Radosław Czapp

Metoda Dekkera

Implementacja algorytmu w języku Julia

Spis treści

1 Opis projektu

1.1 Ogólny opis metody

Metoda Dekkera została zaproponowana przez Theodorusa Dekkera w 1969r. Bazuje ona na połączeniu metody równego podziału (bisekcji) z metodą siecznych. Podobnie jak w metodzie bisekcji rozwiązując równanie f(x)=0 musimy zainicjować metodę Dekkera dwoma punktami, np. x_0 i x_1 takimi, że $f(x_0)$ i $f(x_1)$ mają różne znaki. Jeśli funkcja f jest ciągła na przedziale [x_0, x_1] wtedy twierdzenie Darboux gwarantuje, że zero leży pomiędzy x_0 i x_1.

1.2 Opis algorytmu A

W każdej iteracji zaangażowane są trzy punkty:

- b – ostatnie przybliżenie wyniku
- c – kontrapunkt b, czyli taki punkt, w którym funkcja f ma przeciwny znak niż w punkcie b
- a – poprzednia wartość punktu a, używany jest on do wyliczania następnego punktu metodą siecznych

Metoda bisekcji natomiast używa punktów b i c tworząc punkt m pomiędzy nimi na środku przedziału. Wyliczany jest ciąg xi, którego ostatni element oznaczany jest przez x, a poprzedni przez xp. Oprócz tego jest punkt xk, który jest ostatnim punktem w ciągu, który ma różny znak niż x. Następny punkt x wyliczany jest dwoma metodami : siecznych i bisekcji i wybierany jest ten obliczony z metody siecznych jeśli leży pomiędzy punktem b (ze względów dokładnościowych z pewną poprawką) a punktem m wyliczonym z bisekcji.

Następnie badanie czy f(x) czy f(xk) leży bliżej zera i jeśli f(x) leży bliżej zera wtedy b ma wartość x a c xk, w przeciwnym razie zamiana.

1.2.1 Przykład

Funkcja $f(x)=\frac{1}{x-3}-6$ osiąga +∞ w punkcie 3.0 i zero w $x=3\frac{1}{6}$

a1=3.01

b1 = 4, f(a1) = 94

f(b1) = -5

obliczamy c1 = 3.01

a2 = 4.000000000000	b2 = 3.950000000000	c2 = 3.010000000000
a3 = 3.950000000000	b3 = 3.480000000000	c3 = 3.010000000000
a4 = 3.480000000000	b4 = 3.245000000000	c4 = 3.010000000000
a5 = 3.245000000000	b5 = 3.127500000000	c5 = 3.245000000000
a6 = 3.127500000000	b6 = 3.185075000000	c6 = 3.127500000000
a7 = 3.185075000000	b7 = 3.170992625000	c7 = 3.127500000000
a8 = 3.170992625000,	b8 = 3.166188864569	c8 = 3.170992625000
a9 = 3.166188864569	b9 = 3.166679068378	c9 = 3.166188864569
a10 = 3.166679068378	b10 = 3.166666702220	c10 = 3.166188864569
a11 = 3.166666702220	b11 = 3.166666666664	c11 = 3.166666702220
a12 = 3.166666666664	b12 = 3.166666666667	c12 = 3.166666702220
a13 = 3.166666666667	b13 = 3.166666666667	c13 = 3.166666666667

1.2.2 Własności

Dla funkcji $f(x)=(x+3)(x-1)^2$ mającej zera w punktach -3 i 1, szukając na przedziale [-4, 4/3] występują kłopoty: b zbliża się powoli jednostronnie do zera, a c ma cały czas wartość -4 jeszcze w 74- tej pętli, aż dopiero w następnej (dla double) b osiąga wartość dokładnie 1, f(b) osiąga zero, i c staje s równe b.

Innym problemem jest przypadek, gdy w pobliżu pierwiastka pochodne dążą do zera, co powoduje bardzo powolną zbieżność. Brent podaje taką funkcję: $f(x)=x*\exp(-x^{-2})$ jeśli x różne od zera i zero dla zera.

1.3 Opis algorytmu M

W tej metodzie bada się to kiedy ostatnio przedział |b-c| został zmniejszony o połowę. Tutaj ustawiana jest zmienna age. Dodatkowo, gdy age=3, wtedy wołana jest funkcja rfun, która umożliwia zbieżność szybciej niż lfun.

1.3.1 Przykład

Funkcja $f(x)=(x+3)(x-1)^2$ posiada zera w punktach 3 oraz 1 , będziemy szukać na przedziale [4, 4/3].

a0 = -4.000000000000 b0 = 1.333333333333 c0 = -4.000000000000

age = 1 lfun a2 = 1.333333333333 b2 = 1.232558139535 c2 = -4.000000000000

age = 2 lfun a3 = 1.232558139535 b3 = 1.141223295850 c3 = -4.000000000000

age = 3 rfun a4 = 1.141223295850 b4 = 1.070756096437 c4 = -4.000000000000

age = 4 bisekcja a5 = 1.070756096437 b5 = -1.464621951782 c5 = -4.000000000000

age = 1 lfun a6 = -1.464621951782, b6 = -2.732310975891 c6 = -4.000000000000

age = 1 lfun a7 = -3.366155487945 b7 = -2.732310975891 c7 = -3.366155487945

age = 1 lfun a8 = -2.732310975891 b8 = -2.953018236685 c8 = -3.366155487945

age = 2 lfun a9 = -2.953018236685 b9 = -3.007123150382 c9 = -2.953018236685

age = 1 lfun a10 = -3.007123150382 b10 = -2.999830139829 c10 = -3.007123150382

age = 1 lfun a11 = -2.999830139829 b11 = -2.999999396604, c11 = -3.007123150382

age = 2 lfun a12 = -2.999999396604 b12 = -3.000000000051 c12 = -2.999999396604

age = 1 lfun a13 = -3.000000000051 b13 = -3.000000000000 c13 = -3.000000000051

Funkcja $f(x)=\frac{1}{x-3}-6$ na przedziale [3,01; 4] potrzeba 12 kroków.

1.4 Opis algorytmu R

Identyczny jak Algorytm M z wyjątkiem fragmentu wyliczania x.

1.4.1 Przykład

Dla funkcji $f(x)=\frac{1}{x-3}-6$ na przedziale [3,01; 4] potrzeba tylko 5 kroków.

a1 = 3.010000000000 b1 = 4.000000000000 c1 = 3.010000000000

age = 1 lfun a2 = 4.000000000000 b2 = 3.950000000000 c2 = 3.010000000000

age = 2 rfun a3 = 3.950000000000 b3 = 3.480000000000 c3 = 3.010000000000

age = 1 rfun a4 = 3.480000000000 b4 = 3.245000000000 c4 = 3.010000000000

age = 1 rfun a5 = 3.245000000000 b5 = 3.166666666667 c5 = 3.245000000000

2 Kody

https://pl.wikibooks.org/wiki/Kody_%C5%BAr%C3%B3d%C5%82owe/Metoda_Dekkera

3 Harmonogram

Cel	Data
Specyfikacja projektu	17-05-2019
Wykonanie połowy projektu - zaimplementowanie algorytmu A	31-05-2019
Ukończenie projektu – zaimplementowanie algorytmów M i R	14-06-2019

4 Literatura

http://oai.cwi.nl/oai/asset/20561/20561A.pdf

https://blogs.mathworks.com/cleve/2015/10/12/zeroin-part-1-dekkers-algorithm/

https://en.wikipedia.org/wiki/Brent%27s_method#Dekker's_method