Temă Analiza algoritmilor
Upgrade Newton-Raphson

Caim Alexandru Gabriel

December 22, 2022

1 Introducere

Ecuaţiile neliniare constituie una din cele mai frecvente aplicaţii de calcul numeric care apar în cadrul activităţilor de proiectare din ingineria electrică. Un exemplu concret ar fi rezolvarea repetată a ecuațiilor de stare ale conductoarelor electrice. Multe dintre aceste probleme practice se modelează matematic, după care se aplică metode numerice.

2 Algoritmi utilizați în ecuații neliniare

2.1 Newton-Raphson

Există mulți algoritmi pentru rezolvarea ecuațiilor neliniare, însă cel mai utilizat și cel mai eficient ca număr de pași efectuați este Newton-Raphson. În teorie, acesta se bazează pe găsirea în mai mulți pași a soluției, utilizând intersecția derivatei cu axa Ox. Algoritmul care va fi prezentat în cele ce urmează reușește să micșoreze numărul de pași al acestuia, în unele cazuri ajungând chiar la jumătate.

2.2 Ideea de bază

Metoda are la bază un concept simplu, dar greu de atins, și anume micșorarea distanței dintre punctul de intersecție al derivatei cu Ox și soluția căutată. Vom considera x0' = punctul de intersecție și x0 =

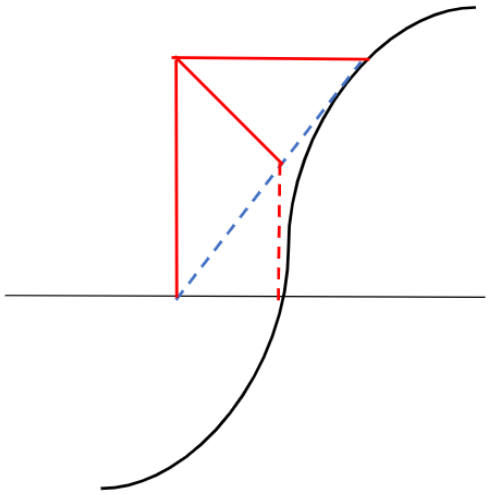

Figura 1: Concavitate

punctul inițial. Întregul algoritm se bazează pe o observație la nivel grafic. Ne dorim găsirea unui punct este mai aproape de soluție decât x0'. Avem 2 cazuri pe care trebuie să opereze algoritmul, când funția este concavă și când funția este convexă.

2.3 Concavitate

Unim dreptele y = f(x0) , x = x0' , tangenta și se formează un triunghi dreptunghic, triunghi care are ca vârfuri punctele (x0,f(x0)), (x0', 0), (x0',f(x0)). Intersecția bisectoarei din vârful unghiului de 90 de grade cu latura opusa are abscisa căutată. De ce am făcut această alegere?: Dacă curba se apropie de orizontala atunci tangenta se depărtează de soluție cu o distanță mai mică în raport cu x0 – x0' . Bisectoarea cade mai aproape de x0' și se apropie mai mult de soluție. Dacă curba se apropie de verticală atunci tangenta se departeaza de solutie cu o distanta mai mare în raport cu x0-x0'. Bisectoare cade mai aproape de x0 față de cazul precedent și se aproprie de soluție prin urmare bisectoarea urmărește acest raport fiind o variantă superioară tangentei normale pentru că exploatează mai bine spațiul x0' - soluție.

2.4 Convexitate

Construim un triunghi simetric cu cel format de dreptele y = f(x0), x=x0' și tangenta(ca cel de la concav), cele 2 triunghiuri avand pe x = x0' latură comună. Din varful ce determină un unghi drept trasăm bisectoare spre latura opusă în triunghiul construit simetric. Pe același principiu bisectoarea urmărește raportul și este o alegere mai bună.Când x0 este negativ atunci funția concavă este tratată ca una convexă și funcția con-

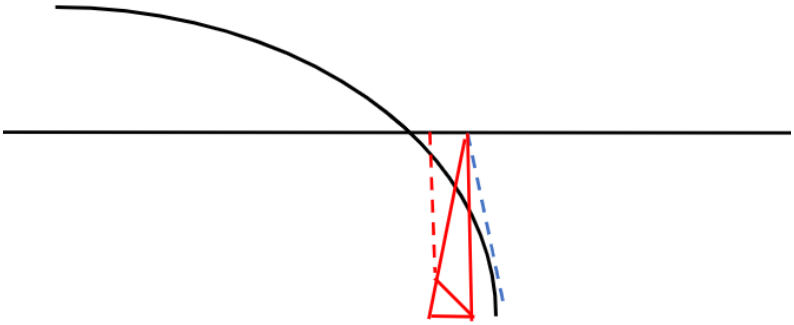

Figura 2: Convexitate

vexă ca una concavă, deoarece ne interesează această caracteristică în raport cu soluția.

3 Modul de selecție al testelor

Acest algoritm îl consider mai eficient decât Newton-Raphson deoarece efectuează un număr mai mic de pași până la atingerea preciziei dorite. Pentru a demonstra acest aspect am realizat o serie de teste care au la bază funții ce prezintă soluții, la început funcții care între punctul ales inițial și soluție prezintă o porțiune injectivă și ulterior am scos această restricție. Testele au fost alese astfel încât să obținem grafice cu o creștere rapidă, dar și grafice cu creștere lentă și să îmbine toate categoriile de funcții, chiar și pe cele trigonometrice.Principalul defect al metodei Newton-Raphson se întâlnea atunci când soluția era un punct de inflexiune, așa că am tratat și această posibilitate. Urmează să analizez complexitatea din punct de vedere al spațiului utilizat, al timpului și numărul de pași.

4 Analiza metodei și a complexității

4.1 Date de intrare

Cei 2 algoritmi sunt foarte asemănători din punct de vedere al datelor de intrare. Este necesară funcția a cărei soluție dorim să o determinăm, precum și derivata acesteia. În algorimul conceput de mine este necesar

să aflăm și semnul derivatei de ordinul 2 pentru a stabili convexitatea și concavitatea față de soluție. Pentru a nu avea un dezavantaj evident în fața lui Newton m-am folosit de legătura dintre monotonia funcției și semnul derivatei, astfel că pentru un interval mic am găsit 2 puncte și am determinat monotonia funției g, de unde a rezultat semnul derivatei de ordin 2. În teorie intervalul cu cât este mai mic cu atât corectitudinea afirmației este mai pronunțată, însă în Octave nu putem face toate operațiile dorite cu o diferență atât de mică între cele 2 puncte. Deci din punct de vedere al datelor de intrare cei doi algoritmi sunt la același nivel.

4.2 Metodă

La nivel de metodă, ambii se folosesc de reprezentarea geometrică a derivatei. Derivata într-un punct x0 ajută la construirea ecuației tangentei la grafic în punctul respectiv, fiind panta acesteia.

y - f(x0) = f'(x0) (x - x0);

Dintr-o perspectivă mai geometrică, pe un interval destul de restrâns funcția se poate confunda cu o dreaptă, mai exact cu tangenta printr-un punct al acelui interval. Dacă am vorbi despre o un polinom de gradul 1, atunci derivata ne-ar conduce direct la soluție, însă în realitate restul funcțiilor prezintă curbe în grafic. Astfel, ne folosim de derivată și de tangenta la grafic pentru a ne apropria treptat de baza curbei, adică de soluție. Această utilizare a derivatei surclasează metoda secantei sau a bisecției. O condiții suplimentară pentru a putea utiliza aceste concepte și acești algoritmi este ca funția să fie continuă și să prezinte soluții, adică graficul să intersecteze cel puțin o dată axa Ox.

Newton-Raphson se folosește de intersecția tangentei cu axa Ox și abscisa acestei intersecții devine abscisa punctului de plecare din următorul pas. Se repetă acest procedeu până se atinge precizia dorită. Acesta este în clipa de față cel mai utilizat algoritm, însă poate fi îmbunătățit. Algoritmul meu se folosește de bisectoarea descrisă anterior pentru a face orice pas mai bun, mai eficient, deoarece ne aduce mai aproape de soluție. Cei doi algoritmi împart și câteva dezavantaje, deoarece la nivel de grafic ambele se bazează pe derivată și ambele pot ajunge să intersecteze Ox în puncte foarte depărtate de soluție. De obicei asta se întâmplă pentru porțiuni de grafic neinjective.

4.3 Complexitatea temporală

Pentru a putea compara cei doi algoritmi vom face o analiză asupra diferenței ce apare la numărul de pași efectuați(numărul de accesări ale instrucțiunii while) cât și asupra diferenței la nivel de complexitatea care apare între operațiile per pas. Din punct de vedere geometric, este evident că numărul de pași este mai mic în cazul algoritmului meu.

La nivel de "greutatea" operațiilor per pas, Newton are câștig de cauză deoarece se bazează pe o singură formulă. Algoritmul meu memorează în variabile auxiliare valorile funcției, pentru ca ulterior acestea să poată fi reutilizate cu un cost mai mic. Având mai multe cazuri posibile de analizat automat sunt necesare mai multe operații, deci la acest capitol algoritmul mai bun este Newton.

Diferența dintre cei doi algoritmi la nivel de timp de execuție se poate defini ca raportul dintre aceste 2 aspecte: dacă numărul de pași este semnificativ redus atunci o alegere bună este algoritmul meu, însă dacă sunt apropiate ca pași cei 2 algoritmi tind să se apropie ca și timp și în unele cazuri cel clasic este mai eficient. Cazurile prezentate anterior, când diferența de pași este mică se întâlnesc atunci când valoare inițială x0 este ales aproape de soluție. În practică o astfel de alegere este aproape imposibilă, aceasta este una dintre marile probleme ale lui Newton. Este adevărat că atinge o convergență ideală într-un timp bun, doar că ceea ce îl face superior altor metode este eficiența pe intervale mici, când graficul se confundă cu o dreaptă(de aceea și în algoritmul propriu dacă se atinge o anumită acuratețe folosim formula lui Newton). Deci, o alegere proastă dezavantajează Newton-Raphson iar la nivel intuitiv se poate vedea un upgrade în planul vitezei de execuție conferit de noul algoritm. Complexitatea temporală este O(nsteps), unde nsteps este numărul de pași efectuat de algoritm.

4.4 Complexitatea spațială

Din punct de vedere al memoriei utilizate nu întâlnim vectori sau matrici, avem o diferență între numărul de variabile utilizate, dar este infimă pentru tehnologia din prezent, așă că nu ne axăm mult pe acest aspect.

4.5 Avantaje și dezavantaje

Algoritmul meu aduce un singur dezavantaj, care poate fi ignorat și constă în șansa ca intervalul dintre cele două numere alese pentru a testa convexitatea să cuprindă și alt tip de interval, însă dacă se poate lucra cu multe zecimale șansa tinde spre nul. Avantajul pe care îl oferă este viteza de execuție și numărul de pași. Newton are ca avantaj implentarea facilă și ca dezavantaj pierderea treptată e eficienței odată cu depărtarea de soluție a valorii alese inițial.

5 Testare

5.1 Modalitate construire teste

Am construit o serie de teste, 25 teste au generat random valorile lui x0 și în alte 5 le-am ales pentru a testa comportamentul celor 2 algoritmi. Am definit funcția și derivata acesteia și am ales o serie de funcții alături de derivatele acestora pentru a le putea testa în vederea găsirii soluțiilor. În testele făcute de mine am căutat să testez și cazurile sensibile ale algoritmului Newton-Raphson, mai exact punctele de inflexiune, dar și punctele forte (funcțiile periodice de perioadă mică). Am ales grafice cu creștere rapidă, dar și cu creștere lentă și în final am ales și o funcție cu un interval neinjectiv și cu caracter impredictibil. Valorile lui x0 au fost atât negative cât și pozitive, aproape de soluție cât și departe de aceasta.

Trebuie să ținem cont că în cazul lui Newton se fac 2 apeluri de funcții, iar in cazul algoritmului meu 3. Astfel, pentru a echilibra balanța și pentru a oferi o testare cât mai corectă vom adăuga în funția derivată încă o serie de if-uri. Am făcut asta deoarece dacă am fi testat o singură funție totul ar fi fost corect(o apelare suplimentară și o operație), însă dacă pentru fiecare funție sunt operații inutile atunci este clar că o apelare în plus va defavoriza algoritmul meu.

5.2 Componente

Componente:
Procesor : Intel(R) Core(TM) i7-7500U CPU @ 2.70GHz 2.90GHz
RAM: 8,00GB(7,89 utilizabili)
Spațiu de stocare disponibil: 467,12GB

5.3 Rezultate funcții random

Test	Algoritmul meu	Newton-Raphson
1	0.57115173339844	1.12577056884766
2	4.35030364990234	8.58943176269531
3	16.66441345214844	18.74774169921875
4	7.82871246337891	9.04411315917969
5	2.35318756103516	2.75383758544922
6	6.88893127441406	8.52847290039062
7	4.14431762695312	5.06471252441406
8	1.44344329833984	1.86517333984375
9	10.57324218750000	12.92423248291016
10	3.22610473632812	3.95349884033203
11	10.37314605712891	12.70959472656250
12	17.31843566894531	21.58863067626953
13	17.89089202880859	22.35234069824219
14	18.20291900634766	23.24670410156250
15	15.41975402832031	21.90321350097656
16	6.42308807373047	7.03510284423828
17	0.68428039550781	0.83292388916016
18	9.89266967773438	13.62412261962891

5.4 Rezultate funcții speciale

Test	Algoritmul meu	Newton-Raphson
19	7.36016082763672	10.06040954589844
20	5.47340393066406	7.37424468994141
21	1.82593536376953	1.18252563476562
22	11.16456604003906	11.55226898193359
23	8.43968200683594	8.31192016601562
24	22.28148651123047	20.25606536865234
25	21.11164093017578	19.17430114746094

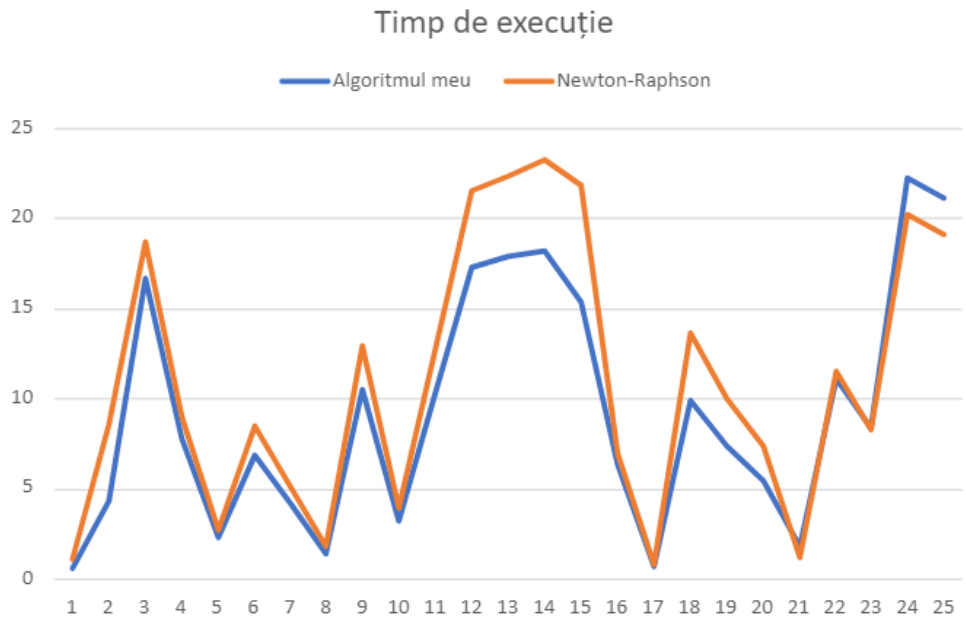

Figura 3: Grafic rezultate valori random

5.5 Rezultate valori speciale

Test	Algoritmul meu	Newton-Raphson
1	0.00674438476562	0.00516510009766
2	0.03493499755859	0.03415679931641
3	0.00292968750000	0.00253295898438
4	0.03324890136719	0.01956939697266
5	0.01103210449219	0.01332092285156

5.6 Concluzii

În urma rulării testelor am remarcat următoarele aspecte:

-dacă diferența la nivel de pași este mică în raport cu numărul de pași total, adică acest raport să tindă către 0, atunci Newton este mai rapid

-dacă acel raport tinde către 1/2 atunci algoritmul meu are câștig de cauză

-ambele nu converg pentru puncte de inflexiune

-în cazul graficelor cu porțiuni injective nu s-a putut determina clar care este algoritmul mai bun, deoarece multe teste nu au atins acuratețea dorită

-pentru valori inițiale care au abscisa în apropierea soluției Newton-Raphson este mai bun(cazul funțiilor periodice cum ar fi sin(x))

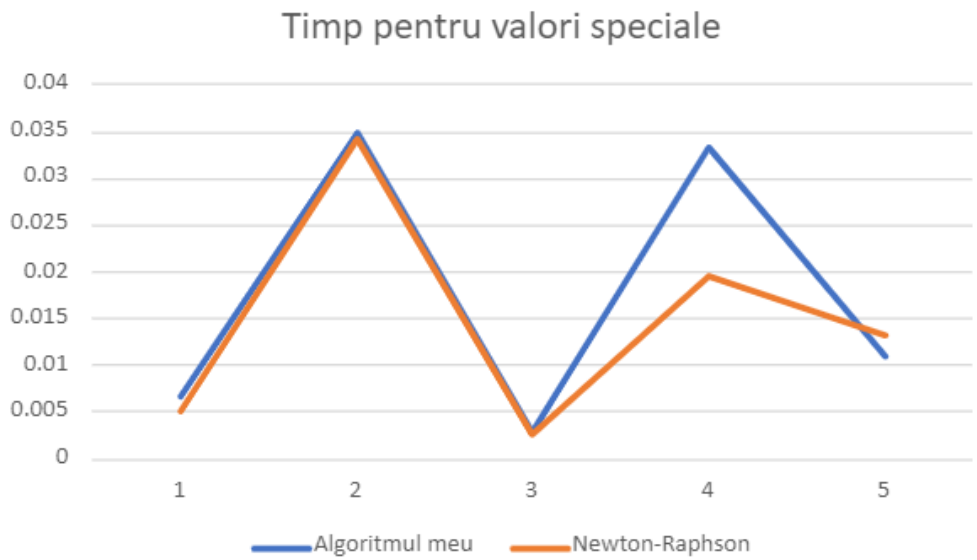

Figura 4: Grafic rezultate valori speciale

-ambele nu converg într-un număr de pași util când vine vorba de puncte de inflexiune -per total algoritmul meu este superior ca viteză

6 Utilizare în practică

În concluzie, consider că neavând detalii clare despre o porțiune exactă unde se poate găsi soluția algoritmul meu este mult mai eficient decât Newton-Raphson. În cazul în care am știi o astfel de porțiune am putea miza pe Newton. În practică o astfel de aproximare inițială este costisitoare și în final nu constituie cea mai bună opțiune.

7 Referințe

https://www.sciencegate.app/keyword/763525
https://www.sciencedirect.com/science/article/pii/S0377042700004350
https://www.math.uni-bielefeld.de/documenta/vol-ismp