UNIVERZA V MARIBORU
EKONOMSKO-POSLOVNA FAKULTETA

Diplomsko delo

STABILNO RAČUNANJE ENAKIH KREDITNIH ANUITET

Stable Computation of Annuity Loan Payments

Kandidat: Aleksander Golob
Študijski program: Poslovna ekonomija
Študijska usmeritev: Finance in bančništvo
Mentor: doc. dr. Igor Perko
Jezikovno pregledala: Špela Brajer, prof. slov.
Študijsko leto: 2015/2016

Maribor, april 2016

ZAHVALA

Zahvalil bi se rad svojim bližnjim za vso njihovo podporo, spodbude in potrpljenje. Posebej očetu, hvala.

Zahvalil bi se rad tudi mentorju, doc. dr. Igorju Perku, za usmeritev k izbrani tematiki ter za vso prejeto pomoč in podporo tako pri nastajanju kot tudi med pisanjem diplomskega dela.

POVZETEK

V diplomskem delu predstavljamo, kako stabilno računati enaka odplačila anuitetnega kredita s pomočjo računalnika. Obrazec za direktno računanje geometričnega zaporedja – znotraj formule za računanje anuitet – pri nizki obrestni meri vsebuje odštevanje dveh približno enakih števil, kar pomeni veliko relativno napako pri računanju z računalnikom v premični piki. Zato je smiselno uporabiti *postopni način* računanja členov geometričnega zaporedja, saj ta ne vsebuje odštevanja. Na ta način je število operacij sicer vezano na število kapitalizacijskih obdobij, vendar dobimo s tem direktno stabilen algoritem. Člene lahko računamo bodisi iz prvega člena geometričnega zaporedja bodisi rekurzivno. Do seštevka, ki predstavlja geometrično zaporedje lahko pridemo tudi z računanjem kombinacij in obrestnih mer. Zato preuredimo enačbo, ki jo uporablja Bohte (1995, str. 81) pri analizi zaokrožitvenih napak. Numeričen rezultat takšnega računskega postopka je eno izmed sosednjih števil točnega rezultata, s čimer dobimo enakovredno rešitev k Excelovi PMT funkciji. Prav tako enakovreden, vendar v primerjavi z Bohtejevim veliko bolj učinkovit, pa je algoritem, ki oblikuje izračune iz skupin obrestnih mer v dvojiških korakih in jih nato poveže.

Ključne besede: finančna matematika, krediti, geometrično zaporedje, numerične metode, analiza zaokrožitvenih napak, Excel-VBA, Octave (programski jezik)

ZUSAMMENFASSUNG

In der Diplomarbeit wird die stabile Berechnung von gleichen Ratenzahlungen eines Annuitätendarlehens dargestellt. Die Gleichung zum direkten Rechnen der Summe der geometrischen Folge – innerhalb der Formel zur Berechnung von Ratenzahlungen – enthält bei einem niedrigen Zinssatz die Subtraktion von zwei ungefähr gleich großen Zahlen, was einen großen relativen Rundungsfehler beim Rechnen mit dem Computer in der Gleitpunktarithmetik bedeutet. Es ist deshalb ratsamer, die Summe der geometrischen Folge *schrittweise* zu errechnen, weil dieser Vorgang keine Subtraktion beinhaltet. Auf diese Art ist zwar die Anzahl der Rechenoperationen an die Anzahl der Kapitalisierungszeiträume gebunden, jedoch erhält man damit einen vorwärtsstabilen Algorithmus. Einzelne Glieder der geometrischen Folge können entweder aus dem ersten Glied oder auch rekursiv errechnet werden. Die Summe, die die geometrische Folge darstellt, kann auch über das Rechnen von Kombinationen und Zinssätzen ermittelt werden. Dazu wird die Gleichung genommen und modifiziert, die Bohte (1995, S. 81) bei der Rundungsfehleranalyse anwendet. Das numerische Resultat dieser modifizierten Gleichung ist eine der Nachbarzahlen der genauen Lösung, womit eine gleichwertige Lösung zur Excel PMT-Funktion erhalten wird. Ebenfalls gleichwertig, aber gegenüber dem von Bohte noch viel effizienter ist ein Algorithmus, der die Berechnungen aus Gruppen von Zinssätzen in Zweierschritten formt und sie dann verbindet.

Schlüsselwörter: Finanzmathematik, Kredite, geometrische Folge, numerische Methoden, Rundungsfehleranalyse, Excel-VBA, Octave (Programmiersprache)

KAZALO

KAZALO SLIK

KAZALO TABEL

KAZALO ALGORITMOV

1 UVOD

Za posameznike, podjetja in druge gospodarske enote izven bančnega sektorja srečanje s periodičnimi denarnimi tokovi najpogosteje pomeni odplačevanje kredita (Čibej & Ferbar Tratar, 2012, str. 91). Kreditojemalec oziroma dolžnik najeti kredit običajno odplača z več obroki, ki jim pravimo anuitete (Marovt & Breznik, 2014, str. 82). V diplomskem delu bomo obravnavali anuitetni način odplačevanja, kar pomeni da so anuitete za celotno obdobje trajanja kredita enake (ibid., str. 85). Ta način je značilen za kreditno poslovanje bank s fizičnimi osebami (ibid.).

1.1 Opis področja in opredelitev problema

Za računanje zneskov anuitet, s katerimi kreditojemalec odplačuje posojilo, imamo na voljo več formul. Ko s pomočjo računalnika izvajamo izračune po teh formulah, pa dobimo rešitve v numerični obliki, ki niso več nujno enake njihovim analitičnim rešitvam.

Bolj je obrestna mera blizu 0 %, večje je odstopanje (z računalnikom) izračunanega rezultata glede na analitično izračunano vrednost. Če pri takšnem računanju za vhodne spremenljivke uporabimo glavnico v višini $10^{\mathbb{Z}}$, število obrokov 1 ter poljubno (pozitivno) obrestno mero, lahko tudi brez preizkusa sklepamo na točnost izračuna.

Ker računamo v aritmetiki s premično piko, nastopi (morebitna) napaka pri računanju anuitete za različne vrednosti glavnice – ob sicer nespremenljivih pogojih (enaki obrestni meri in enakem številu obrokov) – zmeraj na enaki števki v rezultatu. To sicer pomeni, da je lahko pri relativno nizki glavnici napaka tako majhna, da je ni mogoče izraziti v znesku določene valute in bi potemtakem ne imela realnega pomena. Vendar je po drugi strani lahko napaka v izračunu pri relativno visoki glavnici nezanemarljiva.

1.2 Namen, cilji in hipoteze raziskave

Diplomsko delo pišemo z namenom predstavitve postopkov za stabilno računanje periodičnih enakih odplačil anuitetnega kredita s pomočjo računalnika. Poudariti je potrebno, da z Excelovo PMT funkcijo že imamo računski postopek, ki izpolnjuje[1] zahteve po stabilnem računanju, pri čemer ne poznamo algoritma, ki ga funkcija uporablja.

Na začetku diplomskega dela bomo opredelili kaj so krediti ter katere anuitete so podlaga raziskave. Obrazložili bomo načrt odplačila kredita ter iz katerih postavk je sestavljen. Pri tem bo poudarek na pomenu geometričnega zaporedja, ki ga tvorijo razdolžnine v njem.

[1] Trditev temelji na tem, da smo opravili računanje "peš", avtomatizirano računanje "peš" v Excelovi preglednici s pomočjo procedure v VBA programskem jeziku ter računanje s pomočjo računala (v "znanstvenem" načinu) operacijskega sistema Windows.

Ker se izračuni anuitet dandanes izvajajo pretežno z računalniki, bomo predstavili numerično računanje. In ker se izračuni v računalnikih izvajajo s končno natančnostjo, lahko pričakujemo napake. Zato bomo rezultate izračuna ovrednotili.

Predstavili bomo formule iz literature, ki se uporabljajo za računanje kreditnih anuitet. Na podlagi načela o ekvivalenci glavnic bomo posamezne formule uvrstili v skupno splošno formulo (pri čemer obstajata dve različici). Iz te enotne oblike zapisa bomo potem lažje sklepali na primernost izbire posamične formule v določenih okoliščinah.

Bohte (1995, str. 128) piše, da si je potrebno v izogib nevarnosti nestabilnega računanja vsak izraz s stališča numerične stabilnosti ogledati in ga morebiti predelati, saj morda v obstoječi obliki ni primeren za računanje. Tako bomo dele formul, kjer gre iskati vzrok za netočnost izračuna anuitet, računali postopoma. Opravili bomo oceno napak za tako dobljene rezultate. Na koncu dela bomo opravili primerjavo obravnavnih računskih metod. Primerjali bomo točnost rezultata ter čas, ki je potreben za izračun.

1.3 Predpostavke in omejitve

Za računanje anuitet bomo uporabili glavnico kredita (znesek posojila oz. dolga), obrestno mero na kapitalizacijsko obdobje ter število obrokov, potrebnih za odplačilo kredita.

Znesek kredita je lahko poljubna neničelna vrednost. Obrestna mera je pozitivna, nespremenljiva in izračunana po dekurzivni metodi. Število obrokov je dano kot pozitivno in celoštevilsko. Kredit mora biti do konca obdobja trajanja odplačan (prihodnja vrednost mora biti 0). Časovna obdobja med odplačili so enakomerna.

Računali bomo anuitete, ki temeljijo na enakih plačilih na obdobje (anuitetni način) in ki se plačujejo ves čas trajanja kredita.

1.4 Predvidene metode raziskovanja

V poglavju o kreditih ter v poglavju o numeričnem računanju bomo uporabili analizo literature. Formule iz literature in virov bomo uvrstiti med dve splošni formuli, ki bodo vsebovale niz obstoječih formul. Predelali bomo izraze v delih formul, ki vsebujejo odštevanje.

Nato bomo uporabili primerjalno analizo, ko bomo računali anuitete in ko bomo prikazali točnost izračunov. Rezultate izračunov bomo ovrednotili s pomočjo direktne analize zaokrožitvenih napak. Postopki računanja bodo predstavljeni v obliki matematičnih algoritmov znotraj besedila, h katerim bomo v prilogah dodali še opisne algoritme za računalnik. Priloge vsebujejo tudi implementirane algoritme v računsko orodje Excel – preko vgrajenega programskega jezika VBA – ter v program GNU Octave (v nadaljevanju Octave), ki ga uporabljajo predvsem matematiki. Slednji v primerjavi z Excel/VBA računa z aritmetiko, ki ga opredeljuje standard mednarodnega združenja inženirjev elektrotehnike in elektronike (Institute for Electrical and Electronics Engineers – v nadaljevanju IEEE).

2 KREDITI

Pri kreditu [ožje] oz. posojilu gre za določeno vsoto denarja, ki jo finančna organizacija odstopi kreditojemalcu za določen čas. Uporabnik mora dobljeni znesek skupaj s pripadajočimi obrestmi [kot ceno za izposojen denar] vrniti v času in na način, kot je določeno v pogodbi (Čibej & Ferbar Tratar, 2012, str. 7). Izraza posojilo in kredit sicer nista sopomenki, ampak je odnos, ki se pojavi z najetjem posojila, samo ena od možnih oblik kreditnega odnosa (ibid.).

2.1 Osnove obrestnega računa

Banke v grobem delijo kredite na kratkoročne in dolgoročne, meja med obema tipoma je eno leto kot čas dokončnega poplačila obveznosti (ibid.).

Obresti pomenijo nadomestilo za uporabo določenega zneska denarja, ki ga kreditodajalec (posojilodajalec) za določen čas prepusti kreditojemalcu (posojilojemalcu). Obresti so funkcija treh spremenljivk (Čibej & Ferbar Tratar, 2012, str. 7):

- izposojenega zneska (glavnice, G [Dolga, D_0], ali kapitala);
- časa obrestovanja (v dneh – d, mesecih – m, redkeje v letih – l) .
- obrestne mere p (kot tistega sorazmernostnega faktorja, ki pove, koliko denarnih enot nadomestila plačamo za vsakih 100 denarnih enot glavnice, ki smo jo uporabljali eno kapitalizacijsko obdobje, to je obdobje med dvema zaporednima pripisoma oziroma obračunoma obresti).

Osnovna kapitalizacijska doba je leto, vse ostale različice pa so na enega od možnih načinov izpeljane iz podatkov za osnovno obdobje (ibid., str. 8). Sam pridevnik ima koren v besedi kapital, kar opozarja na drugo interpretacijo kapitalizacijske dobe: po preteku kapitalizacijskega obdobja, v tem času nastale obresti zaživijo kot samostojen kapital, na katerega se obračunavajo obresti (ibid.).

2.1.1 Dekurzivno in anticipativno obrestovanje

Pri kreditnih poslih lahko vsaj načeloma srečamo tako dekurzivno kot anticipativno obrestovanje (ibid., str. 9). Načina se razlikujeta z vidika trenutka, v katerem obračunavamo obresti oz. kdaj dospeva (zapade) glavnica, ki je osnova za celoten izračun obresti (ibid.).

Pri prvem načinu, ki je praksa, obračunavamo obresti po preteku ustreznega obdobja, torej za nazaj (ibid.). Take obresti imenujemo dekurzivne obresti, postopek pa dekurzivno obrestovanje (ibid.). Drugi način je anticipativno obrestovanje, pri katerem se anticipativne obresti obračunajo in odvzamejo že na začetku obrestovalnega obdobja (kapitalizacijske dobe) (ibid.).

Izhodišče za računanje obresti je torej

- pri dekurzivnem obrestovanju začetna vrednost glavnice;
- pri anticipativnem obrestovanju pa njena končna vrednost, to je vrednost ob koncu kapitalizacijskega obdobja (ibid.).

Pri nominalno enaki obrestni meri je za kreditojemalca anticipativno obrestovanje v splošnem dražje kot dekurzivno (ibid.).

2.1.2 Načelo ekvivalence glavnic

Ne glede na to, katerega od možnih načinov obrestovanja uporabljamo, se moramo zavedati, da zneskov, ki dospevajo v različnih trenutkih, ne moremo neposredno primerjati (ibid.). Namesto o dospevanju zneskov zlasti v bančni praksi dostikrat govorijo o njihovem valutiranju (ibid.). Valuta nekega dogodka v finančnem poslovanju in zneska, ki ustreza temu dogodku, je datum, ko proučevani dogodek nastopi: "valuta obroka" je na primer datum, ko mora kreditojemalec plačati obrok (ibid.). Pri tem ni nujno, da se valuta zneska ujema z datumom dejanskega pretoka sredstev ali celo knjiženja (ibid., str. 10).

Zneska, ki dospevata v različnih trenutkih, naredimo primerljiva s tem, da oba preračunamo na isti termin (ibid.). Najpogosteje vzamemo za trenutek, na katerega reduciramo zneske, kar termin, ko dospeva eden od njiju: znesek, ki dospeva prej, naobrestimo za čas do dospetja drugega zneska (glavnici v tem primeru dodamo ustrezne obresti), ali pa znesek, ki dospeva prej razobrestimo za isti čas (ibid.).

Ker posamezni zneski, ki sestavljajo proučevani finančni tok, dospevajo v različnih trenutkih, je namesto o enakosti zneskov oziroma glavnic smiselno govoriti samo o njihovi enakovrednosti ali ekvivalenci (ibid.). Dve glavnici sta ekvivalentni, če postaneta enaki po preračunu na skupen termin (ibid.).

Slika 1: Načelo ekvivalence glavnic

Vsota vseh vplačil (reduciranih na neki skupni termin)	**=**	**Vsota vseh izplačil** (reduciranih na isti skupni termin)

Vir: (Čibej & Ferbar Tratar, 2012, str. 10).

Za boljšo nazornost pri prikazovanju dinamike prilivov in odlivov si pomagamo s številsko premico, na kateri označimo posamezna obdobja in valute, ko dospevajo vplačila in izplačila (ibid.). Iz enačbe, ki jo napišemo v skladu s skico in načelom ekvivalence glavnic, potem z algebrskimi operacijami najprej izluščimo neznano količino, šele nato vstavimo konkretne numerične vrednosti znanih količin in izračunamo rešitev (ibid.).

2.2 Obrestnoobrestni račun

Osnovna značilnost obrestnoobrestnega računa je načelo kapitalizacije obresti. Obresti ne računamo samo od prvotne glavnice, ampak tudi od vseh obresti, nastalih v preteklih kapitalizacijskih obdobjih (ibid., str. 29). Vsebinsko gledano kapitalizacija obresti pomeni, da se po preteku kapitalizacijskega obdobja v tem obdobju nastale obresti transformirajo v kapital in se od tega trenutka dalje oplajajo po enaki obrestni meri, kot se to dogaja s prvotno glavnico in vsemi obrestmi iz prejšnjih obdobij.

2.2.1 Dekurzivno obrestovanje in obrestovalni faktor

Če je današnja vrednost glavnice G_0 denarnih enot, imamo po preteku enega kapitalizacijskega obdobja (ibid., str. 30)

$$G_1 = G_0 + G_0\,p = G_0(1 + p).$$

Izraz $1 + p$ se pogosto pojavlja, zato mu damo posebno oznako (ibid.)

$$r = 1 + p \tag{1}$$

in mu zaradi njegove vloge rečemo dekurzivni obrestovalni faktor (ibid.). Iz gornjega obrazca lahko potem dobimo novo obliko (ibid.)

$$G_1 = G_0\,r.$$

Če nadaljujemo, dobimo za vrednost glavnice po preteku dveh kapitalizacijskih obdobij (ibid.)

$$G_2 = G_1 + G_1\,p = G_1(1 + p) = G_1\,r.$$

Če upoštevamo, kako smo izračunali G_1, dobimo (ibid.)

$$G_2 = (G_0\,r)r = G_0\,r^2.$$

Z nadaljevanjem izpeljave dobimo po korakih posebne primere splošnega obrazca za glavnico po n kapitalizacijskih obdobjih obrestnega obrestovanja (ibid.):

$$G_n = G_0\,r^n \qquad (n = 1, 2, \ldots), \tag{2}$$

ki pove, da pri obrestnem obrestovanju glavnica narašča kot *geometrično zaporedje*, kar v zveznem času pomeni eksponentno rast (ibid.). Prvi člen tega zaporedja je začetna vrednost glavnice G_0, njegov količnik pa je obrestovalni faktor r (ibid.).

2.3 Kreditni posli

Za posameznike, podjetja in druge gospodarske enote izven bančnega sektorja srečanje s periodičnimi denarnimi tokovi najpogosteje pomeni odplačevanje kredita (Čibej & Ferbar Tratar, 2012, str. 91). Gre za najpogostejšo obliko poslovnih odnosov, pri katerih se pojavljajo obresti (ibid., str. 91).

Izraz amortizacija kredita pomeni njegovo odplačevanje, amortizacijski načrt je potemtakem samo bolj učeno ime za odplačilni načrt ali načrt vračanja posojila (ibid.). Pri posojilnih poslih lahko vsaj načelno srečamo tako dekurzivno kot anticipativno obrestovanje (ibid.), vendar je slednja v naši praksi izjemno redka (ibid.).

Kreditojemalec oziroma dolžnik najeti kredit običajno odplača z več obroki, ki jim pravimo anuitete (Marovt & Breznik, 2014, str. 82). V diplomskem delu bomo obravnavali anuitetni način odplačila, kar pomeni da so anuitete za celotno obdobje trajanja kredita enake[2] (ibid.). Ta način je značilen za kreditiranje fizičnih oseb (ibid.).

2.3.1 Osnovni pojmi

Zneske, s katerimi dolžnik amortizira posojilo, v praksi običajno ne glede na dolžino obdobja med dvema zaporednima obrokoma bančniki imenujejo *anuitete*, čeprav je ta izraz, ki izvira iz latinske besede anno (leto) najprej pomenil samo letna vračila (ibid.). Vsaka anuiteta je v bistvu sestavljena iz dveh delov, iz *obresti* in *razdolžnine* (ibid.).

Obresti se računajo za obdobje med dvema zaporednima anuitetama in od osnove, ki jo predstavlja ostanek dolga po prejšnjem vračilu, če le ni posebej določeno drugače (ibid.). Ko od zneska anuitete odštejemo obresti, dobimo znesek "neto odplačila" oziroma razdolžnino (ibid.). Za ta znesek se dejansko zmanjša dolg, ko je plačana anuiteta. Zakonodaja posebej zahteva, da se iz dolžnikovega plačila najprej pokrijejo obresti (in morebitni drugi stroški), šele če kaj ostane, se posledično zmanjša tudi stanje dolga (ibid.).

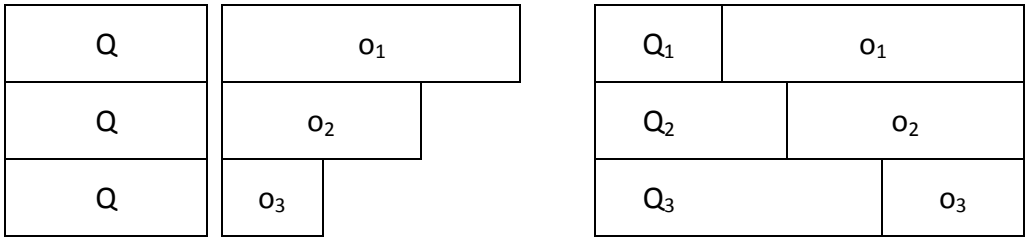
Banke uporabljajo dva načina za določanje velikosti zneska, ki ga kreditojemalec periodično vrača (ibid.). Bistvena razlika med načinoma je v tem, kaj je izhodišče za izračun tega zneska (ibid.).

Pri prvem, *obročnem načinu*, ki je značilnost bančnega kreditnega poslovanja s podjetji in drugimi pravnimi osebami, je fiksen znesek razdolžnine (toliki del začetnega dolga, kolikor je obrokov), na ta znesek pa se vsakič dodajo obresti (ibid.). Pri tem načinu seveda pada skupni znesek, ki se periodično plačuje, največja je prva anuiteta, ko se na vedno enak znesek razdolžnine doda maksimalni obrestni znesek, obračunan od celotnega začetnega stanja dolga (ibid.).

Pri drugem, *anuitetnem načinu*, ki se z nekaj izjemami uporablja pri kreditih fizičnim osebam, je izhodiščna predpostavka, da bo kreditojemalec odplačal posojilo z določenim številom enakih zneskov (ibid.). Znotraj tako določenega zneska – anuitete – se potem spreminja struktura, povečuje se delež razdolžnine in zmanjšuje delež obresti, ki seveda padajo, saj se zaračunavajo od vedno manjšega ostanka dolga (ibid.).

[2] Višina anuitete se sicer lahko tudi spremeni, vendar se bomo omejili le na delo z enakimi anuitetami.

Slika 2: Primerjava obročnega in anuitetnega izračuna za p = 50 % in n = 3



Opombe: Q = enake razdolžnine pri obročnem kreditu; Q_1, Q_2, Q_3, = razdolžnine pri anuitetnem kreditu; o_1, o_2, o_3 = obresti posameznega kapitalizacijskega obdobja.

Prirejeno po: (Čibej & Ferbar Tratar, 2012, str. 95).

2.4 Anuitetni način

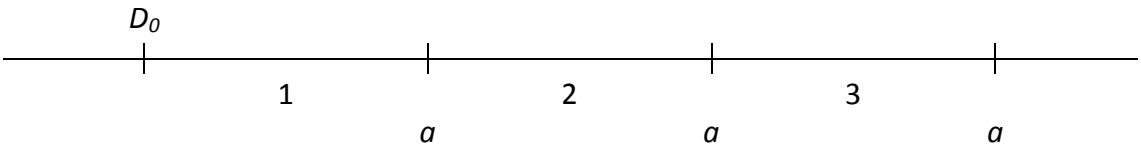
Anuitetnemu načinu bi lahko tudi rekli "način s fiksno anuiteto" (Čibej & Ferbar Tratar, 2012, str. 94). Izraz je nekoliko zavajajoč, ali bolje, točen je samo v stabilnih razmerah, kjer se pogodbeni stranki dejansko lahko za nekaj let vnaprej dogovorita za fiksno velikost posameznega periodičnega vplačila (ibid.). V drugih primerih se velikost tega zneska lahko občasno spremeni (ibid.)

- zaradi odpravljanja inflacijskih problemov (vključevanja revalorizacijskih mehanizmov),
- ker je vnaprej dogovorjeno povišanje obrestne mere po določenem številu obrokov,
- ali iz kakega drugega razloga.

2.4.1 Načrt odplačevanja kredita (amortizacijski načrt)

Pri postopku za sestavljanje amortizacijskega načrta moramo v prvem koraku izračunati oz. določiti višino anuitete (ibid., str. 95). Da bi jo izračunali, si najprej narišimo številsko premico in zapišimo ustrezno enačbo, pri čemer vse zneske reduciramo na konec [tretjega] leta, ko dospeva zadnja anuiteta (ibid.):

Slika 3: Dinamika prilivov in odlivov pri anuitetnem kreditu za 3 obrestovalna obdobja



Opombe: D_0 = dolg; a = anuiteta; 1, 2, 3 = zaporedna številka obrestovalnega obdobja.

Zadnja anuiteta dospeva natanko ob koncu zadnjega [tretjega] leta in se zato sploh ne obrestuje, najzgodnejša pa se je obrestovala [dve] leti, tako da je (ibid.)

$$D_0(1+p)^3 = a + a(1+p) + a(1+p)^2 = a\frac{(1+p)^5 - 1}{p}.$$

Izraz rešimo na neznano anuiteto a in upoštevamo, da je [obrestna mera] pod našimi pogoji enaka 50 % in D_0=1000. Potem je

$$a = \frac{D_0 \times (1+p)^n}{\left(\frac{(1+p)^n - 1}{p}\right)} = \frac{1000 \times (1+50\%)^3}{\left(\frac{(1+50\%)^3 - 1}{50\%}\right)} = 710,53, \qquad p \neq 0$$

če rezultat zaokrožimo na dve mesti (ibid.).

Pri nadaljevanju računanja in vnašanju v tabelo pazimo na to, da je zdaj fiksirani podatek pravkar izračunani znesek anuitete (ibid.). Od te anuitete odštejemo obresti preteklega ostanka dolga in s tem dobimo razdolžnino za tekoče leto (ibid.). Za to razdolžnino se dejansko zmanjša naša obveznost (ibid.). Za zgled izračunajmo podatke za prvo "zaresno" vrstico amortizacijskega načrta (ibid.)

$a_1 = 710,53$

$o_1 = 1000 \cdot 50\% = 500$

$Q_1 = a_1 - o_1 = 710{,}53 - 500 = 210{,}53$

$D_1 = D_0 - Q_1 = 1000 - 210{,}53 = 789{,}47$

Pri ostalih analogno nadaljujemo postopek, da dobimo celoten amortizacijski načrt (Tabela 1) (ibid.).

Tabela 1: Amortizacijski načrt za 50 % obrestno mero/kapitalizacijsko obdobje s faktorji, ki prikazujejo razmerja med razdolžninami

Obrok	Anuiteta	Obresti	Razdolžnina	Faktorji	Ostanek
					1000,00 €
1	–710,53 €	–500,00 €	–210,53 €	1	789,47 €
2	–710,53 €	–394,74 €	–315,79 €	1,5	473,68 €
3	–710,53 €	–236,84 €	–473,68 €	2,25	0 €
∑			*1000,00 €*	*4,75*	

Opombe: Faktorji = faktorji naraščanja glede na izhodiščno razdolžnino.

Tu še zlasti velja, da je koristno kontrolirati stolpne vsote v skladu z ugotovitvami, zapisanimi ob prejšnjem zgledu (ibid., str. 96).

2.4.2 Geometrično zaporedje razdolžnin

Posamezne zneske lahko računamo tudi neposredno iz vhodnih podatkov, če upoštevamo odnose med posameznimi elementi v amortizacijskem načrtu (ibid.). Brez težav namreč lahko dokažemo, da pri dekurzivnem obrestovanju in konstantni anuiteti razdolžnine tvorijo geometrično zaporedje s količnikom $[(1+p)]$ (ibid.). Najprej je

$$D_i = D_{i-1} - (a - o_i) = D_{i-1} - (a - D_{i-1}p) = D_{i-1}(1+p) - a$$

ali na kratko (ko pogledamo začetek in konec) (ibid.)

$D_i = D_{i-1}(1+p) - a.$

Postopek ponovimo za D_{i+1} in nato dobljeni rezultat

$D_{i+1} = D_i(1+p) - a$

odštejemo od prejšnjega, da dobimo naslednjo enačbo:

$D_i - D_{i+1} = (D_{i-1} - D_i)(1+p).$

Ker vemo, da je razdolžnina v tekočem letu ravno razlika med prejšnjim in letošnjim ostankom dolga, smo s tem že dokazali prvo trditev (ibid.):

$$Q_{i+1} = Q_i(1+p) \qquad (i = 1, 2, \dots, n-1) \tag{3}$$

S pomočjo tega obrazca lahko pri amortizacijskem načrtu naredimo dvoje, in sicer kontroliramo že izračunane razdolžnine ali pa neposredno izračunamo kasnejše vrstice tega načrta (Tabela 1).

3 NUMERIČNO RAČUNANJE

Ko se določenega matematičnega problema lotimo s pomočjo računalnika, iščemo rešitev v numerični obliki in ne več v analitični (Plestenjak, 2015, str. 16). To npr. pomeni, da numerična metoda kot rešitev enačbe $x^2 = 3$ ne vrne $\sqrt{3}$, temveč 1.73205 …, pri čemer je število števk odvisno od natančnosti, v kateri računamo (ibid.).

3.1 Premična pika

Realna števila v računalniku, kot jih uporabljamo pri numeričnem računanju, so predstavljena v obliki premične pike kot

$$x = \pm m \cdot b^e, \tag{4}$$

kjer je $\mathrm{m} = 0.\mathrm{c}_1\mathrm{c}_2 \ldots \mathrm{c}_\mathrm{t}$ mantisa in:

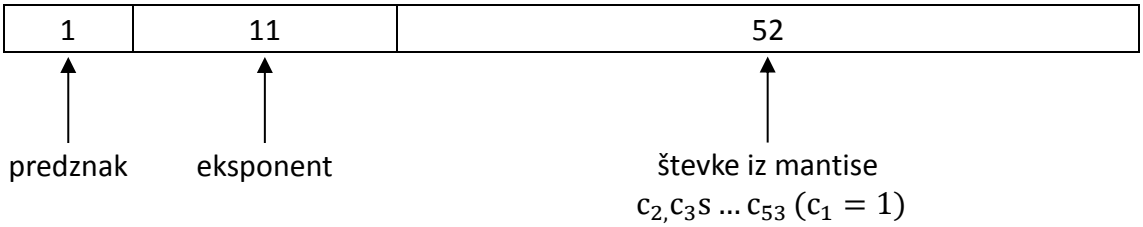
- b: $baza$:, po navadi je binarna (b = 2), lahko tudi desetiška (b = 10),
- t: $dolžina\ mantise$,
- e: $eksponent$, celo število v mejah $L \leq e \leq U$, kjer sta L in U spodnja in zgornja meja,
- c_i: $števka$, celo število v mejah 0 do b – 1 za i = 1,…, t (Plestenjak, 2015, str. 16).

Števila so normalizirana, kar pomeni $c_1 \neq 0$ (ibid., str. 17). Tako po eni strani dosežemo enoličnost zapisov, saj sta sicer $0.1 \cdot b^e$ in $0.01 \cdot b^{e+1}$ različna zapisa istega števila (ibid.) po drugi strani pa ohranjamo natančnost, saj poskrbimo, da ves čas uporabljamo celotno dolžino mantise (ibid.).

3.1.1 Množica predstavljivih števil

Množica vseh predstavljivih je določena s parametri $b, t, L\ in\ U$, kar označimo s $P(b, t, L, U)$ (ibid.). Leta 1985 je IEEE predstavil standard za računanje v aritmetiki s premično piko (standard IEEE 754 – v nadaljevanju standard IEEE), ki je bil vzpostavljen z namenom omogočiti izdelavo stabilnih, učinkovitih in prenosljivih numeričnih programov (Higham, 1996, str. 45). Najbolj znana množica predstavljivih števil v IEEE je $dvojna\ natančnost$: P (2, 53, –1021, 1024), število je shranjeno v 64 bitih.

Slika 4: Razporeditev bitov po parametrih (v dvojni natančnosti)



Prirejeno po: (Plestenjak, 2015, str. 17).

Iz razpona eksponentov vidimo, da ne porabimo vseh [2048] možnosti, ki jih imamo na voljo z zapisom eksponenta z enajstimi biti, temveč le [2046] (Plestenjak, 2015, str. 17).

Preostali vrednosti sta rezervirani za števila 0, ∞, $-\infty$ in nedoločeno vrednost NaN (Not-a-Number), ki jih ni mogoče prikazati po enačbi 1 (ibid.). Tako pri računanju v aritmetiki, ki podpira standard IEEE, dobimo npr. 1/0 = ∞, 3/∞ = 0 in 0/0 = NaN (ibid.). Medtem ko je rezultat vseh operacij v katerih nastopa NaN spet NaN, lahko iz vrednosti $\pm\infty$ z nadaljnjim računanjem spet pridemo do končnih vrednosti (ibid.). V sistemih, ki podpirajo standard IEEE, se tako programi[3] ne zaustavijo, če pride do deljenja z 0 (ibid.).

Standard IEEE dopušča tudi denormalizirana števila, kjer je $c_1 = 0$, pri čemer je eksponent fiksen (v primeru dvojne natančnosti je enak [–1021]) (ibid.). Denormalizirana števila imajo manj števk natančnosti kot normalizirana števila (Higham, 1996, str. 41). Zapolnjujejo praznino med 0 in najmanjšim pozitivnim- [oziroma največjim negativnim] predstavljivim številom in so enakomerno razporejena po realni osi (Plestenjak, 2015, str. 16).

Tabela 2: Pozitivna predstavljiva števila iz množice *P* (2, 3, –1, 1)

normalizirana števila

$0.100_2 \cdot 2^{-1} = 0.2500$	$0.100_2 \cdot 2^{0} = 0.500$	$0.100_2 \cdot 2^{1} = 1.00$
$0.101_2 \cdot 2^{-1} = 0.3125$	$0.101_2 \cdot 2^{0} = 0.625$	$0.101_2 \cdot 2^{1} = 1.25$
$0.110_2 \cdot 2^{-1} = 0.3750$	$0.110_2 \cdot 2^{0} = 0.750$	$0.110_2 \cdot 2^{1} = 1.50$
$0.111_2 \cdot 2^{-1} = 0.4375$	$0.111_2 \cdot 2^{0} = 0.875$	$0.111_2 \cdot 2^{1} = 1.75$

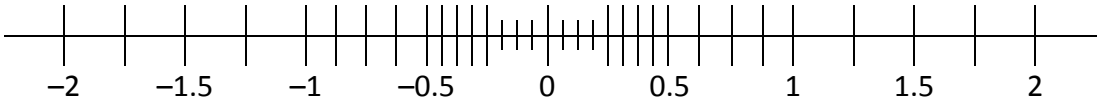
denormalizirana števila

$0.001_2 \cdot 2^{-1} = 0.0625$
$0.010_2 \cdot 2^{-1} = 0.1250$
$0.011_2 \cdot 2^{-1} = 0.1875$

Opombe: Med predstavljiva števila spadajo še zgornja števila z dodanim negativnim predznakom in število 0.

Vir: (Plestenjak, 2015, str. 18).

Slika 5: Razporeditev predstavljivih števil iz množice *P*(2, 3, –1, 1) po realni osi



Opombe: Daljše zareze – normalizirana števila, krajše zareze – denormalizirana števila.

Vir: (Plestenjak, 2015, str. 19).

3.1.2 Rezanje in zaokrožanje

Števila, ki niso predstavljiva (označimo jih s $fl\,(x)$), pri računanju nadomestimo s predstavljivimi približki (Plestenjak, 2015, str. 19). Pri tem lahko izberemo najbližji predstavljivi števili bodisi z leve ali desne (ibid.). V praksi se uporabljata dva načina, katero izmed števil izbrati (Bohte, 1995, str. 64). Pri *rezanju* odrežemo vse števke za katere nimamo prostora [kar pomeni, da se odločimo za tisto predstavljivo število, ki je bližje 0] (Plestenjak, 2015, str. 19). Pri *zaokrožanju* pa izberemo tisto predstavljivo

[3] Glej Priloga 7: Parametri računske aritmetike nekaterih programskih paketov.

število, ki je bližje x (ibid.). V primeru, ko je x ravno na sredini med levim in desnim predstavljivim številom, med njima izberemo tistega, ki ima sodo zadnjo števko (ibid.). Standard IEEE predpisuje zaokrožanje (ibid.).

3.1.3 Osnovna zaokrožitvena napaka

Ko se odločimo za rezanje ali zaokrožanje povzročimo *zaokrožitveno napako*, ki je odvisna od velikosti števila, nad katerim izvedemo postopek (Orel, 1997, str. 6). Njena natančna zgornja meja je odvisna od računalnika (ibid., str. 7). Imenujemo jo osnovna zaokrožitvena napaka (ibid.).

$$fl(x) = x(1+\delta)\ za\ |\delta| \leq u,$$

kjer je

$$u = \frac{1}{2} b^{1-t}.$$

Osnovna zaokrožitvena napaka (u) po IEEE standardu v dvojni natančnosti je: $u = 2^{-53} \approx 1 \cdot 10^{-16}$ (Plestenjak, 2015, str. 20).

V numerični matematiki zapis fl (izraz) pomeni element iz množice predstavljivih števil, ki ga dobimo kot rezultat numeričnega izračunanega izraza, pri čemer pri računanju uporabljamo samo osnovne računske operacije (ibid.). Če opravimo osnovno računsko operacijo na dveh predstavljivih številih, potem po standardu IEEE dobimo kot rezultat predstavljivo število, ki bi ga dobili, če bi operacijo najprej izračunali točno in nato zaokrožili rezultat (ibid.). Izjema je, ko pride do prekoračitve (overflow) ali podkoračitve (underflow) (ibid.). V tem primeru po standardu IEEE dobimo $\pm\infty$ v primeru prekoračitve in 0 v primeru podkoračitve (ibid.).

V primeru, da ne pride do prekoračitve ali podkoračitve, po standardu IEEE, torej za poljubni predstavljivi števili x in y velja (ibid.):

$fl(x \oplus y) = (x \oplus y)(1+\delta)$, kjer je $|\delta| \leq u$, za $\oplus = +,\ -,\ *,\ /$.

Če vzamemo primer odštevanja, ki ga navaja Petrišič (2006, str. 16): 764.2–764.1, dobimo v dvojni natančnosti – s pomočjo programskih paketov za računanje (Excel, VBA, Octave in R; slednja dva računata po IEEE standardu) – numerični rezultat 0.10000 00000 00023. Vendar bi bil dovoljen numeričen rezultat po IEEE standardu kvečjemu 0.10000 00000 00000111022302462516 (v kolikor obravnavamo le odmik navzgor). Vse pomembne števke v numeričnem rezultatu bi torej naj bile enake točnemu rezultatu. Zato menimo, da model računanja v premični piki za odštevanje ne drži v vsakem primeru. Če pa bi takemu odštevanju še sledili izračuni, bi lahko pričakovali, da bi le-ti bili le še na toliko mest natančni, kot je bilo odštevanje natančno (Petrišič, 2006, str. 17).

3.2 Napake pri numeričnem računanju

Ko z numerično metodo rešimo določen problem, dobimo približek za točen rezultat (Plestenjak, 2015, str. 21). Razlika med točnim rezultatom in dobljenim numeričnim približkom je celotna napaka približka (ibid.). Ta napaka je posledica nenatančnosti začetnih podatkov, napake numerične metode in zaokrožitvenih napak med samim računanjem (ibid.).

3.2.1 Relativna napaka

Pri numeričnem računanju vedno izračunamo numerični približek za točno rešitev problema (ibid.). Razlika med približkom in točno vrednostjo je napaka približka (ibid.). Ločimo absolutno in relativno napako (ibid.). Naj bo $\hat{x}$ pribiližek za točno vrednost x. Potem je *absolutna napaka*

$$E_{abs}(\hat{x}) = |x - \hat{x}| \tag{5}$$

in *relativna napaka*

$$E_{rel}(\hat{x}) = \frac{|x - \hat{x}|}{|x|} \tag{6}$$

(če je točna vrednost x enaka 0, je relativna napaka približka $\hat{x} \neq 0$ neskončna) (ibid.).

Pri premični piki gre za ocenjevanje relativnih napak (Bohte, 1995, str. 79). To je namreč neodvisno od skaliranja: skaliranje $x \to \alpha x$ in $\hat{x} \to \alpha\hat{x}$ pusti relativno napako nespremenjeno (Higham, 1996, str. 4).

3.3 Občutljivost problema in stabilnost metode

Z analizo in ocenjevanjem neodstranljive napake se ukvarja *teorija motenj* (*perturbacij*) (Plestenjak, 2015, str. 23). Pri teoriji motenj nas zanima, za koliko se spremeni točen rezultat, če malo zmotimo – perturbiramo – začetne podatke (ibid.). Problem je *občutljiv* (slabo pogojen), če lahko pride do velikih sprememb in *neobčutljiv* (dobro pogojen), če so spremembe majhne (ibid.).

Medtem ko je občutljivost problema povezana s samim problemom in je neodvisna od numerične metode, s katero rešujemo problem, se stabilnost navezuje na numerično metodo (Plestenjak, 2015, str. 26). Ko numerična metoda iz začetnih podatkov izračuna približek za točen rezultat, nas zanima, kako natančen je približek (ibid.).

3.3.1 Direktna stabilnost

Razlika med izračunano in točno vrednostjo je *direktna napaka* (ibid.). Tako je npr. ob predpostavki, da bi pri računanju $\sqrt{3}$ dobili rezultat 1.7 in ne $1.73205\ldots$ (absolutna) *direktna napaka* $0.03205\ldots$ Če je za vse začetne podatke (absolutna oz. relativna) direktna napaka majhna, je proces (absolutno oz. relativno) direktno stabilen, v nasprotnem primeru je direktno nestabilen (ibid.).

3.3.2 Obratna stabilnost

Vprašamo se, za koliko moramo najmanj zmotiti začetne podatke, da je izračunani približek točna rešitev za zmotene podatke (ibid.). Razlika med začetnimi in zmotenimi podatki je *obratna napaka* (ibid.). Ob predpostavki, da smo pri računanju $\sqrt{3}$ dobili rezultat 1.7, naredimo najprej preizkus tega izračuna, torej $1.7^2 = 2.89$, kar bi odšteli od prvotnih 3, s čimer bi (absolutna) *obratna napaka* znašala 0.11. Če numerična metoda za vse začetne podatke vrne rešitev z majhno (absolutno oz. relativno napako) napako je metoda (absolutno oz. relativno) *obratno* stabilna (ibid.).

3.3.3 Stabilnost metode

Analiza obratnih napak omogoča, da zaokrožitvene napake, ki se pojavijo med računanjem, obravnavamo kot motnje začetnih podatkov (ibid., str. 27). Velja pravilo:

$$|\text{direktna napaka}| \lesssim \text{občutljivost} \times |\text{obratna napaka}|. \qquad (7)$$

Ker smo predhodno že izračunali tako direktno (≈ 0.03) kot tudi obratno napako (0.11) za $\sqrt{3}$, lahko iz enačbe izračunamo občutljivost problema. Občutljivost je $\gtrsim 0.29$. Če ima občutljivost vrednost 1, potem je relativna napaka rezultata približno enaka relativni napaki podatka (Petrišič, 2006, str. 33). Če je večje kot 1 pomeni, da se relativna napaka povečuje, če pa je manjše od 1, relativna napaka pada (ibid.). Problemi, ki imajo veliko število pogojenosti so *občutljivi* (slabo pogojeni) (ibid.). Glede na rezultat, ki smo ga dobili pri računanju $\sqrt{3}$, lahko torej rečemo, da (kvadratno) korenjenje predstavlja *neobčutljiv* problem.

Za metodo pravimo, da je *numerično stabilna*, če vedno vrne bližnji rezultat za malo zmotene začetne podatke (Plestenjak, 2015, str. 27). Če je metoda obratno oz. direktno stabilna je tudi numerično stabilna (ibid.). Vse osnovne računske operacije so numerično stabilne, kar smo prikazali na primeru kvadratnega korenjenja[4].

Če je problem občutljiv, tudi stabilna numerična metoda ne pomaga veliko, saj iz enačbe vidimo, da se tedaj numerični in točni rezultat lahko dosti razlikujeta (ibid.). V splošnem je nenatančnost rezultata posledica bodisi (ibid.):

- uporabe stabilne metode na občutljivem problemu ali
- uporabi nestabilne metode.

Natančnost izračuna je zagotovljena le takrat, ko neobčutljiv problem rešimo z numerično stabilno metodo (ibid.).

---

[4] Plestenjak (2015, str. 13) piše, da so osnovne računske operacije [za področje numeričnih metod] seštevanje, odštevanje, množenje, deljenje in *kvadratni koren*, pri čemer pa navaja, da je opredelitev, kaj so osnovne računske operacije, odvisna od okolja, v katerem se računa. Higham (1996, str. 44) piše, da je normalno predpostavljati, da model računanja po IEEE standardu, ki velja za osnovne računske operacije [seštevanje, odštevanje, množenje, deljenje] velja tudi za *kvadratno korenjenje*.

3.4 Analiza zaokrožitvenih napak

Pred samim računanjem lahko izpeljemo dve vrsti analize (Bohte, 1995, str. 78).

- Pri *direktni analizi* ocenjujemo napake delnih rezultatov zapored. Na koncu dobimo oceno za zaokrožitveno napako rezultata. Če dobimo ugodno oceno za zaokrožitveno napako rezultata (npr. nu, kjer je n število računskih operacij, u pa osnovna zaokrožitvena napaka), pravimo, da je proces *direktno stabilen*.
- Pri *obratni analizi* pa pokažemo, da je izračunani rezultat ravno takšen, kot bi ga dobili z eksaktno aritmetiko, a pri nekoliko drugačnih podatkih in ocenimo spremembe podatkov. Če so te spremembe podatkov primerno omejene (npr. z nu , pravimo, da je proces *obratno stabilen*).

Lahko pa uporabimo tudi *sprotno* analizo zaokrožitvenih napak, ki omogoča ocenjevanje napak hkrati z računanjem samim (ibid., str. 94). Pri tem lahko uporabljamo izračunane približke, kar omogoča bolj realistično ocenjevanje (ibid.).

Za oceno točnosti izračunov anuitetnih odplačil bomo uporabili *direktno analizo* zaokrožitvenih napak, saj nas zanima relativna napaka numeričnega rezultata glede na točen rezultat. Ker imamo nizek u $(\approx 2^{-53})$, lahko pri *analizi* zaokrožitvenih napak uporabimo poenostavljen postopek (ibid., str. 84), kar pomeni, da je:

- ocena za relativno napako *množenja več števil* približno enaka nu, torej številu operacij, pomnoženih z osnovno zaokrožitveno napako;
- ocena za relativno napako *seštevanja več števil* prav tako približno enaka nu, torej številu operacij, pomnoženih z osnovno zaokrožitveno napako (velja le, če so vsa števila pozitivna ali vsa števila negativna);
- ocena za relativno napako *odštevanja dveh števil (oz. seštevanja dveh števil različnih predznakov)* $\frac{|x|+|-y|}{|x+(-y)|}$.

3.5 Izogibanje numerični nestabilnosti

Bohte (1995, str. 128) na koncu poglavja o premični piki navaja, da je mogoče iz predhodnih ocen in primerov dobiti izkušnjo, da je pri računanju s premično piko edini vir močnega povečanja relativne napake odštevanje [dveh] približno enakih števil. Temu se je potrebno po možnosti izogniti (ibid.). Če vedno računamo tako, da vemo iz izkušenj, da ne bo napaka po nepotrebnem velika, se odpovemo ocenjevanju napak in si to sposobnost prihranimo le za takrat, ko lahko jamčimo za velikost napake (ibid.).

V izogib nevarnosti nestabilnega računanja si je potrebno izraz [enačbo ali del enačbe], ki v izvirni obliki ni primeren za računanje, ogledati z vidika numerične stabilnosti in ga predelati v za to primerno obliko (ibid., str. 94).

4 FORMULE ZA IZRAČUNAVANJE ANUITET

S periodičnimi denarnimi tokovi se v stabilnih gospodarstvih srečujejo med drugim kreditojemalci, ki v enakih časovnih razmikih po delih odplačujejo svoj dolg. Osnova za vse izračune je ekvivalenca glavnic, o kateri smo predhodno govorili (Čibej & Ferbar Tratar, 2012, str. 70).

4.1 Formule, ki temeljijo na načelu ekvivalence glavnic

Na spletnem mestu Wikipedia (2015) smo zasledili sledeči[5] formuli[6]:

$$a = \frac{D_0 \times (1+p)^n}{\left(\frac{(1+p)^n - 1}{p}\right)}, \quad p \neq 0 \tag{8}$$

$$a = \frac{D_0}{\left(-\frac{(1+p)^{-n} - 1}{p}\right)}, \quad p \neq 0. \tag{9}$$

Načelo ekvivalence glavnic pomeni, da je vsota vseh vplačil (preračunanih na izbrani termin) enaka vsoti vseh izplačil preračunanih na isti termin (ibid., str. 10). Formula (8) preračunava vse zneske na termin, ko dospeva zadnja anuiteta (v našem primeru konec 3. anuitetnega odplačila) (Slika 6a). V kolikor formulo izrazimo z obliko za postopno računanje, lahko vidimo, da se na levi strani računa število preteklih kapitalizacijskih obdobij od dneva *valute kredita*, do dneva preračuna, desna stran enačbe pa računa število kapitalizacijskih obdobij od dneva *posameznih anuitetnih odplačil* pa do dneva preračuna:

a1. $D_0(1+p)^n = a(1+p)^0 + a(1+p)^1 + a(1+p)^2$;

a2. $1000(1+50\,\%)^3 = a(1+50\,\%)^0 + a(1+50\,\%)^1 + a(1+50\,\%)^2$;

a3. $1000 \cdot 3{,}375 = 4{,}75a.$

Formula (9) preračunava vse zneske na dan *najetja kredita* (Slika 6b). Ker so to v primerjavi s formulo (8) 3 obrestovalna obdobja prej, lahko zapišemo postopek kot sledeč (Slika 6b):

b1. $D_0(1+p)^{n-3} = a(1+p)^{0-3} + a(1+p)^{1-3} + a(1+p)^{2-3}$;

---

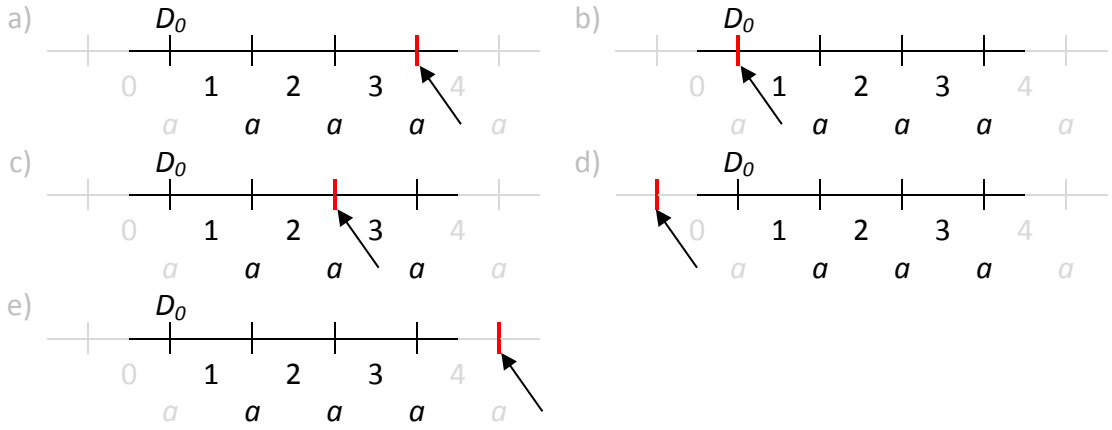
[5] V formuli (8) smo p prestavili iz števca v imenovalec, v formuli (9) smo naredili enako, poleg tega smo še zamenjali vrstni red zapisa v imenovalcu, zato smo morali negirati zapis v imenovalcu. V obeh primerih smo to naredili z namenom, da je razvidna vsota geometričnega zaporedja v imenovalcu, s čimer vse formule v tem poglavju temeljijo na enakem zapisu.

[6] Spremenljivke za vse formule v tem poglavju $a =$ anuiteta, $D_0 =$ glavnica, $n =$ število obrokov, $p =$ obrestna mera na kapitalizacijsko obdobje, $r =$ obrestovalni faktor, $Z =$ številka razdolžnine, vzeta kot osnova za izračun (poljubno celo število)

b2. $1000(1+50\,\%)^{0} = a(1+50\,\%)^{-3} + a(1+50\,\%)^{-2} + a(1+50\,\%)^{-1}$;

b3. $1000 \cdot 1 = 1{,}40 \ldots a.$

Slika 6: Dinamika prilivov in odlivov pri anuitetnem kreditu za 3 obrestovalna obdobja, preračunano na različne termine



Opombe: D_0 = dolg; a = anuiteta; 1, 2, 3 = zaporedna številka obrestovalnega obdobja.

Prirejeno po: (Čibej & Ferbar Tratar, 2012, str. 95).

Formuli, kot del (ene) splošne formule

Glede na to, da si lahko izberemo trenutek, na katerega bomo preračunali vse prilive oz. odlive, lahko uporabimo tudi eno samo formulo, ki ji dodamo še spremenljivko (poimenujmo jo Z), ki opredeljuje, na kateri termin se naj preračunajo zneski. Formula je sledeča:

$$a = \frac{D_0 \times (1+p)^{(n+1)-Z}}{\left(\frac{(1+p)^{(n+1)-Z}-1}{p} - \frac{(1+p)^{-(Z-1)}-1}{p}\right)}, \quad p \neq 0\,, \quad Z = \mathbb{Z}. \tag{10}$$

Tako je pri $Z = 1$ termin za preračun datum *dospetja zadnje anuitete* (v tem primeru je formula (10) enaka formuli (8)). Pri $Z = n + 1$ pa se zneski preračunajo na termin *valute kredita* (v tem primeru je formula (10) enaka formuli (9)). Na Sliki 6c je prikazan primer, ko je $1 < Z < n + 1$, na Sliki 6d je $Z > n + 1$, ter na Sliki 6e je $Z < 1$, pri čemer predstavlja Z številko razdolžnine, ki ima faktor oz. vrednost 1. Vsota geometričnega zaporedja je tako seštevek faktorjev vseh razdolžnin, ki sestavljajo večkratnik vrednosti izhodiščnega faktorja Z.

4.2 Formule, ki temeljijo na razmerju anuitete in zneska prvih obresti

Na spletnem mestu Wikipedia (2015) smo zasledili še sledečo formulo

$$a = D_0 \times p + \frac{D_0}{\left(\frac{(1+p)^n - 1}{p}\right)}, \quad p \neq 0. \tag{11}$$

Izvirnik formule je sicer imel izpostavljene skupne faktorje. Formula (11) je očitno drugačna od formul (8) in (9). Če si pogledamo Tabelo 3, vidimo, da predstavlja izračunana anuiteta v formulah (7) in (8) nadaljevanje geometričnega zaporedja razdolžnin 1, 2 in 3, formula (11) pa je rezultat seštevka razdolžnine 1 in prvih obresti. Absolutni znesek prvih obresti je ob dani glavnici in obrestni meri ne glede na število kapitalizacijskih obdobij namreč vedno enak, zato je mogoče uporabiti tudi drugi naveden način.

Tabela 3: Amortizacijski načrt za 50 % obrestno mero/kapitalizacijsko obdobje

Obrok	Anuiteta	Obresti	Razdolžnina	Faktorji (Z=1)	Dolg
					1000,00 €
1	−710,53 €	−500,00 €	−210,53 €	1	789,47 €
2	−710,53 €	−394,74 €	−315,79 €	1,5	473,68 €
3	−710,53 €	−236,84 €	−473,68 €	2,25	0 €
Anuiteta			−710,53 €		

Opombe: **Faktorji =** Faktorji naraščanja glede na izhodiščno razdolžnino; postopek računanja formul (6) in (7) je označen z zeleno barvo, postopek računanja formule (9) pa je označen z modro barvo.

Formula (11) vsebuje enak seštevek faktorjev, ki predstavljajo razmerja med razdolžninami – glede na izbrano razdolžnino – kot jih vsebuje formula (8) (del, ki je pri obeh formulah obarvan z rdečo). Tukaj še dodajmo formulo (12), ki predstavlja odgovor na formulo (9) (vsebujeta enak seštevek faktorjev; del, ki je obarvan z modro).

$$a = D_0 \times p + \frac{D_0 \times (1+p)^{-n}}{\left(-\frac{(1+p)^{-n} - 1}{p}\right)}, \quad p \neq 0. \tag{12}$$

Formuli, kot del (ene) splošne formule

Tudi pri formulah (11) in (12) lahko uporabimo eno samo formulo, ki ji dodamo še spremenljiv faktor (Z) – formula (13), pri čemer tudi tukaj Z predstavlja številko razdolžnine, ki ima faktor oz. vrednost 1. Vsota geometričnega zaporedja je tako seštevek faktorjev vseh razdolžnin, ki sestavljajo večkratnik vrednosti izhodiščnega faktorja Z. Tako ima formula (11) $Z = 1$, formula (12) pa $Z = n + 1$.

$$a = D_0 \times p + \frac{D_0 \times (1+p)^{-(Z-1)}}{\left(\frac{(1+p)^{(n+1)-Z} - 1}{p} - \frac{(1+p)^{-(Z-1)} - 1}{p}\right)}, \quad p \neq 0\,, \quad Z = \mathbb{Z}. \tag{13}$$

5 ANALIZA FORMUL IN PREFORMULACIJA PROBLEMA

Bohte (1995, str. 128) na koncu poglavja o premični piki navaja, da je mogoče iz predhodnih ocen in primerov dobiti izkušnjo, da je pri računanju s premično piko edini vir močnega povečanja relativne napake odštevanje [dveh] približno enakih števil. Temu se je potrebno po možnosti izogniti (ibid.). V izogib nevarnosti nestabilnega računanja si je potrebno izraz [enačbo ali del enačbe], ki v izvirni obliki ni primeren za računanje, ogledati z vidika numerične stabilnosti in ga predelati v za to primerno obliko (ibid.).

5.1 Analiza formul

V poglavju 4 smo predstavili formule za računanje enakih kreditnih odplačil. V nadaljevanju bomo pogledali ali se te formule, ki so sicer z vidika analitičnega računanja enake, pri numeričnem računanju pod določenimi pogoji obnašajo različno. Na podlagi teh ugotovitev bomo izbrali formulo, ki bo podlaga za računanje s pomočjo dopolnilnih algoritmov.

5.1.1 Prekoračitev obsega

Tabela 4: Faktorji, ki sestavljajo geometrično zaporedje razdolžnin, pri izbrani izhodiščni razdolžnini 1, za 50 % obrestno mero/kapitalizacijsko obdobje

Razdolžnina	Faktorji (Z < 0)	Faktorji (Z = 0)	Faktorji (Z = 1)	Faktorji (Z = 2)	Faktorji (Z = 3)	Faktorji (Z = 4)	Faktorji (Z > 4)
…	…	…	…	…	…	…	…
1	…	1,5	1	≈0,66	≈0,44	≈0,29	…
2	…	2,25	1,5	1	≈0,66	≈0,44	…
3	…	3,375	2,25	1,5	1	≈0,66	…
∑	…	*8,125*	*4,75*	≈3,16	≈2,11	≈1,40	…

Opombe: Z = številka razdolžnine, vzeta kot osnova za izračun (poljubno celo število).

Iz Tabele 4 vidimo, da nižja, kot je vrednost Z, večje je tveganje za prekoračitev številskega obsega v vmesnem rezultatu (ki jo lahko povzroči kombinacija visoke obrestne mere in/ali visokega števila kapitalizacijskih obdobij). Predpostavljamo, da imajo vse formule, pri katerih je $Z \leq 1$ podobne lastnosti (pozitivno potenco), zato bomo v okviru nevarnosti prekoračitve številskega obsega izločili *skoraj* vse te formule. Pri tem je seveda potrebno eno izmed teh formul v skupini obdržati, saj jo je potrebno primerjati še s formulami, ki imajo $Z > 1$. Le-te spadajo namreč v drugo skupino (oz. skupini) in imajo druge lastnosti, ki so lahko v tem primeru sicer bolj ugodne, za drug primer (v okviru medskupinske primerjave) pa manj ugodne. Ker se v množici formul pri katerih je $Z \leq 1$ zgodi prekoračitev številskega obsega najkasneje pri $Z = 1$, bomo zato slednjo obdržali. Iz možne nadaljnje obravnave pa izločimo vse formule, ki imajo $Z < 1$. Ugotovitve veljajo za obe splošni formuli enako. Končna formula, ki jo bomo uporabili mora torej imeti $Z \geq 1$.

5.1.2 Podkoračitev obsega

Pri podkoračitvi obsega je število tako nizko, da se spremeni v vrednost 0, pri čemer velja, da višja kot je vrednost Z, večje je tveganje za podkoračitev številskega obsega. V tem primeru predpostavljamo, da imajo vse formule, pri katerih je $Z \geq n + 1$ podobne lastnosti (negativno potenco), zato bomo v okviru nevarnosti podkoračitve številskega obsega izločili *skoraj* vse te formule. Pri tem je seveda potrebno eno izmed teh formul v skupini obdržati, saj jo je potrebno primerjati formulami, ki imajo $Z < n + 1$. V tej skupini nastopi podkoračitev najkasneje pri $Z = n + 1$, zato je ne izločimo iz postopka izbire. Tudi tukaj veljajo ugotovitve za obe splošni formuli enako. Končna formula, ki jo bomo uporabili, bo tako imela $1 \leq Z \leq n + 1$.

5.1.3 Primernost formule, kot podlaga za dopolnilne algoritme

V enem izmed računskih postopkov, ki bodo sledili (Algoritem 5), je potrebno izračunati izhodiščno obrestno mero za uporabo v omenjenem postopku. Pri formulah, ki imajo $Z \leq 1$, torej (izključno) pozitivno potenco, to sicer ni potrebno. Kot izhodišče se uporabi namreč le p. Vendar pa lahko nastopi pri formulah, ki imajo $Z < 1$ v računskem postopku

$$\frac{(1+p)^{(n+1)-Z} - 1}{p} \quad - \quad \frac{(1+p)^{-(Z-1)} - 1}{p}$$

odštevanje dveh približno enakih števil. Pri formulah, ki pa imajo $Z > 1$, torej (deloma ali v celoti) negativno potenco, pa se izhodiščna obrestna mera (v delu ali izključno) računa z $(1+p)^{-n} - 1$. V obeh primerih vidimo, da bomo pri nizki obrestni meri ponovno naleteli na težave, ki se nanašajo na odštevanje dveh približno enakih števil. Spet velja ugotovitev za obe splošni formuli enako. Smo pa s tem postopkom določili optimalno vrednost Z, najbolj smiselna je torej uporaba formule, ki ima $Z = 1$.

5.1.4 Računanje z neskončno vrednostjo

Imamo torej določeno vrednost Z, ne vemo pa še, katero splošno formulo naj uporabimo. Pri $Z = 1$ sta splošni formuli enaki formuli (8) oz. (11). V kolikor izraz $(1+p)^n$ v formuli zamenjamo z ∞ (neskončno) dobimo sledeči različni rešitvi. V števcu formule (8) bi tako delili neskončno z neskončno in dobili rezultat "ni število". V formuli (11) pa je rezultat enak absolutnemu znesku prvih obresti. Tukaj gre za računski primer, ko imamo kombinacijo visoke obrestne mere in/ali visokega števila kapitalizacijskih obdobij. To pomeni, da če bi rezultat iz formule (11), vnesli v amortizacijski načrt, se kredit sploh ne bi odplačeval, saj so obresti tako visoke, da je razdolžnina 0 in tako se dolg ne bi zmanjševal. Pri tem bi lahko po eni strani sicer rekli, da je pravilen rezultat formule (8), saj ta pravi, da je rezultat "nesmiseln" (ni število). Vendar pa, če premislimo, ima kombinacija visoke obrestne mere in/ali visokega števila kapitalizacijskih obdobij neko realno računsko rešitev, ki se nahaja blizu rezultata, izračunanega s formulo (11). Pri obstoječi natančnosti ne moremo ponazoriti, da je višina anuitete realno višja od zneska (prvih) obresti. Zato bomo kot podlago za računanje z algoritmi – ki bo sledilo – uporabili formulo (11).

Slika 7: Primerjava izračunov formul pri prekoračitvi številskega obsega v vmesnem rezultatu pri *n* = ∞

$$\lim_{f\to\infty} \frac{D_0 \times f}{\left(\frac{f-1}{p}\right)} = NaN, \qquad f = (1+p)^n, \qquad p \neq 0 \tag{8}$$

$$\lim_{f\to\infty} D_0 \times p + \frac{D_0}{\left(\frac{f-1}{p}\right)} = D_0 \times p, \qquad f = (1+p)^n, \qquad p \neq 0 \tag{11}$$

5.2 Odštevanje znotraj formule

Pri analizi formul z vidika numerične nestabilnosti bomo v tem poglavju obravnavali formulo (11) oz. del, s katerim računamo geometrično zaporedje. V prejšnjem razdelku smo obrazložili, zakaj smo izbrali ravno to formulo.

5.2.1 Analiza zaokrožitvenih napak pri odštevanju v formuli

V poglavju o numeričnem računanju smo zapisali, da so množenje, deljenje ter seštevanje numerično stabilne metode. V formuli imamo eno množenje, eno seštevanje, eno deljenje ter še eno deljenje (v imenovalcu imenovalca), ki bi sicer lahko bilo tudi množenje, če bi ga pustili v števcu.

Za preizkus stabilnosti računanja *odštevanja* si bomo predočili sledeč primer. Vzemimo, da smo najeli kredit z odplačilno dobo n = 1 in obrestno mero p = 0,15 %. Tako dobimo $(1+p)^n = 1.0015$. Zmanjševanec in odštevanec v formuli, t. j. 1, sta si očitno blizu, s čimer lahko pričakujemo veliko relativno napako v izračunu.

Sedaj ta izračunani zmanjševanec ter konstanto 1 vstavimo v obrazec za računanje koeficienta pri osnovni zaokrožitveni napaki u v oceni za relativno napako seštevanj $\mathrm{n}+1$ števil (Bohte, 1995, str. 84). Rezultat za $\kappa_n = (|1.0015| + |-1|)/(|1.0015 + (-1)\,|) \approx 1334$, ki ga še pomnožimo z osnovno zaokrožitveno napako u. Dobimo $\kappa_n \approx 1 \cdot 10^{-13}$. To pomeni, da smo mesto v rešitvi, od katerega dalje lahko nastopi napaka s 16. premaknili na 13. števko, torej lahko na zadnjih treh mestih rezultata dobimo slučajne cifre. Tako na primer pri D_0 = 1000 denarnih enot in že danima n in p dobimo numerično rešitev –1001,49999999996. S točno rešitvijo 1001,5 se ne ujema na zadnji števki, kar pomeni, da je napaka manjša od ocenjene. S tem se sicer tudi potrjuje ugotovitev, da daje analiza zaokrožitvenih napak običajno pesimistične napovedi glede točnosti rezultatov (Bohte, 1995, str. 128)[7].

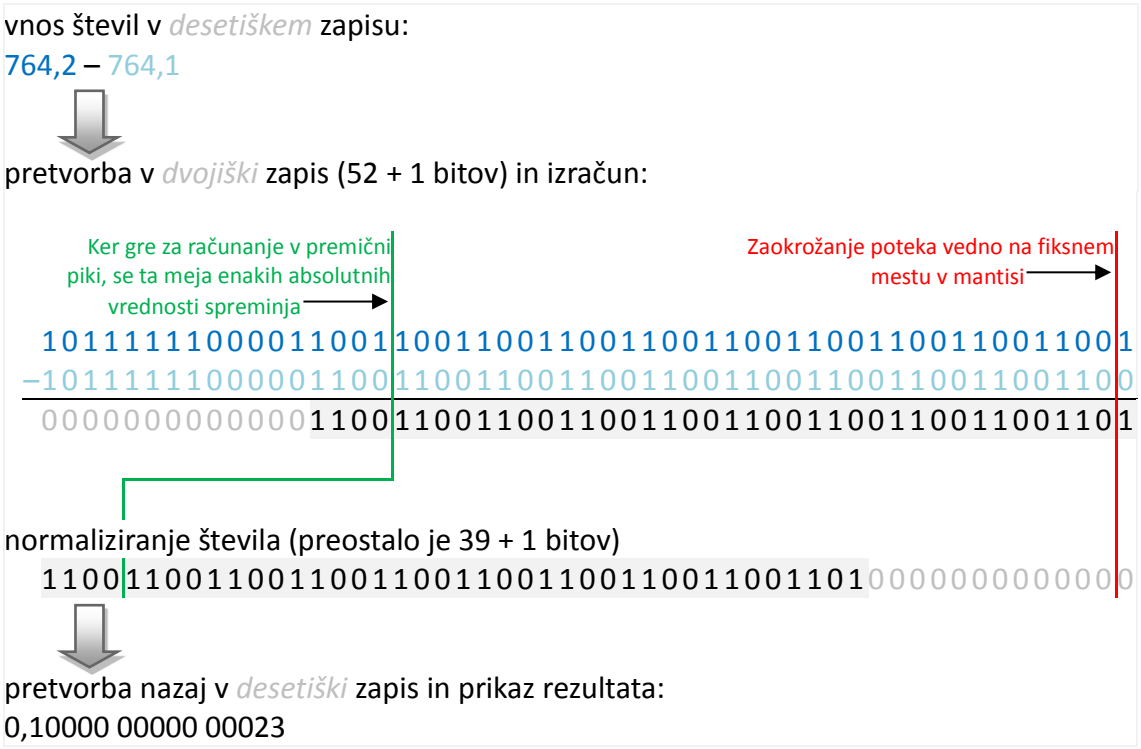
5.2.2 Pomen števk v rezultatu pri odštevanju dveh približno enako velikih števil

Iz predhodnega primera vidimo, da se lahko števke pri numeričnem rezultatu odštevanja dveh približno enako velikih števil več ne skladajo s točno rešitvijo. Tako npr.

[7] Higham (1996, str. 98) obravnava *statistično vrednotenje* ocen zaokrožitvenih napak na točnost rezultatov.

Petrišič (2006, str. 16) piše o " 'slučajnih' " številkah v rezultatu. Plestenjak (2015, str. 34 in 36) pa navaja pojma: "napačne številke" in [številke, ki] "nimajo pravega pomena". Ker Higham (1996, str. 29) navaja, da napake pri računanju ter njihov vpliv na nadaljnje računske postopke niso slučajne, si lahko postavimo vprašanje, od kod dobimo te številke (za naš primer) in kako nam je to lahko v pomoč pri odpravljanju težav z natančnostjo v rezultatu.

Slika 8: Postopek odštevanja v premični piki z dvojno natančnostjo



Ob vnosu števil v računalnik (v desetiškem zapisu) se števila pretvorijo v dvojiško obliko, opravi se izračun in za prikaz rezultata se opravi pretvorba nazaj v desetiški sistem. Desetiška števila, ki se ne dajo točno shranit v dvojiško obliko, potrebujejo za zapis na 15 števk natančno približno 50 bitov natančnosti (število števk pomnoženo s faktorjem $1/\log_{10} 2 \approx 3.33$), kar je dovolj – da se po tem, ko se opravi zaokrožanje števila – doseže "navidezna" natančnost. Za konkreten primer odštevanja, ki ga navaja Petrišič (2016, str. 16) (Slika 8) smo rezultat, ki ga dobimo z računalnikom preverili tudi s "peš" izračunom (s pomočjo preglednice v Excelu), rezultata sta bila na vse številke enaka. To pomeni, da bi moral biti postopek odštevanja, ki ga izpelje računalnik, podoben prikazanemu.

Iz Slike 8 lahko razberemo, da se je pri računanju v dvojiškem zapisu 13 bitov (*na začetku*) izničilo. Ker se v premični piki rezultat po izračunu še normalizira – kar pomeni da se bitni zapis rezultata premakne naprej, da dobimo na prvem mestu bit 1 – nam sedaj (*na koncu*) manjka 13 bitov, torej nam manjka določena natančnost. Zaradi tega je lahko rezultat bodisi previsok (kot v našem primeru) bodisi prenizek. Kaj bi lahko iz tega sklepali?

Slika 9: Primer odštevanja dveh približno enakih števil – analitično in numerično

764,222222222222	764,222222222222
−764,111111111111	−764,111111111111
0,111111111111	0,111111111110972

Vrednosti iz prejšnjega primera smo nekoliko spremenili (uporabili smo 15-mestna zmanjševanec in odštevanec) in opravili tako analitičen kot tudi izračun z računalnikom. Pri tem lahko opazimo (Slika 9), da je skupno število števk pri numeričnem računanju – od 1. števke zmanjševanca (kot absolutno večjega števila v postopku odštevanja), pa do zadnje števke rezultata – 18. Očitno ima rezultat 3 števke preveč. Numerično računanje torej "ohranja" število števk v rezultatu. Tukaj nastopi premislek, da bi lahko numerično rešitev približali analitični, če bi na ustreznem mestu rezultat zaokrožili. S tem bi tudi te "napačne'', "slučajne" oz. števke, ki "nimajo pomena", izničili.

5.2.3 Popravek računske operacije odštevanja

Formula (14) (glej tudi: Priloga 6) temelji na tem, da se poskuša ugotoviti, na kateri števki (F = število pomembnih števk) je potrebno zaokrožiti rezultat. Postopek za odštevanje obravnava za veljavne vhodne spremenljivke vse pozitivne ali negativne vrednosti. Če se odštevata števili z različnima predznakoma, je to seštevanje, ki ni problematično in se zato tukaj ne izvedejo nikakršni popravki. Če pa odštevamo dve števili z enakima predznakoma, je pa to tudi zares odštevanje. Najprej se določi najnižja absolutna vrednost med številoma. "Najnižja" zato, ker imata lahko števili – čeprav sta približno enake velikosti – različno število števk v celem delu in posledično različno števk v decimalnem delu (npr. če je eno višje od vrednosti 1000, drugo pa nižje). To pomeni, da je od 1. števke zmanjševanca (če je ta absolutno večje število v postopku odštevanja), pa do zadnje števke odštevanca 16 mest. Če bi sedaj rekli, da je potrebno rezultat zaokrožiti na število decimalnih mest, ki jih ima višje število (zmanjševanec), bi izgubili eno mesto natančnosti, saj nismo upoštevali zadnje števke odštevanca. "Absolutno" vrednost pa potrebujemo zato, ker bomo računali z logaritmi. Najnižjo absolutno vrednost torej uporabimo kot logaritmand za računanje desetiškega logaritma. Na ta način ugotovimo, koliko števk je v celem delu absolutno nižje vrednosti, preostale števke pa predstavljajo decimalna mesta. Na to število števk se potem zaokroži rezultat odštevanja.

Rezultati, ki ga dobimo s tem algoritmom so sedaj sicer (domnevno) znotraj zahtev, ki jih postavlja IEEE standard za odštevanje. Vendar je takšen algoritem smiselno uporabiti le, kadar je odštevanje *edina* računska operacija ali *zadnja* znotraj niza računskih operacij.

$$x - y = \begin{cases} x - y, & xy \leq 0 \\ \dfrac{\lfloor (x-y) \cdot 10^{F - \lfloor \log_{10} \min(|x|,|y|) \rfloor + 1} \rceil}{10^{F - \lfloor \log_{10} \min(|x|,|y|) \rfloor + 1}}, & xy > 0, \quad F = 15 \end{cases} \tag{14}$$

Ko smo s tem algoritmom opravljali konkretne izračune anuitet (izraz $(1+\mathrm{p})^{\mathrm{n}} - 1$ v formuli smo zamenjali s spremenljivko, ki je vsebovala rezultat odštevanja algoritma), so bili izračuni sicer dobri pri nizki obrestni meri ($p \lesssim 10^{-5}\,\%$), pri relativno višjih obrestnih merah pa je lahko prišlo tudi do slabših rezultatov, kot jih ponuja formula

sama. Razlago za to lahko poiščemo v predhodnem rezultatu potenciranja. Pri izrazito nizkih obrestnih merah je namreč napaka, nastala pri potenciranju, v primerjavi z rezultatom (sedaj popravljenega algoritma) odštevanja dokaj majhna, in jo tudi sledeče deljenje z izrazito nizko obrestno mero toliko ne poveča. Pri nekoliko višjih obrestnih merah pa je napaka pri potenciranju višja in po sledečem odštevanju ter deljenju z nizko obrestno mero rezultat (podobno kot pri samih formulah) preraste v rezultat, ki je bolj "slučajen".

V formuli torej težava ni odštevanje samo, temveč kombinacija sledečih računskih operacij: $((1+p)^n - 1\ )/\ p$. Potenciranje je namreč vrnilo rezultat, ki se ga ni dalo v celoti prikazati. Odštevanje je to napako razkrilo. Kasnejše deljenje z nizko obrestno mero (ki je sledilo odštevanju) je to napako zelo povečalo[8]. Iz tega lahko sklepamo, da nima smisla popravljati metode odštevanja, temveč da moramo preurediti računski postopek.

5.3 Preformulacija problema

Čibej & Ferbar Tratar (2012, str. 69) pišeta, na kaj je potrebno biti pozoren, ko se opravljajo izračuni za periodične denarne tokove. V smislu pravilno izpeljanega računskega postopka je potrebno prilive in odlive prikazati na časovni premici in potem v skladu s skico in načelom ekvivalence glavnic … […] … opraviti izračun (ibid., str. 10). Torej govorita o tem, da je potrebno v postopku računanja najprej izpeljati enačbo, iz katere so razvidna razmerja med prilivi in odlivi glede na trenutek reduciranja. To pomeni postopno računanje. Pred izračunom bi sicer sledila pretvorba v eksplicitni zapis (ibid., str. 95), torej formulo, vendar smo v prejšnjem podpoglavju že ugotovili, da na tak način ne dobimo zadovoljivih rezultatov pri računanju anuitet. Zato bomo del v formuli, ki računa seštevek faktorjev razdolžnin, ki tvorijo geometrično zaporedje, računali postopno. Del formule, ki ga želimo nadomestiti v našem primeru je reduciran na termin, ko dospeva zadnja anuiteta. Ker bomo računali le s faktorji in ne z absolutnimi vrednostmi razdolžnin, bo tako prva razdolžnina imela vrednost 1. To pomeni, da seštevek vseh razdolžnin predstavlja tudi večkratnik vseh razdolžnin na prvo razdolžnino. Vsi sledeči postopki se uporabijo za računanje vsote geometričnega zaporedja S, ki se nato vstavi v formulo (11)

$$a = D_0 \times p + \frac{D_0}{(S)}.$$

5.3.1 Računanje vsote členov geometričnega zaporedja neposredno iz vhodnih podatkov

V kolikor reduciramo vse zneske na isti termin (na termin dospetja zadnje anuitete) lahko posamezne razdolžnine v geometričnem zaporedju računamo neposredno iz vhodnih podatkov. Če nekoliko poenostavimo zapis po Čibej & Ferbar Tratar (2012, str. 95), tako dobimo sledeč računski zapis:

[8] Kar se sklada s primerom, ki ga obravnavata na primer Burden & Faires (2011, str. 24).

$$S = r^0 + r^1 + r^2 + r^3 + \cdots + r^{n-1}, \quad r = (1+p). \tag{15}$$

Če želimo takšen postopek uporabiti za računanje z računalnikom, je smiselno narediti pretvorbo, npr. na način kot je prikazan na Algoritmu 1.

Algoritem 1: Računanje vsote členov geometričnega zaporedja neposredno iz vhodnih podatkov

1. $S_0 = 0$
2. $S_i = S_{i-1} + (1+p)^{i-1}, \quad i = 1, 2, \ldots, n$
3. $S = S_i$

Opombe: Priloga 1 vsebuje konkretne algoritme za uporabo.

Pri tem S_i predstavlja trenutni seštevek členov geometričnega zaporedja. Izhodišče za posamezne vmesne izračune je zmeraj *termin, ko dospeva zadnja anuiteta*.

Analiza zaokrožitvenih napak

Pri neposrednem računanju faktorjev razdolžnin geometričnega zaporedja opravimo toliko seštevanj, kot je kapitalizacijskih obdobij, s katerimi računamo, zmanjšano za 1. Ker imajo v našem primeru vsi sumandi isti predznak in imamo nizko vrednost u (ker računamo z dvojno natančnostjo je to 2^{-53}), lahko analizo zaokrožitvenih napak poenostavimo in rečemo, da je relativna napaka kvečjemu enaka številu računskih operacij $n-1$, pomnoženo z osnovno zaokrožitveno napako u (Bohte, 1995, str. 85). Pri *seštevanju*[9] imamo torej napako omejeno z $(n-1)u$. Ovrednotiti moramo še potenciranje, ki ga bomo obravnavali kot množenje več števil[10]. Tudi za množenje (v splošnem) velja, da je relativna napaka kvečjemu nu (ibid., str. 81). Pri množenju (oz. *potenciranju*) imamo torej napako omejeno z

$$\left((n-2) \times \tfrac{(n-1)}{2}\right) u,$$

pri čemer število med zunanjimi oklepaji predstavlja število vseh množenj. Skupna ocena najvišje napake za postopke *seštevanja* in *potenciranja*, v kolikor poenostavimo, je tako omejena na

$$(n-1)u + \left((n-2) \times \tfrac{(n-1)}{2}\right) u \sim \left(\frac{n^2}{2} + \mathcal{O}(n)\right) u.$$

[9] Seštevanja v $(1+p)$ ne bomo ovrednostili, saj bi iz $(1+p)$ lahko tvorili v obrestovalni faktor, vendar se za to nismo odločili, da bi zagotovili primerljiv zapis med algoritmi.

[10] Z računalnikom smo opravili izračun na dva načina: neposredno s potenciranjem in z množico množenj, rezultati so bili skoraj identični.

5.3.2 Računanje vsote členov geometričnega zaporedja rekurzivno

Namesto, da posamezne faktorje razdolžnin v geometričnem zaporedju reduciramo na isti termin, lahko kot osnovo za računanje faktorjev vzamemo *termin, ko dospeva naslednja anuiteta* (na časovni premici) in ne zadnja. Vendar mora biti ta "naslednja" anuiteta predhodno izračunana. Kar pomeni, da v tem primeru uporabljamo rekurzivni postopek. Postopek je prikazan na Algoritmu 2.

Algoritem 2: Računanje vsote členov geometričnega zaporedja rekurzivno

1. $S_0 = 0$
2. $S_i = S_{i-1} + S_{i-1}p + 1, \quad i = 1, 2, \dots, n$
3. $S = S_i$

Opombe: Priloga 2 vsebuje konkretne algoritme za uporabo.

Analiza zaokrožitvenih napak

Pri rekurzivnem računanju faktorjev razdolžnin geometričnega zaporedja opravimo toliko seštevanj, kot je kapitalizacijskih obdobij s katerimi računamo, zmanjšano za 1. Za seštevanje samo pozitivnih ali samo negativnih števil, kot že predhodno omenjeno, velja, da je relativna napaka nu (Bohte, 1995, str. 85). V Algoritmu 2 imamo 2 postopka *seštevanja*, zato je tukaj skupna napaka omejena na

$$2(n-1)u \sim 2nu.$$

Imamo pa še en postopek množenja. Za množenje, kot tudi že predhodno omenjeno, (v splošnem) velja, da je relativna napaka kvečjemu nu (Bohte, 1995, str. 85), torej imamo napako, omejeno z

$$(n-1)u \sim nu.$$

Skupna ocena najvišje napake za rekurzivni postopek za postopke *seštevanja* in *množenja* je tako

$$3(n-1)u \sim 3nu.$$

Meja najvišje napake narašča torej *linearno* glede na rast števila kapitalizacijskih obdobij. Pri neposrednem postopku – za katerega smo predhodno opravili analizo zaokrožitvenih napak – pa je ocena najvišje napake manj ugodna, saj ta narašča *superlinearno*[11].

[11] Ocena napake raste hitreje, kot pa število kapitalizacijskih obdobij. Pojem iz knjige Plestenjak (2015, str. 55).

5.3.3 Računanje vsote členov geometričnega zaporedja, kjer ti členi medsebojno ne tvorijo geometričnega zaporedja

Bohte (1995, str. 80 in 135) pri analizi zaokrožitvenih napak prikazuje postopek za izpeljavo *formule za računanje koeficienta pri osnovni zaokrožitveni napaki* u v oceni za relativno napako produkta $n + 1$ števil. Formula je sledeča:

$$\kappa_n = \frac{(1+u)^n - 1}{u}. \tag{16}$$

Vidimo lahko, da je zapis enak delu enačbe, ki računa vsoto faktorjev razdolžnin geometričnega zaporedja v formuli (pri čemer κ_n nadomestimo s S in u nadomestimo s p). Iz formule (16) Bohte razvije (ibid., str. 81 in 135) tudi postopni zapis, ki je:

$$\kappa_n = n + \frac{1}{2}n(n-1)u + \frac{1}{6}n(n-1)(n-2)u^2 + \cdots + u^{n-1}. \tag{17}$$

Iz enačbe (17) lahko hitro ugotovimo, da se posamezni členi (pri čemer odmislimo u v vsakem členu) računajo s pomočjo formule za računanje kombinacij n elementov reda i (Usenik, 1998, str. 253):

$$K_n^i = \frac{n!}{i!\,(n-i)!} = \binom{n}{i}, \tag{18}$$

pri čemer K pomeni kombinacije, i je zap. št. člena, n pa je skupno število členov.

Vzemimo primer, da imamo množico z n elementi, n = 10, zanima pa nas vrednost 3. člena zaporedja (ki, kot smo že ugotovili, ni geometrično), torej i=3 (pri tem bomo zanemarili izračune z obrestno mero: p oz. pri Bohtetu spremenljivka u). Rezultat je 120. Enačba (18) uporablja sledeč postopek:

$$K_n^i = \frac{10 \times 9 \times 8 \times 7 \times 6 \times 5 \times 4 \times 3 \times 2 \times 1}{3 \times 2 \times 1 \times 7 \times 6 \times 5 \times 4 \times 3 \times 2 \times 1} = \frac{3628800}{30240}.$$

Čibej & Ferbar Tratar (2012, str. 132) opisujeta, kako se da postopek skrajšati. Ko se pokrajša v števcu in imenovalcu, vidimo kako računa Bohte (glej enačbo 17):

$$K_n = \frac{10 \times 9 \times 8}{3 \times 2 \times 1} = \frac{720}{6}.$$

Kot lahko razberemo, je slednji postopek manj tvegan za prekoračitev obsega v vmesnem rezultatu.

Prekoračitev številskega obsega

Postopni zapis, ki ga navaja Bohte, lahko računa le največ (prvih) 170 členov zaporedja (v dvojni natančnosti). Če bi na primer seštevali 171 faktorjev razdolžnin (ali več), bi pri računanju 171. fakultete (ter višjih) pri *vmesnem koraku* prišlo do prekoračitve številskega obsega. Težava je v tem, da je npr. računski postopek za 171. člen (če je ta zadnji) sledeč: $1/171! \cdot 171!$. Točna rešitev je 1, numerična rešitev pa je: $1/Inf \cdot Inf = NaN$.

Kako bi se lahko temu izognili? Premislek gre v smeri, da se v računski postopek izmenično vnašajo vrednosti iz števca in iz imenovalca. Pri sledečem postopku so vmesni rezultati *zmeraj* med *1* in *končnim rezultatom*:

$$K_n = \frac{10}{1} \times \frac{9}{2} \times \frac{8}{3}.$$

Ker je pri omenjenem postopku potrebno za vsak izračunan člen začeti računati od znova in so pri poznih členih tudi računski postopki dolgotrajnejši, je smiselno računski postopek peljati rekurzivno. Postopek je prikazan z Algoritmom 3.

Algoritem 3: Računanje vsote členov geometričnega zaporedja, kjer ti členi medsebojno ne tvorijo geometričnega zaporedja – preurejen postopek, členi rekurzivno

1. $S_0 = 0, \quad P_0 = 0$

2. $S_i = S_{i-1} + P_i, \quad P_i = \begin{cases} n, & i = 1 \\ P_{i-1} i^{-1}(n - i + 1)p, & i > 1 \end{cases}, \quad i = 1, 2, \dots, n$

3. $S = S_i$

Opombe: P= vrednost posameznega člena zaporedja ; Priloga 3 vsebuje konkretne algoritme za uporabo.

Analiza zaokrožitvenih napak popravljenega algoritma

Pri rekurzivnem računanju faktorjev razdolžnin geometričnega zaporedja (Algoritem 3) opravimo toliko seštevanj, kot je kapitalizacijskih obdobij, s katerimi računamo, zmanjšano za 1. Za seštevanje samo pozitivnih ali samo negativnih števil, kot že predhodno omenjeno, velja, da je relativna napaka nu (Bohte, 1995, str. 85). V Algoritmu 3 imamo 2 postopka *seštevanja*, zato je tukaj skupna napaka omejena na

$$2(n-1)u \sim 2nu.$$

Imamo pa še dva postopka množenja. Za množenje, kot tudi že predhodno omenjeno, (v splošnem) velja, da je relativna napaka kvečjemu nu (Bohte, 1995, str. 85), torej imamo napako omejeno z

$$2(n-1)u \sim 2nu.$$

Imamo pa še en postopek deljenja in eno odštevanje celih števil. Tudi v tem primeru bomo opravili vrednotenje z

$$2(n-1)u \sim 2nu.$$

Skupna ocena najvišje napake za računanje vsote členov pri preurejenem postopku – rekurzivno je tako

$$6(n-1)u \sim 6nu.$$

Meja najvišje napake pri Algoritmu 3 narašča torej *linearno* glede na rast števila kapitalizacijskih obdobij.

5.3.4 Računanje vsote skupin členov geometričnega zaporedja

Ker je ocena za relativno napako enaka številu operacij, pomnoženih z osnovno zaokrožitveno napako (ibid., str. 81), lahko poskušamo zmanjšati število korakov, potrebnih za izračun vsote geometričnega zaporedja. Tako lahko npr. za posojilo, ki se odplačuje na 18 obrokov, tvorimo najprej *izhodiščno skupino* členov vsote. V tej skupini so 1., 5., 9. in 13. člen zaporedja, ki so, kot vidimo, med seboj enako oddaljeni (do te točke je postopek enak kot pri računanju členov *neposredno iz podatkov*) (Algoritem 4, korak 2 in Algoritem 2, korak 2).

Sedaj pa lahko iz te skupine členov računamo naslednjo skupino členov (z *rekurzivnim* postopkom računanja členov – Algoritem 4). V našem primeru bi v prvi ponovitvi tega postopka to bili členi 2, 6, 10 in 14, v drugi ponovitvi členi 3, 7, 11 in 15 ter v tretji ponovitvi členi 4, 8, 12 in 16 (Algoritem 4, korak 4). Tako smo vse skupaj opravili 4 izračune *neposredno iz podatkov* in 3 izračune iz skupin členov, torej *rekurzivno*, skupaj 7. Tako *velikost skupine* kot tudi *število ponovitev (+1)* sta rezultata kvadratnega korena iz števila obrokov. Morebitni celoštevilski ostanek, ki ga dobimo po korenjenju, pa potem spet računamo (posamično) *neposredno iz podatkov* (člen 17 in 18) (Algoritem 4, korak 6). Lahko bi sicer uporabili tudi katera dva druga faktorja (namesto 4*4+2, bi vzeli 2*9), vendar si s korenjenjem običajno zagotovimo najmanjše število skupno potrebnih izračunov in ohranimo pregleden računski zapis. Potrebno je še poudariti, da v koraku 4 (Algoritem 4) obrestovalni faktor ni več $(1+p)$ temveč se k p prišteje vsota geometričnega zaporedja izhodiščne skupine iz koraka 2 (Algoritem 4).

Algoritem 4: Računanje vsote skupin členov geometričnega zaporedja rekurzivno

1. $S_1 = 1, \quad K = \lfloor \sqrt{n} \rfloor$
2. $S_i = S_{i-K} + (1+p)^{i-1}, \quad i = 1K+1, 2K+1, \dots, (K-1)K+1$
3. $S1 = S_j = S_i$
4. $S_j = \begin{cases} S_i, & j = 0 \\ S_{j-1} + \left(S_{j-1}p + S1 \right), & j > 0 \end{cases}, \quad j = 0, 1, 2, \dots, K-1$
5. $S_i = S_j$
6. $S_i = \begin{cases} S_j, & n - K^2 = 0 \\ S_{i-1} + (1+p)^{i-1}, & n - K^2 > 0 \end{cases}, \quad i = K, K^2+1, K^2+2, \dots, n$
7. $S = S_i$

Opombe: K =korak, S_i seštevek členov pri potenciranju, $S1$ skupni seštevek členov pri potenciranju, S_j seštevek skupin členov pri rekurzivnem računanju. Priloga 4 vsebuje konkretne algoritme za uporabo.

Analiza zaokrožitvenih napak

Pri računanju *izhodiščne skupine* členov (Algoritem 4, korak 2) najprej opravimo za $\lfloor\sqrt{n}\rfloor - 1$ izračunov. Za *seštevanje* je ocena najvišje napake $(\lfloor\sqrt{n}\rfloor - 1)u$, za potenciranje pa je nekoliko bolj ugodna kot pri analizi zaokrožitvenih napak za potenciranje pri algoritmu 1 in je $\mathcal{O}(n)u$. Za *rekurzivno* izračunane skupine členov (Algoritem 4, korak 4) je ocena napake $\lfloor\sqrt{n}\rfloor - 2$. Za ostanek (Algoritem 4, korak 6) pa $\sim 2\lfloor\sqrt{n}\rfloor$. Skupna ocena najvišje napake je tako

$$\mathcal{O}(n)u + \sim 4\lfloor\sqrt{n}\rfloor \sim \mathcal{O}(n)u.$$

5.3.5 Računanje vsote geometričnega zaporedja iz skupin obrestnih mer v dvojiških korakih

Za računanje vsote geometričnega zaporedja običajno pretvorimo obrestno mero p v obrestovalni faktor $(1 + p)$. Tako lahko izračunamo faktor (večkratnik) izbranega člena glede na izhodiščni člen. Velikost 3. člena pri p = 50 %, bi lahko npr. računali: $(1 + 50\ \%)(1 + 50\ \%)$. Rezultat bi bil torej $(1{,}5)(1{,}5) = 2{,}25$. Lahko pa poskusimo v izhodišču računati tudi samo z obrestno mero p, postopek izračuna bi bil potem takšen: $\big((50\ \%)(50\ \%) + (50\ \%) + (50\ \%)\big) + 1$. Pri tem bi spet dobili enak rezultat: $\big((0{,}5)(0{,}5) + (0{,}5) + (0{,}5)\big) + 1 = 2{,}25$.

Kako bi lahko del $((1 + p)^n - 1\ )$ v formuli nadomestili s pravkar opisanim postopkom. Ker smo že izračunali, da je:

- $(1 + p)^n = \big((p)(p) + (p) + (p) \ldots\big) + 1$, je potemtakem
- $(1 + p)^n - 1 = \big((p)(p) + (p) + (p) \ldots\big)$.

Kar pomeni, da bomo pri uporabi *obrestovalnega faktorja* naleteli na težave z odštevanjem, če pa pri izračunu uporabimo le *obrestno mero* te težave nimamo. Če uporabimo le *obrestno mero,* se torej izognemo odštevanju. Zato pa moramo preurediti postopek, kar pomeni, da moramo izračun opraviti postopoma. V predhodnem odstavku smo prikazali, kako opraviti posamezen izračun brez odštevanja. Vendar pa v našem primeru potrebujemo množico takšnih izračunov, kako jih torej povezati.

Na Sliki 10 je prikazan postopek s primerom izračuna za to. V našem konkretnem primeru ni več potrebno prišteti na koncu števila 1, saj bi ga – kot smo že ugotovili – morali spet odšteti (kar bi bilo po eni strani nepotrebno, predvsem pa bi pomenilo nevarnost za stabilno računanje).

Slika 10: Postopek za potenciranje (celi eksponent), neposredno iz obrestne mere (*p* = 50 %), v koraku se uporabita rezultat predhodnega koraka ter izhodiščna obrestna mera

$$(1 + p)^1 = \underbrace{p}_{x_1} + 1;\ \underbrace{0{,}5}_{0.5} + 1 = 1{,}5.$$
$$(1 + p)^2 = \underbrace{px_1 + p + x_1}_{x_2} + 1;\ \underbrace{0{,}5 \cdot 0{,}5 + 0{,}5 + 0{,}5}_{1{,}25} + 1 = 2{,}25.$$
$$(1 + p)^3 = \underbrace{px_2 + p + x_2}_{x_3} + 1;\ \underbrace{0{,}5 \cdot 1{,}25 + 0{,}5 + 1{,}25}_{2{,}375} + 1 = 3{,}375.$$
$$\ldots$$

$$(1+p)^{n} = \underbrace{px_{n-1} + p + x_{n-1}}_{x_n} + 1; \underbrace{0{,}5 \cdot x_{n-1} + 0{,}5 + x_{n-1}}_{\dots} + 1 = \dots + 1.$$

Vendar konkretna uporaba postopka s Slike 10 v primerjavi z Algoritmi 1, 2 in 3 ne bi prinesla veliko, saj je tudi v tem primeru število korakov vezano na število kapitalizacijskih obdobij.

S področja računalništva je znan[12] postopek, ko lahko posamezni argument izrazimo z njegovimi (različnimi vnaprej določenimi) parametri (lastnostmi). Te parametre lahko zapišemo z besedo, kar npr. pomeni, da je potem argument izražen s tremi besedami. Imajo pa ti parametri tudi vsak *svojo* vrednost (so torej konstante). Tako ima *prvi parameter* vrednost 1, *drugi* vrednost 2, *tretji* vrednost *4*, *četrti* vrednost 8 … Če sedaj seštejemo vrednosti posameznih parametrov, s katerimi želimo izraziti argument, dobimo (eno) število, ki ga prav tako lahko uporabimo za ponazoritev posameznih parametrov argumenta. S tem postopkom lahko torej sestavimo posamezno celo število. V kolikor ta način prenesemo na področje, ki ga obravnavamo, lahko hitro ugotovimo, da lahko z njim izrazito zmanjšamo skupno število potrebnih izračunov. Postopek je prikazan na Sliki 11.

Slika 11: Postopek za potenciranje (celi eksponent), neposredno iz obrestne mere (p = 50 %), v koraku sta obe spremenljivki rezultat predhodnega koraka – z dopolnilnim izračunom

$$(1+p)^{2^0} = \underbrace{p}_{x_0} + 1; \underbrace{0{,}5}_{0.5} + 1 = 1{,}5$$

$$(1+p)^{2^1} = \underbrace{{x_0}^2 + 2x_0}_{x_1} + 1; \underbrace{0{,}5^2 + 2(0{,}5)}_{1{,}25} + 1 = 2{,}25.$$

$$(1+p)^{2^2} = \underbrace{{x_1}^2 + 2x_1}_{x_2} + 1; \underbrace{1{,}25^2 + 2(1{,}25)}_{4{,}0625} + 1 = 5{,}0625.$$

$$(1+p)^{2^3} = \underbrace{{x_2}^2 + 2x_2}_{x_3} + 1; \underbrace{4{,}0625^2 + 2(4{,}0625)}_{24{,}6289\dots} + 1 = 25{,}6289\dots$$

…

$$(1+p)^{2^{\mathbb{N}}} = \underbrace{{x_{\mathbb{N}-1}}^2 + 2x_{\mathbb{N}-1}}_{x_{\mathbb{N}}} + 1; \underbrace{(\dots)^2 + 2(\dots)}_{\dots} + 1 = \dots + 1.$$

Sestaviti še je potrebno izbrano celo število (primer izračuna za vrednost 11)

$$(1+p)^{2^0+2^1+2^3} =$$

$$\left(\underbrace{0{,}5}_{0.5}\right)\left(\underbrace{0{,}5^2 + 2(0{,}5)}_{1{,}25}\right) = \left(\underbrace{0{,}5 \cdot 1{,}25 + 0{,}5 + 1{,}25}_{2{,}375}\right) = x$$

[12] Primer za to je 2. argument MsgBox funkcije v programskem jeziku VBA.

$$x\left(\underbrace{4{,}0625^2 + 2(4{,}0625)}_{24{,}62\ldots}\right) = \left(\underbrace{2{,}375 \cdot 24{,}62\ldots + 2{,}375 + 24{,}62\ldots}_{85{,}49\ldots}\right) = 85{,}49$$

Celotni postopek je prikazan na Algoritmu 5. V koraku 1 in 2 ugotavljamo, koliko izračunov bo potrebnih, ter ustvarimo prazen vektor, ki lahko shrani te posamezne izračune. V koraku 3 opravimo te izračune in jih shranimo v vektor (kot je tudi prikazano na Sliki 11). V koraku 5 določimo, katere predhodne izračune bomo tudi zares uporabili in iz njih naredimo končni izračun. Tako bi pri številu kapitalizacijskih obdobij 11 ($= 2^{0+1} + 2^{1+1} + 2^{3+1}$) za izračun uporabili 1., 2. in 4. rezultat, ki smo ga predhodno shranili v vektor. V koraku 6 predhodni rezultat še delimo z obrestno mero, da dobimo vsoto členov geometričnega zaporedja.

Algoritem 5: Računanje vsote geometričnega zaporedja iz skupin obrestnih mer v dvojiških korakih

1. $ne = \lfloor \log_2 n \rfloor + 1$
2. $V = [v_1 \quad v_2 \quad \cdots \quad v_{ne}]$
3. $v_i = \begin{cases} p, & i = 1 \\ {v_{i-1}}^2 + 2v_{i-1}, & i > 1 \end{cases}, \quad i = 1, 2, \ldots, ne$
4. $nt_0 = 2n, \; S_0 = 0$
5. $S_i = \begin{cases} S_{i-1}, & \frac{nt_i}{2} = \left\lfloor \frac{nt_i}{2} \right\rfloor \\ S_{i-1}v_i + S_{i-1} + v_i, & \frac{nt_i}{2} \neq \left\lfloor \frac{nt_i}{2} \right\rfloor \end{cases}, \quad nt_i = \left\lfloor \frac{nt_{i-1}}{2} \right\rfloor, \quad i = 1, 2, \ldots, ne$
6. $S = \frac{S_i}{p}$

Opombe: ne – število elementov v vektorju, V – vektor, v –elementi v vektorju, nt –trenutno število kapitalizacijskih obdobij; Priloga 5 vsebuje konkretne algoritme za uporabo.

Analiza zaokrožitvenih napak

Pri računanju posameznih členov, ki jih hranimo v vektorju opravimo za $\lfloor \log_2 n \rfloor + 1$ postopkov, ki vsebujejo vsak 2 množenji in seštevanje. Torej je najvišja napaka omejena na kvečjemu $\sim 3\lfloor \log_2 n \rfloor u$. Pri določitvi končnega seštevka uporabimo ponovno $\lfloor \log_2 n \rfloor + 1$ izračunov, v katerem imamo 2 seštevanji in množenje. Torej je tudi v drugem delu računskega postopka napaka omejena na $\sim 3\lfloor \log_2 n \rfloor u$. Skupaj torej znaša ocena napake

$$\sim 3\lfloor \log_2 n \rfloor u + \sim 3\lfloor \log_2 n \rfloor u \sim 6\lfloor \log_2 n \rfloor u.$$

Kar pomeni, da bi uporaba tega postopka morala vrniti zelo dober rezultat. V kolikor npr. predpostavljamo, da je n celo število s 15 števkami, potem je najvišja mogoča napaka (pri dvojni natančnosti) $\sim 300u$.

5.3.6 Povzetek analize zaokrožitvenih napak za postopke računanja vsote geometričnega zaporedja

Število računskih postopkov pri Algoritmih 1, 2 in 3 je v izhodišču vezano na število kapitalizacijskih obdobij. Vsak tak računski postopek pa pri posameznem izmed teh algoritmov vsebuje različno število računskih operacij – pri postopku računanja *neposredno* iz vhodnih podatkov ter pri postopku računanja *rekurzivno* je število računskih operacij, ki jih vsebujeta v posameznem obhodu/koraku, podobno: 4 oz. 3. Pri tem da bolj ugodno oceno za relativno napako metoda računanja *rekurzivno*, ta namreč ne vsebuje potenciranja, ki ga lahko ovrednotimo kot niz množenj, kar pomeni dodatno število računskih operacij, ter poslabšanje ocene predvidene relativne napake. Zaradi računskega postopka potenciranja se torej ocena rasti napake glede na rast števila kapitalizacijskih obdobij – pri postopku računanja *neposredno* iz vhodnih podatkov – ki je v izhodišču ocenjena kot linearna "spremeni" v superlinearno. Za (popravljen) Algoritem 3 postopek – kjer *členi vsote niso neposredno povezani z geometričnim zaporedjem* – velja enako kot za Algoritem 2.

Tabela 5: Primerjava ocene direktne analize zaokrožitvenih napak pri metodah za postopno računanje členov geometričnega zaporedja

Št. alg.		Ocena za relativno napako	Rast napake glede na rast števila kapitalizacijskih obdobij	Število obhodov zank/-e v algoritmu (število računskih operacij v posamezni zanki)
1.	Členi – *neposredno*	$\sim \left(\frac{n^2}{2} + \mathcal{O}(n)\right) u$	superlinearno	$n\ (4)$
2.	Členi – *rekurzivno*	$\sim \mathcal{O}(\mathrm{n})\mathrm{u}$	linearno	$n\ (3)$
3.	Členi – negeometrično (popravljena) – *rekurzivno*	$\sim \mathcal{O}(\mathrm{n})\mathrm{u}$	linearno	$n\ (6)$
4.	Skupine členov s korenjenjem – *rekurzivno*	$\sim \mathcal{O}(\mathrm{n})\mathrm{u}$	linearno	med $2\lfloor\sqrt{n}\rfloor - 1$ in $4\lfloor\sqrt{n}\rfloor - 1$ $(4 + 3 + 4)$
5.	Skupine obrestnih mer v dvojiških korakih – *rekurzivno*	$\sim \mathcal{O}(\log_2 \mathrm{n})\mathrm{u}$	sublinearno	$\lfloor\log_2 \mathrm{n}\rfloor + 1$ $(5 + 5)$

Opombe: $\mathcal{O}$= velikostni razred; zaporedne številke v tabeli označujejo zaporedno številko algoritma oz. pripadajočih prilog.

Obe računski metodi, ki ne računata več posameznih členov vsote geometričnega zaporedja posamezno, temveč po skupinah (vsaka metoda na svoj način), lahko izpostavimo pri metodi, ki računa *skupine členov geometričnega zaporedja s korenjenjem*, da tudi ta vsebuje potenciranje v dveh izmed treh korakov in se zato ocena predvidene najvišje relativne napake pri tej metodi iz začetka zelo ugodne

(sublinearne[13]-) "spremeni" na raven ocene, ki jo ima že predhodno opisan postopek računanja posameznih členov *rekurzivno* – Algoritem 2, torej gre za linearno rast napake glede na rast števila kapitalizacijskih obdobij. Po drugi strani pa metoda računanja *skupin obrestnih mer v dvojiških korakih – rekurzivno* ohranja zelo ugodno oceno predvidene najvišje relativne napake in je s tem edini postopek, pri katerem je rast napake glede na rast števila kapitalizacijskih obdobij sublinearna.

Iz povzetega lahko torej izpeljemo, da so postopki (torej Algoritem 2, Algoritem 3 in Algoritem 5), ki ne vsebujejo računske operacije potenciranja (oz. računanj fakultet), tudi ohranili korelacijo med rastjo števila kapitalizacijskih obdobij in rastjo ocene predvidene najvišje relativne napake. Pri ostalih dveh algoritmih se je ocena poslabšala, saj je potenciranje (v našem primeru) mogoče ovrednotiti kot računski postopek, ki vsebuje dodatno število računskih postopkov. Omenjeni trije imajo – skupaj s postopkom, ki računa *skupine členov geometričnega zaporedja s korenjenjem* – tudi absolutno gledano najbolj ugodno oceno predvidene najvišje relativne napake glede na število kapitalizacijskih obdobij n linearna- in sublinearna rast) .

[13] Ocena napake raste počasneje, kot pa število kapitalizacijskih obdobij. Pojem iz knjige Plestenjak (2015, str. 55).

6 PRIMERJAVA TOČNOSTI IZRAČUNOV IN UČINKOVITOSTI ALGORITMOV

V minulem poglavju smo opisali postopke, s katerimi stabilno računamo enake kreditne anuitete. Opravili smo tudi analizo zaokrožitvenih napak za te postopke, kar pomeni, da smo ocenjevali najvišjo pričakovano napako numerične rešitve v primerjavi s točno rešitvijo. Kriterija, s katerima bomo merili primernost posamezne metode za računanje kreditnih anuitet, bosta natančnost in ekonomičnost. Merilo za natančnost bo računalo operacijskega sistema Windows, ki v znanstvenem načinu računa na 32 števk natančno. V postopek primerjave bomo vključili tudi Excelovo PMT funkcijo (za računanje kreditnih odplačil), ki glede na preizkuse prav tako uporablja numerično stabilen algoritem. Za merilo ekonomičnosti bomo uporabili časovno zahtevnost algoritmov, kar pomeni, da bomo ugotavljali, kolikšen čas je potreben za posamezen izračun. Meritve bomo opravili s pomočjo programskega paketa Octave.

6.1 Numerična nestabilnost formul pri obrestni meri: blizu 0 %

V prejšnjem poglavju smo predstavili 5 načinov za stabilno računanje anuitet. Za vsakega smo opravili analizo zaokrožitvenih napak. Opravili pa bomo še konkretne izračune, s katerimi bomo ugotavljali, kateri algoritem je glede na dane okoliščine smiselno izbrati.

Ugotovili smo torej, da pride pri izračunu s formulo do velike relativne napake v rezultatu, ko je obrestna mera blizu 0 %. Poglejmo sedaj, kako ob konkretnih podatkih delujejo dopolnilni postopki za računanje kreditnih anuitet: opravili bomo primerjavo izračunov, ki jih dobimo s formulo in z dopolnilnimi metodami za 3 različne ročnosti ob enaki (nizki) obrestni meri.

Če za izračun uporabimo podatke $D_0 = 100000$, $p = 10^{-6}$ %, $n_1 = 1$, $n_2 = 360$, $n_3 = 100000$, dobimo naslednje rezultate.

Tabela 6: Rezultati – in čas trajanja izračuna postopkov za računanje kreditnih anuitet pri $p = 10^{-6}$ %

Št. alg.		$n_1 = 1$	$n_2 = 360$	$n_3 = 100000$
	točen izračun *(trajanje izračuna)*	–100000,001000000 *(x/1000s)*	–277,778279166967 *(x/1000s)*	–1,00050008833333 *(x/1000s)*
	Excel-PMT funkcija	–100000,001000000 *(0/1000s)*	–277,778279166967 *(0/1000s)*	–1,00050008833333 *(0/1000s)*
	Formula	–100000,001607747 *(0/1000s)*	–277,778280860210 *(0/1000s)*	–1,00050009441089 *(0/1000s)*
1.	Členi – *neposredno*	–100000,001000000 *(0/1000s)*	–277,778279166964 *(4/1000s)*	–1,00050008833030 *(590/1000s)*
2.	Členi – *rekurzivno*	–100000,001000000 *(0/1000s)*	–277,778279166967 *(4/1000s)*	–1,00050008833335 *(520/1000s)*

Št. alg.		$n_1 = 1$	$n_2 = 360$	$n_3 = 100000$
3.	Členi – negeometrično (popravljena) – *rekurzivno*	–100000,001000000 *(0/1000s)*	–277,778279166967 *(10/1000s)*	–1,00050008833333 *(1050/1000s)*
4.	Skupine členov s korenjenjem – *rekurzivno*	–100000,001000000 *(0/1000s)*	–277,778279166970 *(2/1000s)*	–1,00050008833636 *(14/1000s)*
5.	Skupine obrestnih mer v dvojiških korakih – *rekurzivno*	–100000,001000000 *(1/1000s)*	–277,778279166967 *(2/1000s)*	–1,00050008833333 *(3/1000s)*

Opombe: Napačne števke so označene z rdečo in osenčene; trajanje izračuna je na tisočinke sekunde natančno. V tabelo smo vnesli tiste vrednosti, ki so se pri več poskusih posameznih izračunov največkrat izpisale; točne vrednosti so izračunane z računalom operacijskega sistema Windows, njihova vrednost na 32 števk natančno je: pri n_1 –100000,001, pri n_2 –277,778279166966643503517986359 3 in pri n_3 –1,00050008833333151944450116600 34.

Pri izračunu za eno kapitalizacijsko obdobje ($n_1 = 1$), dajo vsi postopni načini rezultat, ki je točen, kar ustreza predpostavki, da nastane pri eni računski operaciji napaka, ki je kvečjemu velikosti osnovne zaokrožitvene napake u. Če se bolj natančno izrazimo, gre pri posameznem postopku v resnici za *do 3 zanke*, ki vsebujejo *do 5 izračunov*. Vendar je to še vedno premalo za uresničitev (pesimistične) ocene za relativno napako izračuna. V rezultatu izračuna s pomočjo formule pa je zadnjih 6 števk napačnih. Z vidika časovne zahtevnosti lahko rečemo, da je pri tej ročnosti ($n_1 = 1$) hitrost izračuna tako visoka, da je ni mogoče izmeriti. Le metoda, ki računa *skupine obrestnih mer v dvojiških korakih – rekurzivno* (Algoritem 5), je časovno rahlo bolj potratna. To gre pripisati temu, da omenjen algoritem vzpostavi vektor, opravi izračun z dvema zankama ter opravi še en funkcijski izračun. Pri tej ročnosti je razlika v časovni zahtevnosti posameznih algoritmov resda zanemarljiva, vendar bi tukaj kljub temu predlagali uporabo algoritma, ki računa *vsote členov geometričnega zaporedja rekurzivno* (Algoritem 2 v Tabeli 6). Slednji potrebuje za izračun namreč le 3 računske operacije (glej Tabelo 5), kar je najmanj med vsemi algoritmi. Če pogledamo izračune pri višjih ročnostih ($n_3 = 100000$), lahko sklepamo, da je v primerjavi z Algoritmom 1 (ki prav tako potrebuje n obhodov zanke), tudi zares nekoliko hitrejši.

Pri višjih ročnostih $n_2 = 360$ je rezultat, ki ga da formula, na zadnjih 8 števkah napačen, kar pomeni, da ni prišlo do bistvene spremembe v relativni napaki, na kar bi vplivala spremenjena obrestna mera. Pri algoritmih pa je zaznati prva rahla odstopanja v natančnosti izračuna. To se zgodi pri Algoritmih 1 in 4, ki imata to skupno lastnost, da se njuni posamezni členi (v celoti ali deloma) računajo neposredno iz potence. Slednji sicer omogoča najhitrejši izračun pri tej ročnosti $n_2 = 360$, kar je rezultat tega, da v primerjavi z Algoritmi 1, 2 in 3 zmanjša število potrebnih izračunov na velikostni razred kvadratno korenjenega števila kapitalizacijskih obdobij n. Ker pa je tudi Algoritem 5 enako hiter kot Algoritem 4 ter hkrati na vseh 15 mest natančen, je smotrno izbrati prvo omenjenega.

Ob računanju z zelo visokim številom kapitalizacijskih obdobij $n_3 = 100000$ pridemo do sledečih ugotovitev. Tudi v tem primeru je zadnjih 8 števk v rezultatu pri formuli napačnih. Pri Algoritmih 1 in 4 se je predhodno ugotovljena napaka še rahlo povečala, pri Algoritmu 2 pa zasledimo odstopanje na zadnji števki. Algoritma 3 in 5 ostajata edina, ki vrneta rezultat na vseh 15 števk natančno, vendar je razlika v hitrosti, ki jo potrebujeta za izračun, velika. Algoritem, ki temelji na računanju *členov negeometrično* potrebuje za izračun več kot 300-krat toliko časa, kot ga potrebuje algoritem za računanje *skupin obrestnih mer v dvojiških korakih – rekurzivno*. Pri zelo visokem številu kapitalizacijskih obdobij namreč pride do izraza način izračuna po dvojiških korakih, torej, ko vsak izračun vsebuje dvakratnik izračunov iz prejšnjega koraka računanja. Takšen postopek potem tudi v primerjavi z Algoritmom 4 – ki zmanjša število potrebnih izračunov na velikostni razred kvadratnega korena iz števila kapitalizacijskih obdobij n – dokaj močno pridobi.

Iz Tabele 6 lahko razberemo, da so edini računski postopki, ki dajo v vseh treh primerih točen rezultat na vseh 15 števk natančno, naslednji: postopek za *računanje členov kot kombinacije (*popravljena Bohte*) – rekurzivno,* postopek za računanje *skupin obrestnih mer v dvojiških korakih – rekurzivno* ter Excelova PMT funkcija. Z namenom, da bi ugotovili, ali kateri izmed omenjenih (dveh) algoritmov uporablja enak algoritem kot ga uporablja PMT funkcija, smo naredili še dodatne izračune. Pri tem smo uporabili še nižjo obrestno mero na kapitalizacijsko obdobje kot v prejšnjem primeru. V Tabeli 7 so prikazani rezultati te primerjave. Pri ročnosti $n_1 = 10000$ je rezultat PMT funkcije drugačen od rezultata Algoritma 3, pri $n_2 = 100000$ pa je drugačen od rezultata Algoritma 5, kar pomeni, da ne moremo izhajati iz tega, da bi PMT funkcija vsebovala katerega izmed predstavljenih algoritmov. Iz Tabele 7 lahko razberemo tudi, da Algoritem 3 – ki računa *člene negeometrično* – pri ročnosti $n_1 = 10000$ vrne bolj natančen rezultat kot pa Excelova PMT funkcija.

Tabela 7: Rezultati in čas trajanja izračuna postopkov za računanje kreditnih anuitet pri $D_0 = 100000$, $p = 10^{-12}$ %, $n_1 = 10000$, $n_2 = 100000$

Št. alg.		$n_1 = 10000$	$n_2 = 100000$
	točen izračun *(trajanje izračuna)*	–10,0000000005001 *(x/1000s)*	–1,00000000050001 *(x/1000s)*
	Excel-PMT funkcija	–10,0000000005000 *(0/1000s)*	–1,00000000050001 *(0/1000s)*
	Formula	–10,00799917293<u>44</u> *(0/1000s)*	–1,00079991819<u>344</u> *(0/1000s)*
1.	Členi – *neposredno*	–10,000000000499<u>7</u> *(72/1000s)*	–1,0000000004996<u>2</u> *(590/1000s)*
2.	Členi – *rekurzivno*	–10,0000000005001 *(62/1000s)*	–1,00000000050001 *(520/1000s)*
3.	Členi – negeometrično (popravljena) –*rekurzivno*	–10,0000000005001 *(100/1000s)*	–1,00000000050001 *(1050/1000s)*
4.	Skupine členov s korenjenjem – *rekurzivno*	–10,000000000500<u>4</u> *(5/1000s)*	–1,0000000005003<u>9</u> *(14/1000s)*

Št. alg.		$n_1 = 10000$	$n_2 = 100000$
5.	Skupine obrestnih mer v dvojiških korakih – *rekurzivno*	–10,0000000005000 *(3/1000s)*	–1,00000000050000 *(3/1000s)*

Opombe: Napačne števke so označene z rdečo in osenčene oz. so podčrtane; trajanje izračuna je na tisočinke sekunde natančno, v tabelo smo vnesli tiste vrednosti, ki so se pri več poskusih posameznih izračunov največkrat izpisale; vrednosti so izračunane z računalom operacijskega sistema Windows, njihova vrednost na 32 števk natančno je: pri n_1 –10,0000000005000500000083366590 62, pri n_2 –1,0000000005000050000833366575 24.

Pri povzetku analize zaokrožitvenih napak v prejšnjem poglavju smo ugotovili, da so Algoritem 2, Algoritem 3 in Algoritem 5 postopki, ki ne vsebujejo računske operacije potenciranja (oz. računanj fakultet) in so zato tudi ohranili korelacijo med rastjo števila kapitalizacijskih obdobij in rastjo ocene predvidene najvišje relativne napake. Primerjalni izračuni, opravljeni v tem poglavju, potrjujejo te ugodne ocene. Algoritem 2, Algoritem 3 in Algoritem 5 so tudi edini računski postopki (v primerjavi), ki v celoti računajo rezultate rekurzivno. V našem primeru torej ne drži priporočilo, o katerem piše Datta (2010, str. 42), namreč da se je potrebno izogibati rekurzivnih postopkov, saj ti lahko napake v vmesnih rezultatih še povečujejo. Kot smo že omenili, je lahko razlog dobrih rezultatov uporabe rekurzivnih postopkov v tem, da ti – v primerjavi s postopki, ki vsebujejo potenciranja – realno ne povečajo števila računskih operacij.

6.2 Točnost izračunov v okviru najbližjega predstavljivega števila oz. sosednjega predstavljivega števila

V minulem poglavju smo že ugotovili, da so:

- Algoritem 3
- Algoritem 5 in
- PMT funkcija

računski postopki, s katerimi dobimo *zelo* natančne rezultate. Kaj to pomeni? Ker pri numeričnem računanju običajno ne moremo dobiti točne rešitve, smo zadovoljni z rezultatom, ki je *najbližje predstavljivo število*. V kolikor nam to ne uspe, smo morebiti zadovoljni tudi z drugim *sosednjim številom*, t. j. poleg *najbližjega predstavljivega števila* še kot rezultat morebiti pričakujemo *drugo najbližje predstavljivo število*.

Pri računanju s temi (in drugimi) računskimi postopki smo ugotovili, da nobeden izmed njih

- ne vrne rezultata, ki je zmeraj *najbližje predstavljivo število*, čeprav po drugi strani
- vsak izmed računskih postopkov, ki smo jih našteli v vrstičnem seznamu vrne rezultat, ki je zmeraj *sosednje predstavljivo število.*

Ne da se pa napovedati, kateri algoritem je v katerih okoliščinah najbolj smiselno uporabiti. Vidik, ki ga lahko ocenimo je čas trajanja računskega postopka. Na vsak način ima po tem kriteriju Algoritem 5 pred Algoritmom 3 veliko prednost, saj je pri višjih ročnostih izrazito hitrejši. Hitrost izračuna je lahko povezana tudi s programskim paketom, s katerim se računa. Tako je programski jezik VBA kompiliran programski jezik,

Octave in R pa sta npr. *Interpreted* programska jezika. Kompilirani jeziki so hitrejši. Razlika v hitrosti branja programske kode med tipoma programskih jezikov je lahko več 10-krat.

6.3 Okoliščine uporabe algoritmov

Znesek katerekoli valute je običajno sestavljen iz celega dela, ki ima določeno število števk (mest), ki navzgor vsaj teoretično ni omejeno, ter iz decimalnega dela, ki pa ima fiksno število števk (mest), po navadi 2. Ob konkretnem izračunu torej napaka, ki bi nastala pri računanju na 3., 4. ali višjem decimalnem mestu, ne bi več pomenila realnega oškodovanja, saj se je ne bi dalo ponazoriti s konkretnim zneskom.

V praksi to pomeni, da bo izračunano anuitetno odplačilo imelo enako *relativno napako* pri najetem posojilu v višini 1.000 denarnih enot kot tudi v višini 100.000 denarnih enot. Vendar pa je za obe pogodbeni strani zanimiva predvsem *absolutna napaka,* saj odraža znesek, ki ga bosta plačevali bodisi preveč ali pa tudi premalo. V kolikor npr. predpostavljamo, da je izračun za posamezno anuitetno odplačilo natančen na 5 števk, potem je izračun/znesek 111,1111 denarnih enot[14] pravičen, po drugi strani pa bosta pri izračunu/znesku 11111,11 denarnih enot pogodbeni strani plačevali za do 0,50 denarne enote bodisi preveč oz. premalo.

Ker so zneski izraženi v fiksni – in ne v premični piki – se bodo napačne števke v algoritmih pokazale kot relevantne šele pri *nominalno visokih zneskih posojil*. Visoko število kapitalizacijskih obdobij (pri enaki višini posojila) pa po drugi strani zmanjšuje absolutno vrednost anuitetnih odplačil, kar pomeni, da napačne števke pri *nizkem številu kapitalizacijskih obdobij* bolj vplivajo na točnost rezultata. Izhodišče za nastanek napačnih števk, pa je, kot smo že ugotovili, *izrazito nizka obrestna mera na obdobje odplačila*, ki pa jo lahko pričakujemo, kadar so *obdobja med odplačili zelo kratka* (Če bi na primer računali obrestno mero iz letne v obrestno mero na 1 sekundo, bi n znašal 31,536,000), naprej pa lahko ob zelo kratkih obdobjih med odplačili pričakujemo, da imamo zelo visoko število kapitalizacijskih obdobij. S čimer lahko zaključimo krogotok oz. ga ponovno začnemo.

V realnih okoliščinah se ne bo pogosto zgodilo, da bomo naleteli na kombinacijo *visokega zneska posojila*, *nizkega števila kapitalizacijskih obdobij*, *izrazito nizke obrestne mere na posamezno obdobje odplačila* in *zelo kratkimi obdobji med odplačili.* Kar pomeni, da je verjetnost da dobimo napačen izračun pri računanju v *dvojni natančnosti*, ki je merljiv v realnem znesku, dokaj majhna. Zato je potrebno pretehtati, kdaj je računanje samo s formulo zadostno ter kdaj je smiselna uporaba teh dopolnilnih algoritmov. Slednje je sicer mogoče uporabiti v splošnem za računanje vsote členov geometričnega zaporedja za področja, kjer se zahteva "popolna" natančnost izračunov.

[14] Predpostavljamo, da je najmanjša realna vrednost denarne enote 1/100 le-te, ter da smo zadnjo pravilno števko pridobili z zaokrožanjem.

7 SKLEP

V diplomskem delu smo predstavili, kako stabilno računati periodično enaka odplačila anuitetnega kredita s pomočjo računalnika. Pri tem igra osrednjo vlogo obrazec za *direktno* računanje geometričnega zaporedja – znotraj formule za računanje anuitet. Pri nizki obrestni meri tako v tem obrazcu pride do odštevanja dveh približno enakih števil. Samo odštevanje sicer ne bi predstavljalo težave, če bi odštevali števili, katerih vrednost je v računalniku zapisana točno. V našem primeru pa se pred odštevanjem opravi računski postopek potenciranja, pri katerem se običajno pojavi določena napaka, ki pa še ni problematična. Težava nastopi šele potem, ko smo z odštevanjem razkrili določeno napako (iz postopka potenciranja) in jo potem delili z zelo nizkim številom – to je z nizko obrestno mero. Kar pomeni, da je relativna napaka ostala enaka, absolutna napaka pa se je povečala. Pri tem računski sistem, ki ga predstavlja obrazec za računanje vsote geometričnega zaporedja, *ni občutljiv* na sprememba podatka – torej obrestne mere – ki gre proti 0 %. Kar pomeni, da imamo nestabilno računsko metodo.

Rešitev problema predstavlja postopno računanje vsote geometričnega zaporedja. V kolikor imamo namreč v samem postopku le množenje več števil oz. seštevanje več števil (ki so vsa pozitivna ali vsa negativna), potem je takšen postopek numerično stabilen. V literaturi se za ta primer (računanja anuitete) običajno navaja postopek, ki preračuna vsa anuitetna odplačila na *en* termin. Lahko pa se uporabi postopek, ki preračunava posamezne vrednosti glede na predhodni izračun, kar pomeni torej, da gre za *rekurzivno* računanje. Slednji daje pri visokem številu kapitalizacijskih obdobij bolj natančne rezultate, saj se nominalno število računskih operacij, ki jih potrebuje za izračun, v teku postopka ne spremeni. Postopek, ki preračunava na *en* termin, pa v postopku uporablja tudi potenciranje, kar je mogoče šteti kot niz množenj, torej gre za dodatno število korakov, kar se pri večjem številu kapitalizacijskih obdobij odraža v obliki manjše natančnosti.

Če *kombiniramo oba načina*, lahko število aritmetičnih operacij zmanjšamo na velikostni razred kvadratnega korena iz števila kapitalizacijskih obdobij. S tem postopkom sicer zelo pridobimo na hitrosti, vendar pa natančnost ostaja na ravni prvoomenjenega postopka, ki preračunava na *en* termin. Lahko pa dosežemo oboje: torej majhno število korakov in hiter izračun. To nam omogoča računanje s postopkom, ki opravlja izračune iz skupin obrestnih mer, ki jih tvori v dvojiških korakih, ter shrani v vektorju. V drugem koraku iz teh elementov vektorja sestavi naravno število, ki je enako številu kapitalizacijskih obdobij. Za število kapitalizacijskih obdobij, ki je zapisano s 15 mesti, potrebuje le približno 100 računskih obhodov. Numerični rezultat takšnega računanja pa je levo ali desno sosednje število točnega rezultata.

Na ta način se k Excelovi *PMT funkciji*, ki pri računanju prav tako vrne rezultat, ki je sosednje predstavljivo število, pridruži še učinkovita in izredno stabilna metoda, ki računa *skupine obrestnih mer v dvojiških korakih – rekurzivno*. V to skupino, pa sodi še ena metoda. Za to smo *preuredili enačbo, ki jo* pri analizi zaokrožitvenih napak *uporablja Bohte* (1995, str. 81). Čeprav ta formula potrebuje toliko obhodov zanke, kot je število kapitalizacijskih obdobij, pa to ne vpliva na visoko natančnost izračuna.

LITERATURA IN VIRI

1. Bohte, Z. (1995). *Uvod v numerično računanje.* Ljubljana: Društvo matematikov, fizikov in astronomov Slovenije.

2. Burden, R. L., & Faires, J. D. (2011). *Numerical analysis.* Australia: Brooks/Cole.

3. Čibej, J. A., & Ferbar Tratar, L. (2012). *Matematika za poslovno rabo (2. del).* Ljubljana: Ekonomska fakulteta.

4. Datta, B. N. (2010). *Numerical Linear Algebra and Applications (2nd Ed.).* Philadelphia: Society for Industrial and Applied Mathematics.

5. Higham, N. J. (1996). *Accuracy and Stability of Numerical Algorithms.* Philadelphia: Society for Industrial and Applied Mathematics.

6. Marovt, J., & Breznik, K. (2014). *Praktikum iz poslovno-finančne matematike.* Maribor: Fakulteta za naravoslovje in matematiko.

7. Orel, B. (1997). *Osnove numerične matematike.* Ljubljana: Fakulteta za računalništvo in informatiko.

8. Petrišič, J. (2006). *Reševanje enačb.* Ljubljana: Fakulteta za strojništvo.

9. Plestenjak, B. (2015). *Razširjen uvod v numerične metode.* Ljubljana: Društvo matematikov, fizikov in astronomov Slovenije.

10. Usenik, J. (1998). *Matematične metode.* Novo mesto: Visoka šola za upravljanje in poslovanje.

11. Wikipedia. (2015). *Amortization calculator - Wikipedia, the free encyclopedia*. Prevzeto 28. april 2016 iz Wikipedia, the free encyclopedia: https://en.wikipedia.org/wiki/Amortization_calculator.

PRILOGE

Priloga 1: Računanje anuitete – vsota členov geometričnega zaporedja neposredno iz vhodnih podatkov (opisni algoritem, za Excel-VBA, za Octave)

```
% vhodne spremenljivke: (p) – obrestna mera/kapitalizacijsko obdobje, …
% …(n) – število obrokov, (D0) – glavnica

% metoda računanja členov neposredno iz vhodnih podatkov
seštevek_ObrestovalnihFaktorjev = 0
za zapŠt_ObrestovalnegaFaktorja = 1 do n
   trenutniObrestovalniFaktor = (1 + p)^(zapŠt_ObrestovalnegaFaktorja – 1)
   seštevek_ObrestovalnihFaktorjev = seštevek_ObrestovalnihFaktorjev +
                                     trenutniObrestovalniFaktor
konecza

% spremenljivka se vstavi v formulo, pravilno se prikaže tudi denarni tok
anuiteta_exponent = –(D0 × p + D0 / (seštevek_ObrestovalnihFaktorjev))
```

```
Option Explicit

' metoda računanja členov neposredno iz vhodnih podatkov
Function payment_exponent(p As Double, n As Double, D0 As Double) As Double
   Dim cum_temp_r As Double, i As Double, temp_r As Double

   cum_temp_r = 0
   For i = 1 To n
      temp_r = (1 + p)^(i – 1)
      cum_temp_r = cum_temp_r + temp_r
   Next i

   payment_exponent = –(D0 × p + D0 / cum_temp_r)
End Function
```

```
# metoda računanja členov neposredno iz vhodnih podatkov
function retval_payment_exponent = payment_exponent(p, n, D0)
   format long

   cum_temp_r = 0;
   for i = 1 : n
      temp_r = (1 + p)^(i – 1);
      cum_temp_r = cum_temp_r + temp_r;
   endfor

   retval_payment_exponent = –(D0 * p + D0 / cum_temp_r);
endfunction
```

Priloga 2: Računanje anuitete – vsota členov geometričnega zaporedja rekurzivno (opisni algoritem, za Excel-VBA, za Octave)

```
% vhodne spremenljivke: (p) – obrestna mera/kapitalizacijsko obdobje, …
% …(n) – število obrokov, (D0) – glavnica

% metoda dodajanja glede na predhodni seštevek razdolžnin
seštevek_ObrestovalnihFaktorjev = 0
za zapŠt_ObrestovalnegaFaktorja = 1 do n
   trenutniObrestovalniFaktor = seštevek_ObrestovalnihFaktorjev × p + 1
   seštevek_ObrestovalnihFaktorjev = seštevek_ObrestovalnihFaktorjev +
                                     trenutniObrestovalniFaktor
konecza

% spremenljivka se vstavi v formulo, pravilno se prikaže tudi denarni tok
anuiteta_recursive = –(D0 × p + D0 / (seštevek_ObrestovalnihFaktorjev))
```

```
Option Explicit

' metoda dodajanja glede na predhodni seštevek razdolžnin
Function payment_recursive(p As Double, n As Double, D0 As Double) As Double
   Dim cum_temp_r As Double, i As Double, temp_r As Double

   cum_temp_r = 0
   For i = 1 To n
      temp_r = cum_temp_r * p + 1
      cum_temp_r = cum_temp_r + temp_r
   Next i

   payment_recursive = –(D0 × p + D0 / cum_temp_r)
End Function
```

```
# metoda dodajanja glede na predhodni seštevek razdolžnin
function retval_payment_recursive = payment_recursive(p, n, D0)
   format long

   cum_temp_r = 0;
   for i = 1 : n
      temp_r = cum_temp_r * p + 1;
      cum_temp_r = cum_temp_r + temp_r;
   endfor

   retval_payment_recursive = –(D0 * p + D0 / cum_temp_r);
endfunction
```

Priloga 3A: Računanje anuitete – vsota členov geometričnega zaporedja, kjer ti členi medsebojno ne tvorijo geometričnega zaporedja (opisni algoritem)

```
% vhodne spremenljivke: (p) - obrestna mera/kapitalizacijsko obdobje, …
% …(n) - število obrokov, (D0) - glavnica

% metoda, pri kateri členi vsote medsebojno ne tvorijo geometričnega zaporedja
seštevek_Zmnožkov_Členov = 0
zmnožek_Posameznega_Člena= 0
za zapŠt_ObrestovalnegaFaktorja = 1 do n
    če zapŠt_ObrestovalnegaFaktorja==1
        zmnožek_Posameznega_Člena = n
        seštevek_Zmnožkov_Členov = zmnožek_Posameznega_Člena
    sicer
        zmnožek_Posameznega_Člena = zmnožek_Posameznega_Člena ×
                                    (n – zapŠt_ObrestovalnegaFaktorja + 1) × p
        seštevek_Zmnožkov_Členov = seštevek_Zmnožkov_Členov +
                                   zmnožek_Posameznega_Člena
    konecče
konecza

% spremenljivka se vstavi v formulo, pravilno se prikaže tudi denarni tok
anuiteta_errAnalysis = –(D0 × p + D0 / (seštevek_Zmnožkov_Členov))
```

Priloga 3B: Računanje anuitete – vsota členov geometričnega zaporedja, kjer ti členi medsebojno ne tvorijo geometričnega zaporedja (za Excel-VBA)

```
1  Option Explicit
2
3  ' metoda, pri kateri členi vsote medsebojno ne tvorijo geometričnega zaporedja
4  Function payment_errAnalysis(p As Double, n As Double, D0 As Double) As Double
5      Dim cum_product_i As Double, product_i As Double, i As Double
6
7      cum_product_i = 0
8      product_i = 0
9      For i = 1 To n
10         If i = 1 Then
11             product_i = n
12             cum_product_i = product_i
13         Else
14             product_i = product_i / i * (n - i + 1) * p
15             cum_product_i = cum_product_i + product_i
16         End If
17     Next i
18
19     ' spremenljivka se vstavi v formulo, pravilno se prikaže tudi denarni tok
20     payment_errAnalysis = –(D0 * p + D0 / (cum_product_i))
21 End Function
```

Priloga 3C: Računanje anuitete - vsota členov geometričnega zaporedja, kjer ti členi medsebojno ne tvorijo geometričnega zaporedja (za Octave)

```
1  #  metoda, pri kateri členi vsote medsebojno ne tvorijo geometričnega zaporedja
2  function retval_payment_errAnalysis = payment_errAnalysis(p, n, D0)
3     format long
4
5     cum_product_i = 0;
6     product_i = 0;
7     for i = 1 : n
8        if i == 1
9           product_i = n;
10          cum_product_i = product_i;
11       else
12          product_i = product_i / i * (n - i + 1) * p;
13          cum_product_i = cum_product_i + product_i;
14       endif
15    endfor
16
17    #  spremenljivka se vstavi v formulo, pravilno se prikaže tudi denarni tok
18    retval_payment_errAnalysis  = –(D0 * p + D0 / (cum_product_i));
19 endfunction
```

Priloga 4A: Računanje anuitete – vsota skupin členov geometričnega zaporedja rekurzivno (opisni algoritem)

```
1  % vhodne spremenljivke: (p) – obrestna mera/kapitalizacijsko obdobje, …
2  % … (n) – število obrokov, (D0) – glavnica
3  % PRIMER IZRAČUNA ZA 40 OBROKOV = 6 × 6 + 4 (= 15 iteracij v treh zankah)
4
5  % izračun kvadratnega korena = 6
6  kvadratniKoren_ŠtObrokov_zaokroženNavzdolNaCeloŠtevilo =
7                                          odvzemiDaBoCeloŠtevilo(kvadratniKoren(n))
8
9  % seštevanje 1. skupine OBRESTOVALNIH FAKTORJEV: 1., 7., 13., 19., 25. in 31. …
10 % … (=skupaj 6 iteracij)
11 seštevek_ObrestovalnihFaktorjev = 0
12 za zapŠt_ObrestovalnegaFaktorja = 1 korak
13                    kvadratniKoren_ŠtObrokov_zaokroženNavzdolNaCeloŠtevilo do
14                    kvadratniKoren_ŠtObrokov_zaokroženNavzdolNaCeloŠtevilo^2
15
16    trenutniObrestovalniFaktor = (1 + p)^(zapŠt_ObrestovalnegaFaktorja – 1)
17    seštevek_ObrestovalnihFaktorjev = seštevek_ObrestovalnihFaktorjev +
18                                          trenutniObrestovalniFaktor
19 konecza
20
21 % računanje 2., 3., 4., 5. in 6. skupine OBR. FAKTORJEV (skupaj 6 – 1 = 5 iteracij)
22 rezultatPrveZankeKotOsnovaZaDodajanje = seštevek_ObrestovalnihFaktorjev
23 štPreostalihIteracij =
24                    kvadratniKoren_ŠtObrokov_zaokroženNavzdolNaCeloŠtevilo – 1
25 medtemKo štPreostalihIteracij > 0
26    trenutniObrestovalniFaktor = seštevek_ObrestovalnihFaktorjev × p +
27                                    rezultatPrveZankeKotOsnovaZaDodajanje
28    seštevek_ObrestovalnihFaktorjev = seštevek_ObrestovalnihFaktorjev +
29                                          trenutniObrestovalniFaktor
30
31    štPreostalihIteracij = štPreostalihIteracij – 1
32 konecmedtemKo
33
34 % prištejejo se še OBR. FAKTORJI iz ostanka začetnega korenjenja (4 iteracije)
35 medtemKo n >= zapŠt_ObrestovalnegaFaktorja
36    trenutniObrestovalniFaktor = (1 + p)^(zapŠt_ObrestovalnegaFaktorja – 1)
37    seštevek_ObrestovalnihFaktorjev = seštevek_ObrestovalnihFaktorjev +
38                                          trenutniObrestovalniFaktor
39
40 zapŠt_ObrestovalnegaFaktorja = zapŠt_ObrestovalnegaFaktorja + 1
41 konecmedtemKo
42
43 % spremenljivka se vstavi v formulo, pravilno se prikaže tudi denarni tok
44 anuiteta_sqrt = –(D0 × p + D0 / (seštevek_ObrestovalnihFaktorjev))
```

Priloga 4B: Računanje anuitete – vsota skupin členov geometričnega zaporedja rekurzivno (za Excel-VBA)

```
Option Explicit

Function payment_sqrt(p As Double, n As Double, D0 As Double) As Double
    Dim sqrtOf_numOfPeriods_roundedDownToInteger As Long
    Dim cum_temp_r As Double, i As Double, temp_r As Double
    Dim resultFromFirstLoop As Double, remainingIterations As Long

    ' PRIMER IZRAČUNA ZA 40 OBROKOV = 6 * 6 + 4 (= skupaj 15 iteracij)

    ' izračun kvadratnega korena = 6
    sqrtOf_numOfPeriods_roundedDownToInteger = Int(Sqr(n))

    ' seštevanje 1. skupine OBR. FAKTORJEV: 1., 7., 13., 19., 25. in 31. ...
    ' ... (=skupaj 6 iteracij)
    cum_temp_r = 0
    For i = 1 To sqrtOf_numOfPeriods_roundedDownToInteger ^ 2 _
                Step sqrtOf_numOfPeriods_roundedDownToInteger
        temp_r = (1 + p) ^ (i – 1)
        cum_temp_r = cum_temp_r + temp_r
    Next i

    ' računanje 2., 3., 4., 5. in 6. skupine OBR. FAKTORJEV (skupaj 6 – 1 = 5 iteracij)
    resultFromFirstLoop = cum_temp_r
    remainingIterations = sqrtOf_numOfPeriods_roundedDownToInteger – 1
    Do While remainingIterations > 0
        temp_r = cum_temp_r * p + resultFromFirstLoop
        cum_temp_r = cum_temp_r + temp_r

        remainingIterations = remainingIterations – 1
    Loop

    ' prištejejo se še OBR. FAKTORJI iz ostanka začetnega korenjenja (4 iteracije)
    Do While n >= i
        temp_r = (1 + p) ^ (i – 1)
        cum_temp_r = cum_temp_r + temp_r

        i = i + 1
    Loop

    ' spremenljivka se vstavi v formulo, pravilno se prikaže tudi denarni tok
    payment_sqrt = –(D0 * p + D0 / (cum_temp_r))
End Function
```

Priloga 4C: Računanje anuitete – vsota skupin členov geometričnega zaporedja rekurzivno (za Octave)

```
function retval_payment_sqrt = payment_sqrt(p, n, D0)
   format long

   #  PRIMER IZRAČUNA ZA 40 OBROKOV = 6 * 6 + 4 (= skupaj 15 iteracij)

   #  izračun kvadratnega korena = 6
   sqrtOf_numOfPeriods_roundedDownToInteger = floor(sqrt(n));

   #  seštevanje 1. skupine OBR. FAKTORJEV: 1., 7., 13., 19., 25. in 31. …
   #  … (=skupaj 6 iteracij)
   cum_temp_r = 0;
   for i = 1 : sqrtOf_numOfPeriods_roundedDownToInteger : …
               sqrtOf_numOfPeriods_roundedDownToInteger ^ 2
      temp_r = (1 + p) ^ (i – 1);
      cum_temp_r = cum_temp_r + temp_r;
   endfor

   #  računanje 2., 3., 4., 5. in 6. skupine OBR. FAKTORJEV (skupaj 6–1 = 5 iteracij)
   resultFromFirstLoop = cum_temp_r;
   remainingIterations = sqrtOf_numOfPeriods_roundedDownToInteger – 1;
   while remainingIterations > 0
      temp_r = cum_temp_r * p + resultFromFirstLoop;
      cum_temp_r = cum_temp_r + temp_r;

      remainingIterations = remainingIterations – 1;
   endwhile

   ##  VBA-jeva "For" zanka po zaključku vseh instrukcij še enkrat poveča …
   ##  … vrednost števca , Octave-jeva pa ne. Zato sledeč dodatek
   i = i + sqrtOf_numOfPeriods_roundedDownToInteger;

   #  prištejejo se še OBR. FAKTORJI iz ostanka začetnega korenjenja (4 iteracije)
   while n >= i
      temp_r = (1 + p) ^ (i – 1);
      cum_temp_r = cum_temp_r + temp_r;

      i = i + 1;
   endwhile

   #  spremenljivka se vstavi v formulo, pravilno se prikaže tudi denarni tok
   retval_payment_sqrt = –(D0 * p + D0 / (cum_temp_r));
endfunction
```

Priloga 5A: Računanje anuitete – vsota geometričnega zaporedja iz skupin obrestnih mer v dvojiških korakih (opisni algoritem)

```
% vhodne spremenljivke: (p) – obrestna mera/kapitalizacijsko obdobje, …
% …(n) – število obrokov, (D0) – glavnica

% ustvari se prazen vektor
dolžinaVektorja = odvzemiDaBoCeloŠtevilo(dvojiškiLogaritem(n)) + 1
vektor_ObrestneMere =
                        ustvariVektor_dolžineVrednostiArgumenta(dolžinaVektorja)

% vektor se zapolni z obrestnimi merami
za trenutniElementVVektorju = 1 do dolžinaVektorja
   če trenutniElementVVektorju == 1
      vektor_ObrestneMere(trenutniElementVVektorju) = p
   sicer
      vektor_ObrestneMere(trenutniElementVVektorju) =
                        vektor_ObrestneMere(trenutniElementVVektorju – 1)^2 +
                        vektor_ObrestneMere(trenutniElementVVektorju – 1) × 2
   konecče
konecza

% računanje spremenljivke (1 + p)^n – 1 iz določenih elementov vektorja
trenutnoŠteviloObrokov = n
seštevek_ObrestnihMer = 0
za trenutniElementVVektorju = 1 do dolžinaVektorja
   če jeNeparno(trenutnoŠteviloObrokov)
      seštevek_ObrestnihMer =
                           seštevek_ObrestnihMer ×
                           vektor_ObrestneMere(trenutniElementVVektorju) +
                           seštevek_ObrestnihMer +
                           vektor_ObrestneMere(trenutniElementVVektorju)
   konecče
   trenutnoŠteviloObrokov =
                        odvzemiDaBoCeloŠtevilo(trenutnoŠteviloObrokov / 2)
konecza

% spremenljivka se vstavi v formulo, pravilno se prikaže tudi denarni tok
anuiteta_log2 = –(D0 × p + D0 / (seštevek_ObrestnihMer / p))
```

Priloga 5B: Računanje anuitete – vsota geometričnega zaporedja iz skupin obrestnih mer v dvojiških korakih (za Excel-VBA)

```
Option Explicit

Function payment_log2(p As Double, n As Double, D0 As Double) As Double
    Dim cum_temp_p As Double, i As Integer, lengthOfVector As Integer
    Dim temp_n As Double, temp_p As Double, temp_p_InVector() As Double

    ' ustvari se prazen vektor
    lengthOfVector = Int(Application.Log(n, 2)) + 1
    ReDim temp_p_InVector(1 To lengthOfVector)

    ' vektor se zapolni z izračuni OBRESTNIH MER za obdobja po log2 korakih
    For i = 1 To lengthOfVector
        If i = 1 Then
            temp_p_InVector(i) = p
        Else
            temp_p_InVector(i) = temp_p_InVector(i – 1) ^ 2 + _
                                 temp_p_InVector(i – 1) * 2
        End If
    Next i

    ' računanje skupne OBRESTNE MERE za izbrano obdobje, …
    ' … vendar samo iz določenih elementov vektorja
    temp_n = n
    cum_temp_p = 0
    For i = 1 To lengthOfVector
        If Application.IsOdd(temp_n) Then
            cum_temp_p = cum_temp_p * temp_p_InVector(i) + _
                         cum_temp_p + temp_p_InVector(i)
        End If
        temp_n = Int(temp_n / 2)
    Next i

    ' spremenljivka se vstavi v formulo, pravilno se prikaže tudi denarni tok
    payment_log2 = –(D0* p + D0 / (cum_temp_p / p))
End Function
```

Priloga 5C: Računanje anuitete – vsota geometričnega zaporedja iz skupin obrestnih mer v dvojiških korakih (za Octave)

```
function retval_payment_log2 = payment_log2(p, n, D0)
   format long

   #  ustvari se prazen vektor
   lengthOfVector = floor(log2(n)) + 1;
   temp_p_InVector = cell(lengthOfVector, 1);

   #  vektor se zapolni z izračuni OBRESTNIH MER za obdobja po log2 korakih
   for i = 1 : lengthOfVector
      if i == 1
         temp_p_InVector{i, 1} = p;
      else
         temp_p_InVector{i, 1} = temp_p_InVector{i – 1, 1} ^ 2 + ...
                                 temp_p_InVector{i – 1, 1} * 2;
      endif
   endfor

   #  računanje skupne OBRESTNE MERE za izbrano obdobje, …
   #  … vendar samo iz določenih elementov vektorja
   temp_n = n;
   cum_temp_p = 0;
   for i = 1 : lengthOfVector
      if mod(temp_n, 2) == 1
         cum_temp_p = cum_temp_p * temp_p_InVector{i, 1} + ...
                      cum_temp_p + temp_p_InVector{i, 1};
      endif
      temp_n = floor(temp_n / 2);
   endfor

   #  spremenljivka se vstavi v formulo, pravilno se prikaže tudi denarni tok
   retval_payment_log2 = –(D0 * p + D0 / (cum_temp_p / p));
endfunction
```

Priloga 6A: Odštevanje v aritmetiki s premično piko (opisni algoritem)

```
1  %  vhodni spremenljivki: zmanjševanec in odštevanec
2
3  %  odštevanje števil različnih predznakov (== SEŠTEVANJE)
4  če nimataEnakihPredznakov(zmanjševanec, odštevanec)
5     odštevanje = zmanjševanec – odštevanec
6
7  %  odštevanje števil enakih predznakov
8  sicer
9     najvišjeŠteviloPomembnihŠtevk_priUporabljeniNatančnosti = 15
10
11    najnižjaAbsolutnaVrednost = najnižjaVrednost(
12                                   absolutno(zmanjševanec), absolutno(odštevanec))
13    številoŠtevkVCelemDelu = zaokroži NavzdolNaCeloMesto(
14                                (desetiškiLogaritem(najnižjaAbsolutnaVrednost))) + 1
15    številoŠtevkVDecimalnemDelu =
16                        najvišjeŠteviloPomembnihŠtevk_priUporabljeniNatančnosti –
17                        številoŠtevkVCelemDelu
18    odštevanje = zaokroži Razliko_na_številoŠtevkVDecimalnemDelu(
19                         zmanjševanec – odštevanec, številoŠtevkVDecimalnemDelu)
20 konecče
```

Algoritem v primeru 100 % relativne napake ne uspe popraviti le-te. Priloga 7 prikazuje morebitne razloge zakaj algoritem ne deluje v vseh primerih za programska jezika VBA in R. Vendar pa deluje za Excel, kljub temu, da pri slednjem uporabimo VBA programski jezik. Pri VBA lahko nastopi 100 % relativna napaka pri odštevanju namreč šele takrat, ko je eno izmed števil zapisano s 16-mestno natančnostjo. Ker pa Excel dovoljuje vnose le na 15-mest natančno, se ta primer ne zgodi.

Priloga 6B: Odštevanje v aritmetiki s premično piko (za Excel-VBA)

```
1  Option Explicit
2
3  Function subtraction(minuend As Double, subtrahend As Double) As Double
4      Dim minAbsValue As Double
5      Dim numOfSignifDigitsInIntegerPartOfNum As Integer
6      Dim numOfSignifDigitsInDecimalPartOfNum As Integer
7      Const maxSignifDigitsInNum_forUsedPrecision As Byte = 15
8
9      ' odštevanje števil različnih predznakov (== SEŠTEVANJE)
10     If Sgn(minuend) * Sgn(subtrahend) < 1 Then
11         subtraction = minuend – subtrahend
12
13     ' odštevanje števil enakih predznakov
14     Else
15         minAbsValue = Application.Min(Abs(minuend), Abs(subtrahend))
16         numOfSignifDigitsInIntegerPartOfNum = _
17                                               Int(Application.Log10(minAbsValue)) + 1
18         numOfSignifDigitsInDecimalPartOfNum = _
19                                           maxSignifDigitsInNum_forUsedPrecision – _
20                                           numOfSignifDigitsInIntegerPartOfNum
21         subtraction = Application.Round(minuend – subtrahend, _
22                                           numOfSignifDigitsInDecimalPartOfNum)
23     End If
24 End Function
```

Priloga 6C: Odštevanje v aritmetiki s premično piko (za Octave)

```
function retval_subtraction = subtraction(minuend, subtrahend)
   format long

   #  odštevanje števil različnih predznakov (== SEŠTEVANJE)
   if sign(minuend) * sign(subtrahend) < 1
      retval_subtraction = minuend – subtrahend;

   #  odštevanje števil enakih predznakov
   else
      maxSignifDigitsInNum_forUsedPrecision = 15;

      minAbsValue = min(abs(minuend), abs(subtrahend));
      numOfSignifDigitsInIntegerPartOfNum = floor (log10(minAbsValue)) + 1;
      numOfSignifDigitsInDecimalPartOfNum = ...
                                          maxSignifDigitsInNum_forUsedPrecision – ...
                                          numOfSignifDigitsInIntegerPartOfNum;
      retval_subtraction = roundb((minuend – subtrahend) * ...
                                  (10 ^ numOfSignifDigitsInDecimalPartOfNum)) / ...
                                  (10 ^ numOfSignifDigitsInDecimalPartOfNum);
   endif
endfunction
```

Priloga 7: Parametri računske aritmetike nekaterih programskih paketov

	Aritmetika po standardu IEEE	Vrednosti ob prekoračitvi (overflow)	Vrednost ob podkoračitvi (underflow) / (denormalizirana števila)	Osnovna zaokrožitvena napaka[15] (pri dvojni natančnosti)	...
Excel	Ne	#ŠTEV! (prekine računanje)	0 / (Ne)	$u = 2^{-48} \approx 3.5 \cdot 10^{-15}$	...
VBA	Ne	#INF (prekine računanje)	0 / (Da)	$u = 2^{-53} \approx 1.1 \cdot 10^{-16}$	...
Octave	Da[1]	$\infty, -\infty$	0 / (Da)	$u = 2^{-53} \approx 1.1 \cdot 10^{-16}$	...
R	Da[1]	$\infty, -\infty$	0 / (Da)	$u = 2^{-53} \approx 1.1 \cdot 10^{-16}$	...

	...	Število pomembnih števk	Število bitov v mantisi za prikaz poljubnega števila, glede na uporabljeno število pomembnih števk: razpoložljivih / teoretično potrebnih	Najvišja relativna napaka pri odštevanju
Excel	...	15	48 / 50	100%
VBA	...	16	53 / 54	100%
Octave	...	15	53 / 50	≈ 9,38%
R	...	vnosi = 16 rezultat = 22	53 / 54	100%

Opombe: Primer najvišje relativne napake pri odštevanju – za Excel: 99999999999999,8-99999999999999,7=0; za VBA in R: 999999999999999.8-999999999999999.7=0; za Octave: 99999999999999.9-99999999999999.8=0.109375.

[15] Za izračun smo uporabili algoritem po Bohtetu (1995, str. 71).