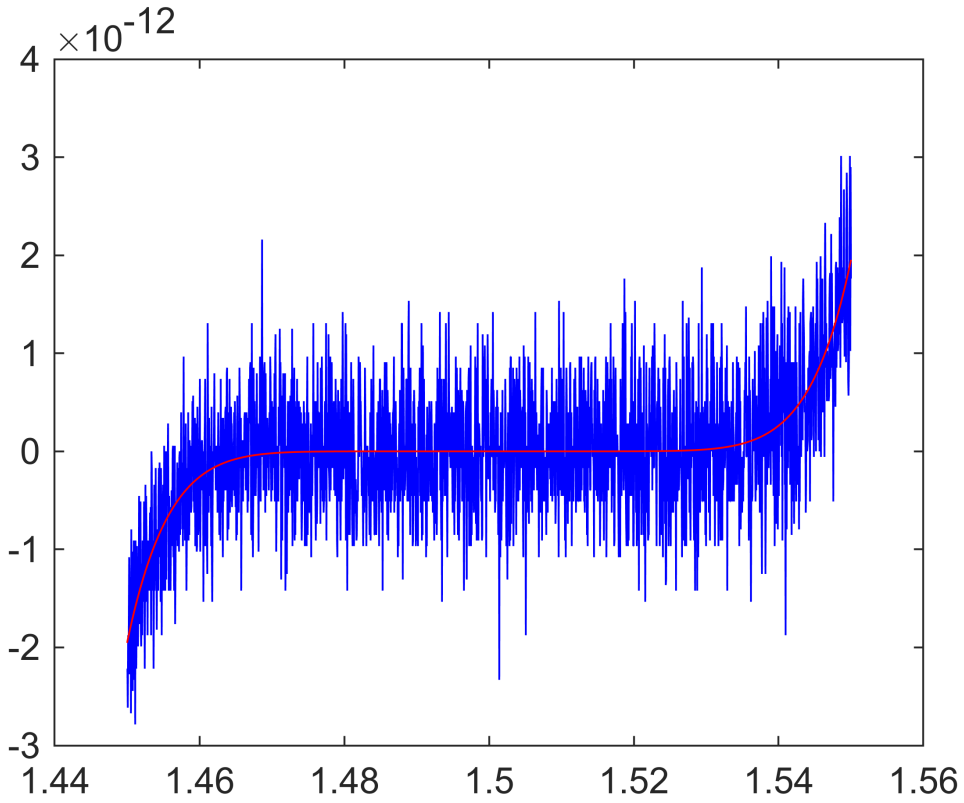
```
w = 1×9
   1.500000000000000   1.500000000000000   1.500000000000000   1.500000000000000 ···
```

```
p=poly(w) % Wygenerowanie współczynników wielomianu 9 stopnia wspomnianego na wstępie
```

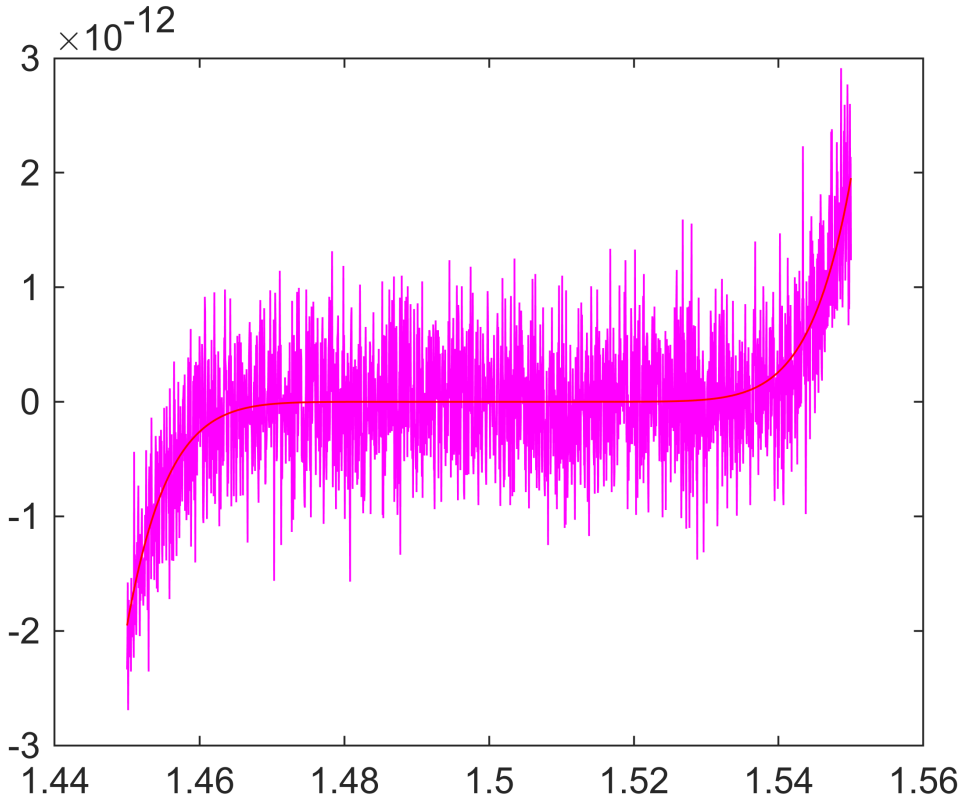
```
p = 1×10
10^2 ×
   0.010000000000000  -0.135000000000000   0.810000000000000  -2.835000000000000 ···
```

Narysowanie wykresów rozważanego wielomianu w przedziale [1.45, 1.55], zawsze we wspólnym układzie współrzędnych z tymże wielomianem obliczonym jako 9 potęga dwumianu

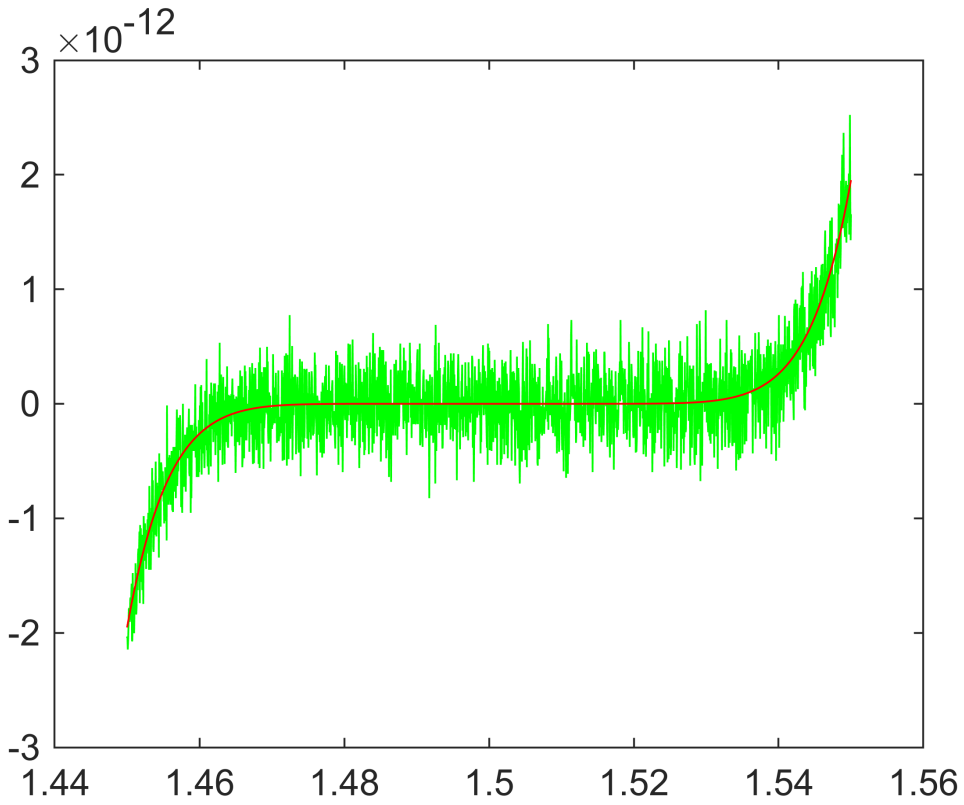
```
x=linspace(1.45,1.55,2049); % Wygenerowanie gęstej siatki punktów
y0=(x-3/2).^9; % Obliczenie wielomianu jako potęgi dwumianu (do porównań)
plot(x,pval1(p,x),'b-',x,y0,'r-')
```




```
plot(x,pval2(p,x),'m-',x,y0,'r-')
```

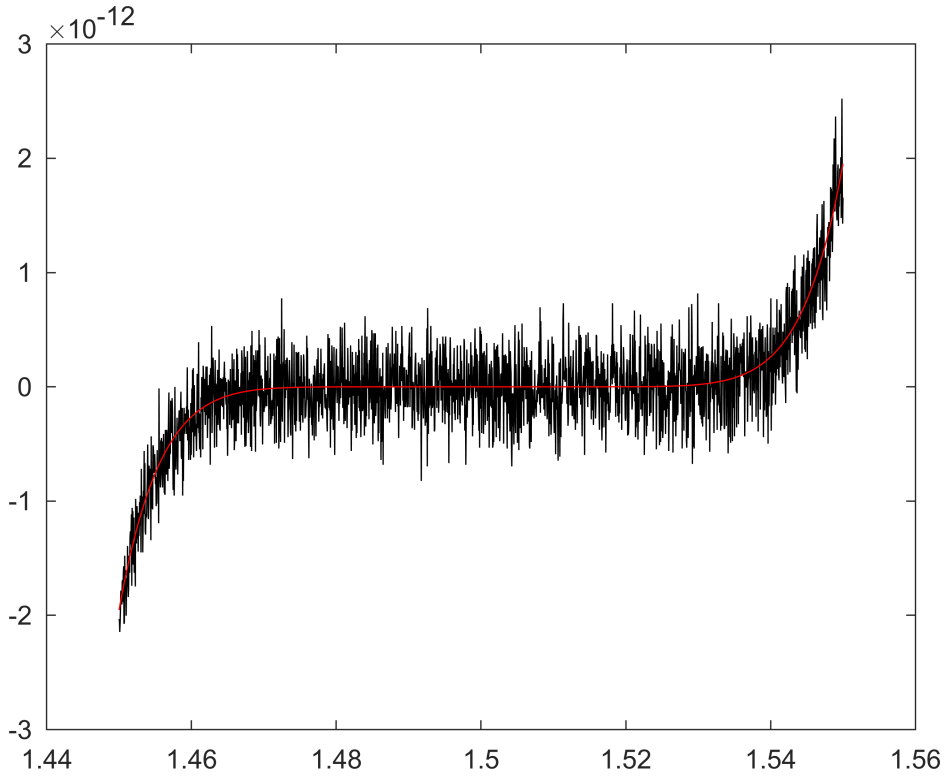


```
plot(x,horner(p,x),'g-',x,y0,'r-')
```



Porównano w analogiczny sposób wyniki obliczeń wrzystkich trzech wariantów funkcji z wbudowaną funkcją Matlab'a `polyval` najpierw poprzez zwizualizowanie przebiegu funkcji na tle narysowanych wykresów zdefiniowanych przez nas funkcji, a następnie poprzez obliczenie pierwiastka błędu średniokwadratowego kolejnych funkcji.

```
plot(x,polyval(p,x),'k-',x,y0,'r-')
```




```
err_pval1 = sqrt(mean((pval1(p,x)-y0).^2))
```

```
err_pval1 =
     5.912429915366391e-13
```

```
err_pval2 = sqrt(mean((pval2(p,x)-y0).^2))
```

```
err_pval2 =
     4.792622125230379e-13
```

```
err_horner = sqrt(mean((horner(p,x)-y0).^2))
```

```
err_horner =
     2.601181922156722e-13
```

```
err_polyval = sqrt(mean((polyval(p,x)-y0).^2))
```

```
err_polyval =
     2.601181922156722e-13
```

Błąd obliczony dla funkcji `pval1` jest największy, dla funkcji `pval2` błąd jest mniejszy ale wciąż tego samego rzędu. Funkcja `horner` wykazuje się najmniejszym błędem, równym co do wartości temu, obliczonego dla funkcji wbudowanej `polyval.` Może to oznaczać, że metoda hornera jest wykorzystywana w implementacji funkcji `polyval`.

Część 2. Eksperymenty dotyczące numerycznego obliczania pochodnej

Dokonano porównania ilorazów różnicowych z różnicą centralną i progresywną, jako narzędzia do wyznaczania przybliżonych wartości pochodnej, na przykładzie funkcji:

$$f(x) = e^{\sqrt{x}}$$

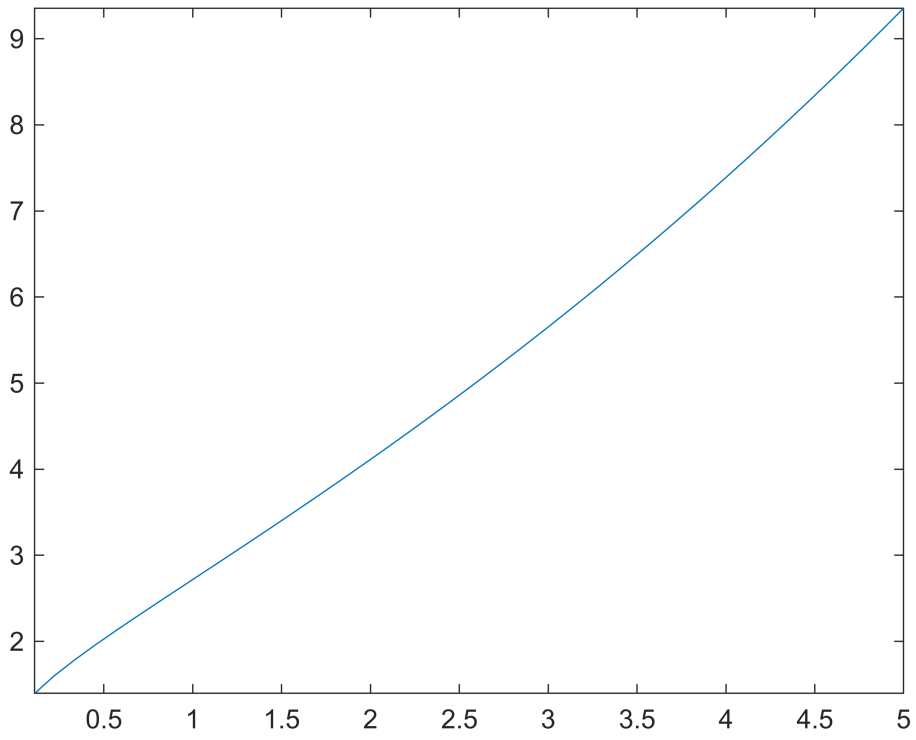
wygenerowanej przez plik gen3INFcw1 przy pomocy kodu zamieszczonego poniżej. Wykorzystując Symbolic Math Toolbox obliczono pochodne tej funkcji do trzeciego rzędu włącznie.

```
[fun,x0]=gen3INFcw1(240661) % 240661 jest numerem indeksu
```

```
fun = function_handle with value:

    @(x)exp(sqrt(x))
x0 =
     4
```

```
fplot(fun)
```




```
syms x
f(x)=fun(x)
```

```
f(x) = e^(x^(1/2))
```

```
f1(x)=diff(f)
```

```
f1(x) =
```

$$\frac{e^{\sqrt{x}}}{2\sqrt{x}}$$

```
f2(x)=diff(f1)
```

```
f2(x) =
```

$$\frac{e^{\sqrt{x}}}{4x} - \frac{e^{\sqrt{x}}}{4x^{3/2}}$$

```
f3(x)=diff(f2)
```

```
f3(x) =
```

$$\frac{e^{\sqrt{x}}}{8x^{3/2}} - \frac{3e^{\sqrt{x}}}{8x^{2}} + \frac{3e^{\sqrt{x}}}{8x^{5/2}}$$

```
f1x0=double(f1(x0))
```

```
f1x0 =
   1.847264024732663
```

```
f2x0=double(f2(x0))
```

```
f2x0 =
   0.230908003091583
```

```
f3x0=double(f3(x0))
```

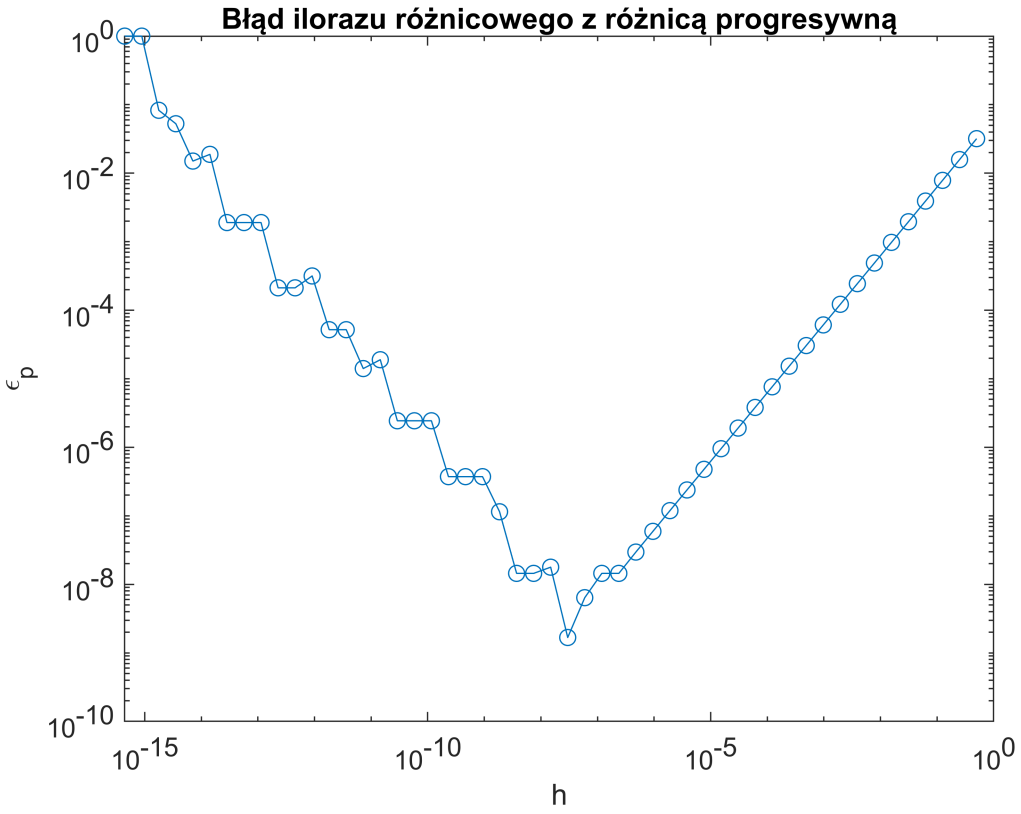
```
f3x0 =
   0.028863500386448
```

Obliczenia dla ilorazu różnicowego z różnicą progresywną

```
h0=0.5; % Początkowa długość kroku różniczkowania
H=h0*(2.^(0:-1:-50)) % Generacja wektora z malejącymi długościami kroku do 2^(-50)
```

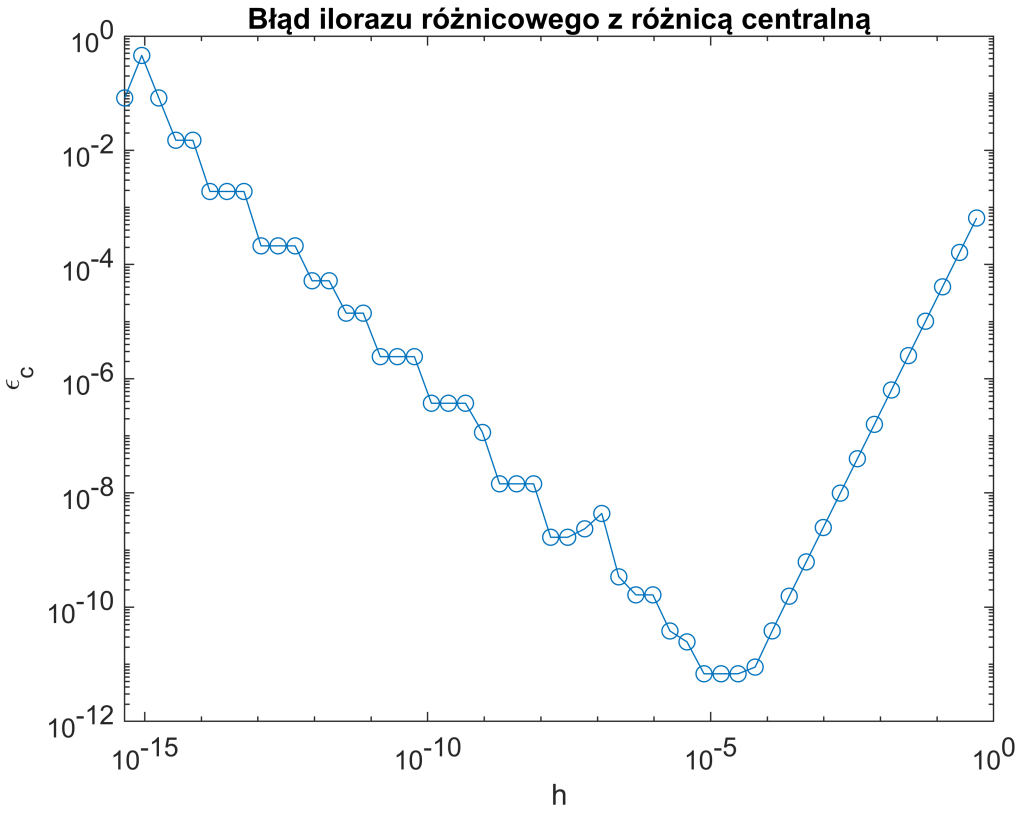
```
H = 1×51
   0.500000000000000   0.250000000000000   0.125000000000000   0.062500000000000 ···
```

```
Dp= (fun(x0+H)-fun(x0))./H; % Obliczenie ilorazu różnicowego z różnicą progresywną dla wszystki
erp= abs((Dp-f1x0)/f1x0); % Obliczenie modułu błędu względnego
loglog(H,erp,'o-') % Wykres w skali logarytmicznej
xlabel('h');ylabel('\epsilon_p');title('Błąd ilorazu różnicowego z różnicą progresywną')
```



Obliczenia dla ilorazu różnicowego z różnicą centralną

```
Dc=(fun(x0+H)-fun(x0-H))./(2*H) ; % Obliczenie ilorazu różnicowego z różnicą centralną dla wszy
erc = abs((Dc-f1x0)/f1x0); % Obliczenie modułu błędu względnego
loglog(H,erc,'o-') % Wykres w skali logarytmicznej
xlabel('h');ylabel('\epsilon_c');title('Błąd ilorazu różnicowego z różnicą centralną')
```

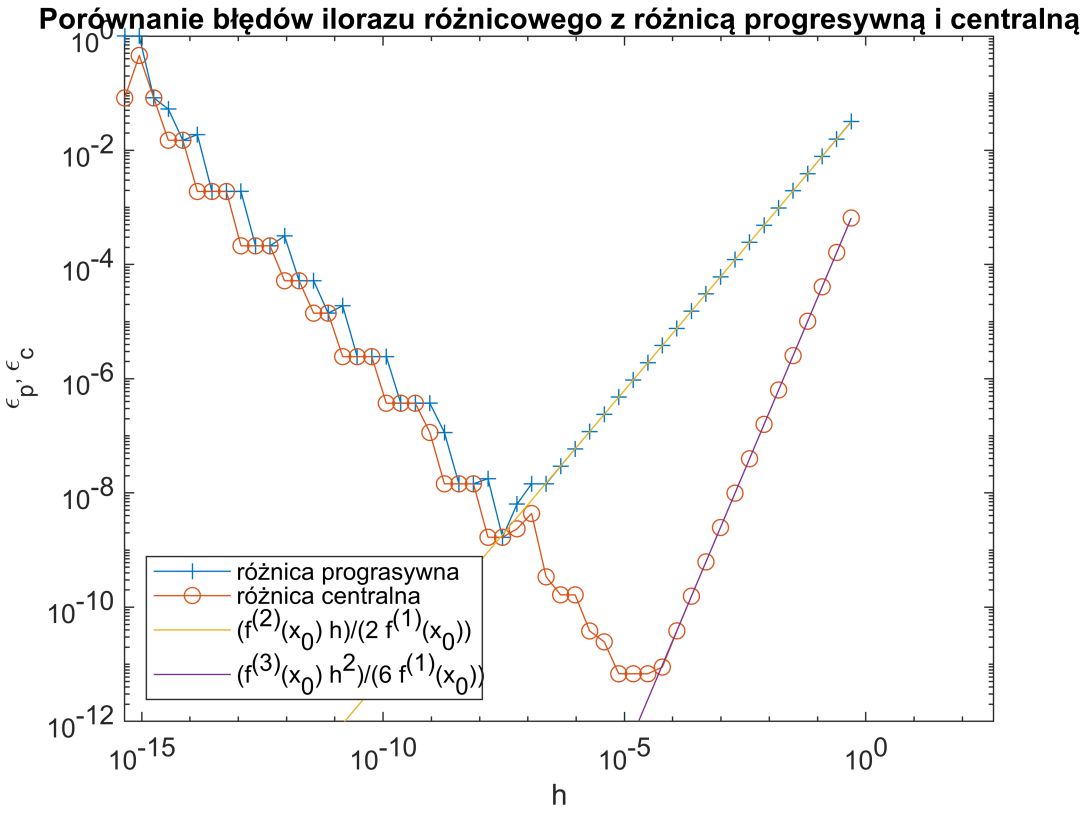


Porównanie: Narysowano te same wykresy co w poprzednich dwóch sekcjach i dodano linie, które odpowiadają przybliżonej zależności między krokiem różniczkowania i błędem obcięcia. Jak wiadomo dla błąd obcięcia dla zwykłego (progresywnego) ilorazu różnicowego wynosi $\varepsilon_{obc} \approx \frac{f''(x_0)}{2f'(x_0)}h$, natomiast dla ilorazu różnicowego z różnicą centralną $\varepsilon_{obc} \approx \frac{f'''(x_0)}{6f'(x_0)}h^2$

```
loglog(H,erp,'+-',H,erc,'o-',H,abs(f2x0/f1x0)*H/2,'-',H,abs(f3x0/f1x0)*H.^2/6,'-')
legend('różnica prograsywna','różnica centralna','(f^{(2)}(x_0) h)/(2 f^{(1)}(x_0))','(f^{(3)}(
xlabel('h');ylabel('\epsilon_p, \epsilon_c');title('Porównanie błędów ilorazu różnicowego z ró

ylim([1e-12 1.000])

xlim([0 446])
```



Porównanie obliczeń z podwójną i pojedynczą precyzją

```
Hs=single(H); % Konwersja długości kroku do pojedynczej precyzji
x0s=single(x0); % Konwersja punktu różniczkowania
f1x0s=single(f1(x0s))
```

```
f1x0s = single

   1.8472641
```

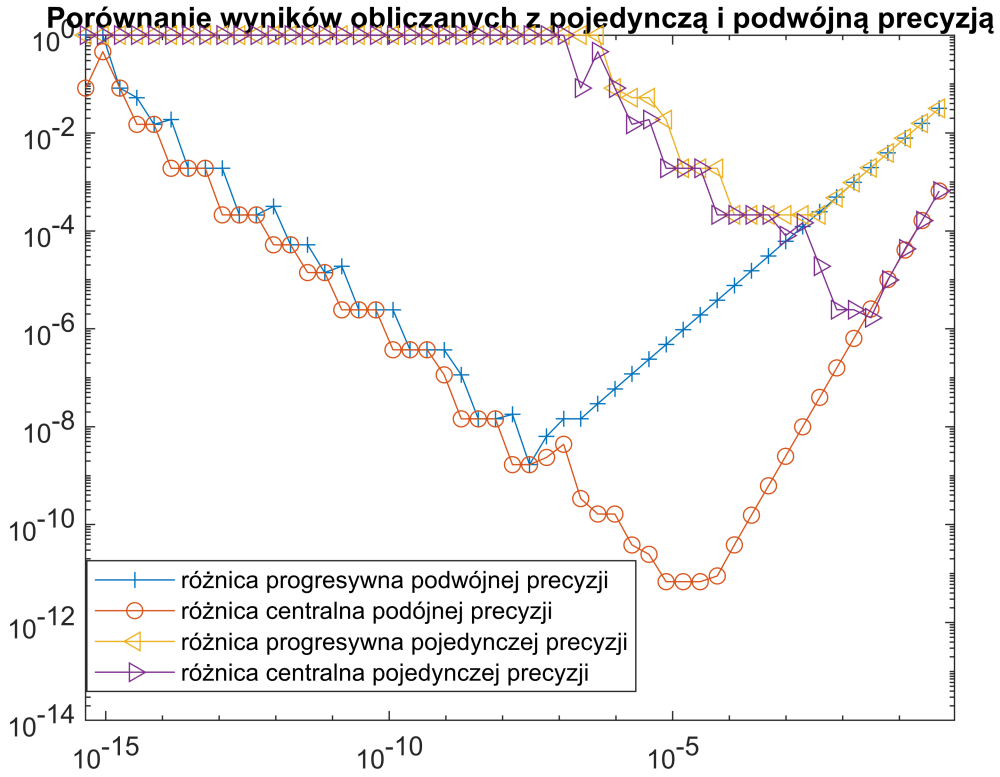
```
Dps=(fun(x0s+Hs)-fun(x0s))./Hs; % Powtórzenie obliczeń z różnicą progresywną w pojedynczej prec
erps= abs((Dps-f1x0s)/f1x0s); % Obliczenie modułu błędu względnego dla obliczeń z pojedynczą pr

Dcs=(fun(x0s+Hs)-fun(x0s-Hs))./(2*Hs) ; % Obliczenie ilorazu różnicowego z różnicą centralną dl
ercs=abs((Dcs-f1x0s)/f1x0s);

loglog(H,erp,'+-',H,erc,'o-',Hs,erps,'<-',Hs,ercs,'>-')
legend('różnica progresywna podwójnej precyzji', 'różnica centralna podójnej precyzji', ...
    'różnica progresywna pojedynczej precyzji', 'różnica centralna pojedynczej precyzji')
title('Porównanie wyników obliczanych z pojedynczą i podwójną precyzją')

ylim([1e-14 1])
legend(["różnica progresywna podwójnej precyzji","różnica centralna podójnej precyzji","różnica
legend("Position", [0.13202,0.14437,0.4875,0.15595])
```

```
xlim([0.000 0.937])
ylim([1e-14 1])
```



Obserwacje

Błąd dla pojedynczej precyzji dla zakresu liczb poniżej 1e-6 jest stały, co może wynikać z zakresu reprezentacji tych liczb. Wykres dla błędu różnicy progresywnej przyjmuje wartości nie mniejsze niż wykres dla metody różnicy centralnej. Istnieją zakresy, w których błędy danej metody pokrywają się dla liczb pojedynczej i podwójnej precyzji.

Uwagi i wnioski

Można korzystać z pojedynczej precyzji podczas obliczeń w przypadku korzystania z liczb większych od około 1e-3, ze względu na obecność błędu mniejszego o 4 rzędy wielkości w przypadku metody różnicy progresywnej i 6 rzędów wielkości w przypadku metody różnicy centralnej, względem zadanych liczb i jest zbieżny błędowi w przypadku korzystania z podwójnej precyzji.

Metoda różnicy progresywnej wykazuje się wyższym błędem względnym niemal dla każdego zakresu liczb, a największe znaczenie ma to w przypadku liczb większych od 1e-7. Oznacza to, że dzielenie przez bardzo małe liczby prowadzi do znacznego wzrostu wartości błędu względnego.

```
function y = pval1(p, x)
n=length(p)-1; % Odczytanie stopnia wielomianu
tmp=zeros(size(x)); % Zainicjowanie zmiennej do sumowania
```

```
for k=1:n+1
    tmp = tmp + p(k) * x.^(n-k+1); % Aktualizacja pomocniczej zmiennej sumacyjnej
end
y=tmp;
end

function y = pval2(p, x)
n=length(p)-1; % Odczytanie stopnia wielomianu
pow = ones(size(x)); % zainicjowanie zmiennej w której będę przechowywane kolejne potęgi (odpow
tmp = p(end)*pow; % Zainicjowanie zmiennej sumacyjnej
for k=1:n
    pow = pow .* x; % Obliczenie kolejnej potęgi
    tmp = tmp + p(end-k)*pow; % Aktualizacja zmiennej sumacyjnej
end
y=tmp;
end

function y = horner(p, x)
n=length(p)-1; % Odczytanie stopnia wielomianu
tmp = p(1);  % zaincjowanie zmiennej pomocniczej odpowiadającej zawartości najbardziej wewnętrz
for k=2:n+1
    tmp = tmp .* x + p(k); % Aktualizacja zmiennej pomocniczej tak by odpowiadała kolejnemu bar
end
y = tmp;
end
```