BENCHOP
The BENCHmarking project in Option Pricing.
Version 2.0

August 27, 2015

1 Purpose and aims

The purpose of BENCHOP is to provide a set of benchmark problems that can be used for comparing methods and evaluating new methods. Implementations in MATLAB of standard methods to compare against are also provided. All MATLAB-implementations will be available to everybody at `www.it.uu.se/research/project/compfin/benchop`. Any use of the code in the future is expected to be adequatly cited.

The aim of BENCHOP is to serve as a take off for future development of methods in option pricing. We expect that future papers in the field will compare method performances with the methods and codes in BENCHOP. Thanks to this, we believe that we will contribute to a more uniform comparison and understanding of different methods' pros and cons in the future.

2 Formulation of test problems

2.1 Problem 1 - Black-Scholes model for one underlying asset with constant volatility; European call option with hedging parameters, American put option and Barrier call up-and-out option

- SDE-setting:
$$dS = rSdt + \sigma SdW.$$
- PDE-setting:
$$\frac{\partial u}{\partial t} + \frac{1}{2}\sigma^2 s^2 \frac{\partial^2 u}{\partial s^2} + rs\frac{\partial u}{\partial s} - ru = 0.$$

- Parameters:
  - I $\sigma = 0.15,\ r = 0.03,\ T = 1.0,$ and $K = 100.$
  - II $\sigma = 0.01,\ r = 0.10,\ T = 0.25,$ and $K = 100.$
- Problem-specific data:
  - a. European call: $\phi(s,T) = \max(s-K,0)$.
  - b. American put: $\phi(s,T) = \max(K-s,0)$, $u(s,t) \geq \phi(s)$, $0 \leq t \leq T$.
  - c. Barrier call up-and-out: $\phi(s,T) = \max(s-K,0)$, $0 \leq s < B$; $\phi(s,T) = 0$, $s \geq B$, $B = 1.25K$.
- Output:
  - a. u and hedging parameters $\Delta = \frac{\partial u}{\partial s}$, $\Gamma = \frac{\partial^2 u}{\partial s^2}$ and $\mathcal{V} = \frac{\partial u}{\partial \sigma}$ at $t = 0$ and $s = 90,\ 100,\ 110$ for parameter set I and $s = 97,\ 98$ and 99 for parameter set II.
  - b. u at $t = 0$ and $s = 90,\ 100,\ 110$ for parameter set I and $s = 97,\ 98$ and 99 for parameter set II.
  - c. u at $t = 0$ and $s = 90,\ 100,\ 110$ for parameter set I and $s = 97,\ 98$ and 99 for parameter set II.
- Benchmark: the computational time to obtain the output with relative error $< 10^{-4}$ in each point. For a. this means four different values of the computational time, one for u, one for Δ, one for Γ and one for $\mathcal{V}$.
- Graph: for the resulting parameter setting the solution u (for a. also Δ, Γ, $\mathcal{V}$) in [60:1:160] (for c. in [60:1:125]) should be computed. *The computed u (for a. also Δ, Γ, $\mathcal{V}$) − the "exact" solution u (for a. also Δ, Γ, $\mathcal{V}$) =the error in your computed solution* will be used for figures in the article.
- You shall submit twelve different MATLAB-codes (eight for a. and two each for b. and c.). The MATLAB-codes shall compute u (for a. also Δ, Γ, $\mathcal{V}$) in the measurement points (90, 100, 110 and 97, 98, 99 respectively) with a relative error less than 10^{-4}. The code shall have the following headings:
  - 1a European call parameter set I

```
[U]=BSeuCallUI_xxx(S,K,T,r,sig)

[Delta]=BSeuCallDeltaI_xxx(S,K,T,r,sig)

[Gamma]=BSeuCallGammaI_xxx(S,K,T,r,sig)
```

`[Vega]=BSeuCallVegaI_xxx(S,K,T,r,sig)`

- 1a European call parameter set II

  `[U]=BSeuCallUII_xxx(S,K,T,r,sig)`

  `[Delta]=BSeuCallDeltaII_xxx(S,K,T,r,sig)`

  `[Gamma]=BSeuCallGammaII_xxx(S,K,T,r,sig)`

  `[Vega]=BSeuCallVegaII_xxx(S,K,T,r,sig)`

- 1b American put parameter set I

  `[U]=BSamPutUI_xxx(S,K,T,r,sig)`

- 1b American put parameter set II

  `[U]=BSamPutUII_xxx(S,K,T,r,sig)`

- 1c European up and out parameter set I

  `[U]=BSupoutCallI_xxx(S,K,T,r,sig,B)`

- 1c European up and out parameter set II

  `[U]=BSupoutCallII_xxx(S,K,T,r,sig,B)`

2.2 Problem 2 - Black-Scholes model for one underlying asset with constant volatility and discrete dividends; European call option and American call option

- SDE-setting:
$$dS = rSdt + \sigma SdW - \delta(t-\tau)D \cdot S.$$
- At time $\tau = \alpha \cdot T$ a dividend of $D \cdot S$ is paid, $\alpha = 0.8$ and $D = 0.03$.

$$\frac{\partial u}{\partial t} + \frac{1}{2}\sigma^2 s^2 \frac{\partial^2 u}{\partial s^2} + rs\frac{\partial u}{\partial s} - ru = 0 \quad , \quad \text{at time } \tau \; u(s,\tau^-) = u(s(1-D),\tau^+).$$

- Parameters:
  - $\sigma = 0.15,\ r = 0.03,\ T = 0.5,$ and $K = 100.$
- Problem-specific data:
  - a. European call: $\phi(s,T) = \max(s-K,0)$.
  - b. American call: $\phi(s,T) = \max(s-K,0)$, $u(s,t) \geq \phi(s)$, $0 \leq t \leq T$.
- Output: u at $t = 0$ and $s = 90$, 100, 110.
- Benchmark: the computational time to obtain the output with relative error $< 10^{-4}$ in each point.
- Graph: for the resulting parameter setting the solution u in [60:1:160] should be computed. *The computed u−the "exact" solution u* will be used for figures in the article.
- You shall submit two different MATLAB-codes (one for a. and one for b.). The MATLAB-codes shall compute u in the measurement points (90, 100, 110) with a relative error less than 10^{-4}. The code shall have the following headings:
  - 2a European call
    `[U]=BSeuCallDD_xxx(S,K,T,r,sig,D,alpha)`
  - 2b American call
    `[U]=BSamCallDD_xxx(S,K,T,r,sig,D,alpha)`

2.3 Problem 3 - Black-Scholes model for one underlying asset with local volatility; European call option

- SDE-setting:
$$dS = rSdt + \sigma SdW.$$
- PDE-setting:
$$\frac{\partial u}{\partial t} + \frac{1}{2}\sigma^2 s^2 \frac{\partial^2 u}{\partial s^2} + rs\frac{\partial u}{\partial s} - ru = 0.$$
- Pay-off function: $\phi(s,T) = \max(s-K,0)$.
- Local volatility functions:
  - I $\sigma(s,t) = 0.15 + 0.15(0.5+2t)\frac{(s/100-1.2)^2}{(s/100)^2+1.44}$, $r = 0.03,\ T = 1.0,$ and $K = 100.$

  - II The local volatility $\sigma(s,t)$ is described in Appendix A, $r = 0.03$, $T = 0.5$, and $K = 100$. The computation of the volatility function is provided in `localvol.m`. Note that the volatility surface is different for different values of today's price s_0 and a new volatility surface needs to be computed for each measurement point (=today's price) $s_0 = 90$, 100, 110. This means that the option price in each s_0 has to be computed using a different volatility function.

- Output: u at $t = 0$ and $s = 90$, 100, 110.

- Benchmark: the computational time to obtain the output with relative error $< 10^{-4}$ in each point.

- You shall submit two different MATLAB-codes (one for local volatility and parameters I and one for local volatility and parameters II). The MATLAB-codes shall compute u in the measurement points (90, 100, 110) with a relative error less than 10^{-4}. The code shall have the following headings:

  - Parameter set I
    `[U]=BSeuLocVolI_xxx(S,K,T,r,@sig)`

  - Parameter set II
    `[U]=BSeuLocVolII_xxx(S,K,T,r,@sig)`

2.4 Problem 4 - Heston model for one underlying asset; European call option

- SDE-setting:

$$\begin{aligned} dS &= rSdt + \sqrt{V}SdW_1, \\ dV &= \kappa(\theta - V)dt + \sigma\sqrt{V}dW_2. \end{aligned}$$

  W_1 and W_2 have correlation ρ.

- PDE-setting:

$$\frac{\partial u}{\partial t} + \frac{1}{2}vs^2\frac{\partial^2 u}{\partial s^2} + \rho\sigma vs\frac{\partial^2 u}{\partial s \partial v} + \frac{1}{2}\sigma^2 v\frac{\partial^2 u}{\partial v^2} + rs\frac{\partial u}{\partial s} + \kappa(\theta - v)\frac{\partial u}{\partial v} - ru = 0.$$

- Pay-off function: $\phi(s, v, T) = \max(s - K, 0)$.

- Parameters: $\sigma = 0.25$, $r = 0.03$, $T = 1.0$, $K = 100$, $\rho = -0.5$, $\kappa = 2$ and $\theta = 0.0225$.

- Output: u at $t = 0$, $s = 90$, 100, 110 and $v = 0.0225$.

- Benchmark: the computational time to obtain the output with relative error $< 10^{-4}$ in each point.
- Graph: for the resulting parameter setting the solution u in s =[60:1:160], $v = 0.0225$ should be computed. *The computed u−the "exact" solution u* will be used for figures in the article.
- You shall submit a MATLAB-code that computes u in the measurement points $s = (90, 100, 110)$, $v = 0.0225$ with a relative error less than 10^{-4}. The code shall have the following heading:
  - `[U]=HSTeuCall_xxx(S,K,T,r,V,kap,th,sig,rho)`

2.5 Problem 5 - Merton model for one underlying asset; European call option

- SDE-setting:
$$dS = (r - \lambda\xi)Sdt + \sigma SdW + sdQ.$$
- PIDE-setting:
$$\frac{\partial u}{\partial t} + \frac{1}{2}\sigma^2 s^2 \frac{\partial^2 u}{\partial s^2} + (r - \lambda\xi)s\frac{\partial u}{\partial s} - (r + \lambda)u + \lambda \int_0^\infty u(sy, v, \tau)p(y)dy = 0.$$
- Pay-off function: $\phi(s, T) = \max(s - K, 0)$.
- The jump process Q is a compound Poisson process with intensity $\lambda > 0$ and $Q + 1$ has a log-normal distribution $p(y) = \frac{1}{\sqrt{2\pi}y\delta} e^{-\frac{(\log y - \gamma)^2}{2\delta^2}}$.
- Parameters: $\sigma = 0.15,\ r = 0.03,\ T = 1.0,\ K = 100,\ \lambda = 0.4,\ \delta^2 = 0.16\ \gamma = -0.5$ and $\xi = e^{\gamma + \delta^2/2} - 1$.
- Output: u at $t = 0$ and $s = 90$, 100, 110.
- Benchmark: the computational time to obtain the output with relative error $< 10^{-4}$ in each point.
- Graph: for the resulting parameter setting the solution u in [60:1:160] should be computed. *The computed u−the "exact" solution u* will be used for figures in the article.
- You shall submit a MATLAB-code that computes u in the measurement points $s = (90, 100, 110)$ with a relative error less than 10^{-4}. The code shall have the following heading:
  - `[U]=MRTeuCall_xxx(S,K,T,r,sig,lambda,gamma,delta)`

2.6 Problem 6 - Black-Scholes model for two underlying assets with constant volatility; European call, spread option

- SDE-setting:

$$\begin{aligned} dS_1 &= rS_1dt + \sigma_1 S_1 dW_1, \\ dS_2 &= rS_2dt + \sigma_2 S_2 dW_2. \end{aligned}$$

  W_1 and W_2 have correlation ρ.

- PDE-setting:

$$\frac{\partial u}{\partial t} + \frac{1}{2}\sigma_1^2 s_1^2 \frac{\partial^2 u}{\partial s_1^2} + \rho\sigma_1\sigma_2 s_1 s_2 \frac{\partial^2 u}{\partial s_1 \partial s_2} + \frac{1}{2}\sigma_2^2 s_2^2 \frac{\partial^2 u}{\partial s_2^2} + rs_1\frac{\partial u}{\partial s_1} + rs_2\frac{\partial u}{\partial s_2} - ru = 0.$$

- Pay-off function: $\phi(s_1, s_2, T) = \max(s_1 - s_2 - K, 0)$

- Parameters: $\sigma_1 = \sigma_2 = 0.15,\ r = 0.03,\ T = 1.0,\ K = 0,$ and $\rho = 0.5$.

- Output: u at $t = 0$ and $(s_1, s_2) = (100, 90)$, $(100, 100)$, $(100, 110)$, $(90, 100)$, $(110, 100)$.

- Benchmark: the computational time to obtain the output with relative error $< 10^{-4}$ in each point.

- Graph: for the resulting parameter setting the solution u in s_1 =[60:1:160], $s_2 = 100$ should be computed. *The computed u−the "exact" solution u* will be used for figures in the article.

- You shall submit a MATLAB-code that computes u in the measurement points in $(s_1, s_2) = (100, 90)$, $(100, 100)$, $(100, 110)$, $(90, 100)$, $(110, 100)$ with a relative error less than 10^{-4}. The code shall have the following heading:

  - `[U]=BSeuCallspread_xxx(S,T,r,sig1,sig2,rho)`
    where `S` is a $(2 \times N)$-vector with N measurement points (s_1, s_2).

A Local volatility II

The total implied variance surface in terms of the time of maturity T and the log moneyness $x = \log \frac{K}{F_T}$ in the forward price $F_T = s_0 e^{\delta T}$, where δ is the risk free interest rate (there may also be a carry cost) is given by $\sigma(T, x) = \sqrt{w_g(T, x)}$ where

$$w_g(T, x) = a + \frac{r - \ell}{2}(x - m) + \frac{r + \ell}{2}\sqrt{(x - m)^2 + s^2}, \tag{1}$$

and a, r, ℓ, m, s are parameters that depend on T. We use

$$
\begin{array}{rcrcrcl}
a &=& 0.01 + 0.03\sqrt{T+0.04} & & \frac{da}{dT} &=& \frac{0.015}{\sqrt{T+0.04}} \\
r &=& 0.06(1 - 0.87\sqrt{(T)}) & & \frac{dr}{dT} &=& \frac{-0.348}{\sqrt{T}} \\
\ell &=& 0.31(1 - 0.7\sqrt{T}) & \Rightarrow & \frac{d\ell}{dT} &=& \frac{-0.1085}{\sqrt{T}} \\
m &=& 0.03 + 0.01T & & \frac{dm}{dT} &=& 0.01 \\
s &=& 0.15(0.4 + 0.6\sqrt{T+0.04}) & & \frac{ds}{dT} &=& \frac{0.045}{\sqrt{T+0.04}}
\end{array}
$$

In order to get the local volatility in terms of T and x, we need to compute the partial derivatives of w_g

$$
\begin{array}{rcl}
\frac{\partial w_g(T,x)}{\partial T} &=& \frac{da}{dT} + \frac{(\frac{dr}{dT} - \frac{d\ell}{dT})}{2}(x-m) + \frac{r-\ell}{2}(-\frac{dm}{dT}) \\
&+& \frac{\frac{dr}{dT} + \frac{d\ell}{dT}}{2}\sqrt{(x-m)^2 + s^2} + \frac{r+\ell}{2}\frac{(x-m)(-\frac{dm}{dT}) + s\frac{ds}{dT}}{\sqrt{(x-m)^2+s^2}}, \\
\frac{\partial w_g(T,x)}{\partial x} &=& \frac{r-\ell}{2} + \frac{r+\ell}{2}\frac{(x-m)}{\sqrt{(x-m)^2+s^2}}, \\
\frac{\partial^2 w_g(T,x)}{\partial x^2} &=& \frac{r+\ell}{2}\frac{s^2}{((x-m)^2+s^2)^{3/2}},
\end{array}
\tag{2}
$$

Using Dupire's formula for the local variance expressed in these quantities we evaluate everything at T and x

$$
w_{\text{local}}(x,T) = \frac{w_g(T,x) + T\frac{\partial w_g}{\partial T}(T,x)}{\left(1 - \frac{x\frac{\partial w_g}{\partial x}}{2w_g}\right)^2 - \left(\frac{\frac{\partial w_g}{\partial x}}{2w_g}\right)^2\left(\left(\frac{w_g T}{2}+1\right)^2 - 1\right) + \frac{T\frac{\partial^2 w_g}{\partial x^2}}{2}}. \tag{3}
$$

Replacing K by s and T by t and using

$$
x = \log\frac{s}{F_t(s_0)} = \log\frac{s}{s_0 e^{\delta t}}
$$

we can compute w_{local} and $\sigma_{\text{local}} = \sqrt{w_{\text{local}}}$. Note that when s goes to zero, x blows up, and the volatility also grows rapidly, but we can evaluate the surface for values close to $s = 0$ for $0 \le t \le 0.5$.

This is implemented in the function `localvol.m`.

B Reference values

```
# BENCHOP Reference Values

Problem 1:
   1a_european_call
      Result1:
      S=[90,100,110]; sig=0.15; r=0.03; T=1.0; K=100;
         U=[2.758443856146076   7.485087593912603   14.702019669720769];
         Delta=[0.334542751969886   0.608341880846395   0.818694517094515];
         Gamma=[0.026971755100040   0.025609261020380   0.015975258690289];
         Vega=[32.770682446548165  38.413891530570481  28.995094522875153];

      Result2:
      S=[97,98,99]; sig=0.01; r=0.1; T=0.25; K=100;
         U=[0.033913177006141   0.512978189232598   1.469203342553328];
         Delta=[0.138001659888508   0.831964783803436   0.998616182178259];
         Gamma=[0.454451267361807   0.512594211115861   0.009158543351289];
         Vega=[10.689829936518105  12.307387008891816   0.224407208464969];


   1b_american_put
      Result1:
      S=[90,100,110]; sig=0.15; r=0.03; T=1.0; K=100;
         U = [10.726486710094511 4.820608184813253 1.828207584020458];
      Result2:
      S=[97,98,99]; sig=0.01; r=0.1; T=0.25; K=100;
         U = [3.000000000000682 2.000000000010786   1.000000000010715];



   1c_european_up_out
      Result1:
      S=[90,100,110]; sig=0.15; r=0.03; T=1.0; K=100; B=1.25*K;
         U=[1.822512255945242   3.294086516281595   3.221591131246868];

      Result2:
      S=[97,98,99]; sig=0.01; r=0.1; T=0.25; K=100; B=1.25*K;
         U=[0.033913177006134   0.512978189232598   1.469203342553328];




Problem 2:
S=[90,100,110]; K=100; T=0.5; r=0.03; sig=0.15; D=0.03; alpha=0.8;
   2a_european_call
      Result:
         U=[0.623811094545539   3.422712192881193   9.607009109943803]



   2b_american_call
      Result:
         U=[0.837358764004664   4.484034330080377  11.877216586990031]




Problem 3:
   Result1:
   r=0.03; T=1; K=100; sig=@(s,t)=0.15+0.15*(0.5+2*t)*(s/100-1.2).^2./((s/100).^2+1.44);
      U=[2.95517, 7.64217, 14.78425]

   Result2:
   r=0.03; T=0.5; K=100; sigma<=localvol.m
      U=[2.1433, 6.8740, 14.2983]
```

Problem 4:
S=[90,100,110]; K=100; T=1.0; r=0.03; V=0.0225; kap=2; th=0.0225; sig=0.25; rho=-0.5;
   Result:
      U=[2.302535842814927, 7.379832496149447, 14.974005277144057];

Problem 5:
S=[90,100,110]; K=100; T=1.0; r=0.03; sig=0.15; lambda=0.4; gamma=-0.5; delta=sqrt(0.16);
   Result:
      U=[7.542526012669795, 14.336250885310660, 22.359969565189871];

Problem 6:
S1=[100,100,100,90, 110;
    90,100,110,100,100] T=1.0; r=0.03; sig1=0.15; sig2=0.15; rho=0.5;
   Result:
      U=[12.021727425647768, 5.978528810578943, 2.500244806693065, 2.021727425647768, 12.500244806693061];