

Technisch-Wetenschappelijke Software Debuggen, profilen & timen in Matlab

Dirk Nuyens

Academiejaar 2006-2007

1 Opgave voor tijdens de oefenzitting

Experimenteer in de Matlab IDE met de debugger en profiler op code voor het prijzen van een Europese put optie. We gebruiken hiervoor Matlab scripts van Desmond Higham:

<http://www.maths.strath.ac.uk/~aas96106/algfiles.html>

In deze Matlab scripts wordt er voornamelijk gebruik gemaakt van vectorisatie om de code te versnellen. Maar eerst geven we een klein beetje achtergrond over wat (en hoe) er juist berekend wordt.

Een Europese put optie is een financieel product waarbij de houder van de optie op een vooraf vastgelegd tijdstip T de mogelijkheid heeft om een bepaald onderliggend product (bijvoorbeeld aandelen) te verkopen tegen een vastgelegde prijs K . Indien het onderliggende product op moment T minder waard is dan K dan zal de houder zijn optie uitoefenen en het verschil in prijs opstrijken. In het andere geval laat de houder zijn optie gewoon verstrijken.

De prijs van het onderliggende product stellen we voor met $S(t)$, van stock price. We stellen nu die prijs S voor op discrete tijdstippen en veronderstellen dat deze prijs van tijdstip t naar $t + \delta t$ vermenigvuldigd kan worden met vaste $u > 1$ en $d < 1$ voor opwaartse en neerwaartse bewegingen van S , voor een vaste δt . De kans voor een opwaartse beweging is p , die voor de neerwaartse is $1 - p$.

We nemen M stappen om tot aan tijdstip T te geraken zodat $M\delta t = T$. De waarde van de optie op tijdstip $t = 0$ stellen we eenvoudig voor met S .

1. Ga na dat je een binaire boom krijgt waarvan de takken terug samenkomen: vertrekkende vanaf tijdstip $t = 0$ met op- en neerwaartse prijsbewegingen van het onderliggende product. Je merkt nu dat op het *ide* tijdstip, $t = i\delta t$, we $i + 1$ mogelijke prijzen van het onderliggende product hebben:

$$S_k^{(i)} = d^{i-k} u^k S, \quad 0 \leq k \leq i.$$

Dus op tijdstip T zijn er $M + 1$ mogelijke prijzen (voor $M + 1$ mogelijke waarden van k) van het onderliggende product en de bijhorende $M + 1$ waarden van de optie zijn dan

$$V_k^{(M)} = \max(K - S_k^{(M)}, 0).$$

De bijhorende kans voor deze toestand volgt uit de boom.

We willen nu weten wat de waarde van de optie is op tijdstip $t = 0$, dat is $V = V_0^{(0)}$. De waarde op het *ide* tijdstip kunnen we bepalen met behulp van een gewogen gemiddelde van de prijzen op tijdstip $i + 1$:

$$V_k^{(i)} = e^{-r\delta t} \left(pV_{k+1}^{(i+1)} + (1-p)V_k^{(i+1)} \right).$$

(Hierbij is r de “risicoloze” interestvoet en actualiseert de vermenigvuldiging met $e^{-r\delta t}$ de waarde van het geld.)

- Bekijk het script `euro1` en overtuig jezelf ervan dat dit script inderdaad de bovenstaande formules implementeert. Hiervoor gebruik je de Matlab debugger. Je kan eenvoudigweg breakpoints zetten in de Matlab editor. Experimenteer met de mogelijkheden van de debugger. Hoe zit het met de indices in de code in vergelijking met die in de formules?
- Plaats een conditioneel breakpoint op de eerste lus dat stopt wanneer de laatste iteratie begint. (Waar zet je dit breakpoint?) Bekijk dan de vector `W` in de array editor van Matlab terwijl je door de binnenste lus stapt. Merk ook op dat je Matlab command line veranderd is in `K>>`, ook hier kan je commando's intypen.
- Maak nu functies van de scripts, gegeven de initiële stock price S , de strike price K , de einddatum T , de risicoloze interestvoet r , de volatiliteit van het aandeel σ en het aantal discretisatiestappen M . Doe dit voor de scripts `euro1`, `euro2`, ..., `euro5`. Bekijk grondig de aanpassingen van script tot script.
- Gebruik de Matlab profiler op je functies. Kan je besluiten welke functie het snelste is? Wat is de resolutie van deze tijdmetingen?
- Schrijf een functie `funtimer` met de volgende interface:

```
function ts = funtimer(n1, n2, fun, varargin)
% inputs:
%   n1          number of timings (outer loop)
%   n2          number of calls to time at once (inner loop)
%   fun         a function handle
%   varargin    inputs to fun
% outputs:
%   ts          vector of n1 execution times in seconds of fun
%              averaged over n2 runs
```

Gebruik deze functie voor het opmeten van de uitvoertijden van je functies. (Het doorgeven van een functie `bar` doe je met de syntax `@bar`; oproepen doe je als `bar(varargin{:})`.) Experimenteer met verschillende waarden voor `n1` en `n2`. Plot de meetwaarden voor de verschillende scripts (met bijvoorbeeld `n1 = 100`) op één figuur. Herhaal de metingen maar wissel eens van desktop of versleep een venster tijdens de metingen. Is het gemiddelde van zo 100 metingen nu nog een goede maat voor de uitvoertijd? (Merk op dat je dit effect in feite altijd hebt: afhankelijk van de resolutie waarop je tijdmetingen doet kan het afhandelen van b.v. een interrupt je tijdmetingen al beïnvloeden.)

2 Kleine opgave voor thuis

- Schrijf een Matlab functie die de resolutie van de timer(s!) in Matlab tracht te achterhalen.
- De Matlab interpreter wordt alsmaar "slimmer" en sneller (o.m. door de JIT-accelerator). Tracht te achterhalen of het uitmaakt om de constante bewerkingen uit een lus te verwijderen. Bijvoorbeeld door $\sqrt{2}$ op voorhand uit te rekenen in:

```
for i=1:n
    a(i) = sqrt(2) * a(i);
end;
```

Doe dezelfde test eens in Octave. Herhaal ook de tijdmetingen van hierboven in Octave.

- Maak een bruikbare grafiek om de tijdmetingen van je verschillende functies voor de Europese put optie voor te stellen (met $M = 256$ zoals in de originele scripts). Maak ook een grafiek in functie van het aantal discretisatiestappen M . (Op beide grafieken moeten we kunnen vergelijken tussen de verschillende methoden.) Is de keuze van M belangrijk?

Stuur de code alsook een PDF met de figuren en wat commentaar naar Dirk Nuyens. Geef aan hoeveel tijd je besteedde aan de opdracht voor thuis.