Constructing General Monte-Carlo Stochastic Differential Equation Solver

Khoa Tran - MATH 5111 - Professor Stojanovic -Mid-term Presentation

Source: Computational Financial Mathematics using Mathematica book.

This presentation will focus on building the general SDEsolver (i.e. the solver that can be used for both scalar and vector data).

We will then apply the apply the solver to solve (i.e. show trajectories of) some example SDEs.

SDEsolver for scalar (non-vector) data:

First, the (first-order) SDE:

$$dY(t) = f(t,y(t))dt + \sigma(t,y(t))dB(t)$$

Where f(t,y(t)) is a ordinary function, thus $f(t,y(t))dt$ is the deterministic part, while the second part $\sigma(t,y(t))dB(t)$ is the stochastic (random) part with $\sigma(t,y(t))$ is the function of standard deviation (or volatility in financial math) and $dB(t)$ as the realization of the Brownian motion for each dt.

We discretize the equation above (i.e. replace $dY(t) = y_{t+1} - y_t$) and get:

$$y_{t+1} = y_t + f(t,y(t))dt + \sigma(t,y(t))dB(t)$$

The Monte-Carlo SDEsolver for such equation is:

In[•]:=
```
SDEsolver [f_, σ_, y0_, t0_, t1_, K_] := Module[{dt, dB, G}, dt = N[(t1 - t0)/K];
  dB = Sqrt[dt] Table[Random[NormalDistribution[]], {K}];
  G[{t_, y_}, db_] := {t + dt, y + f[t, y] dt + σ[t, y] db};
  FoldList[G, {t0, y0}, dB]];
```

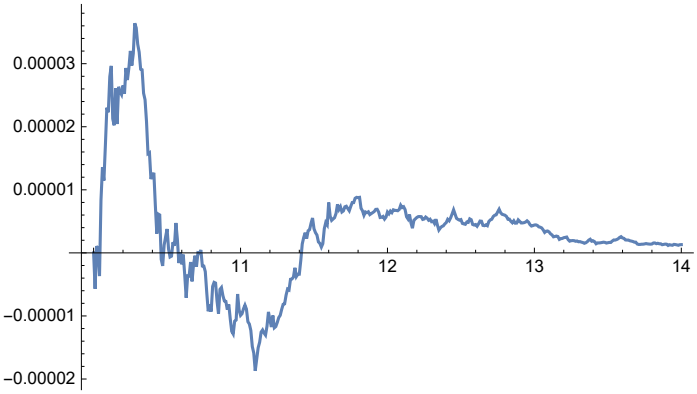
For demonstration, I'll use our old homework:

$$\int e^{-t}\, dB\,(t)$$

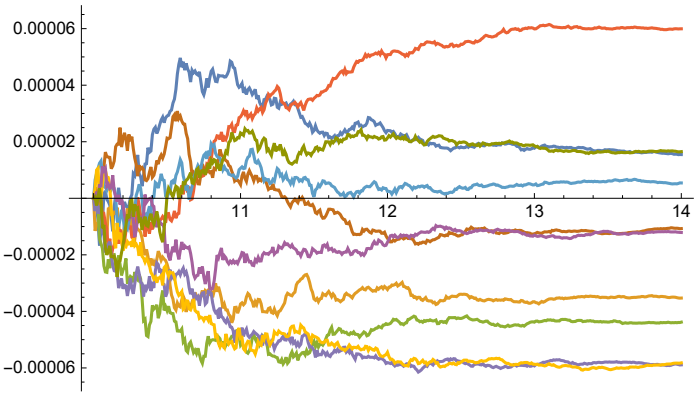
Here $dY(t) = e^{-t}dB(t)$, assuming $y(0) = 0$, t_0=10, t_1 = 14, K = 400:

In[ ]:=
```
e[t_, x_] := Exp[-t];
pl1 = ListPlot[SDE = SDEsolver[0 &, e, 0, 10, 14, 400], Joined -> True, PlotRange → All]
MultiTraject = Table[SDE = SDEsolver[0 &, Exp[-#1] &, 0, 10, 14, 400], {10}];
ListPlot[MultiTraject, Joined -> True, PlotRange → All]
```

Out[ ]=



Out[ ]=



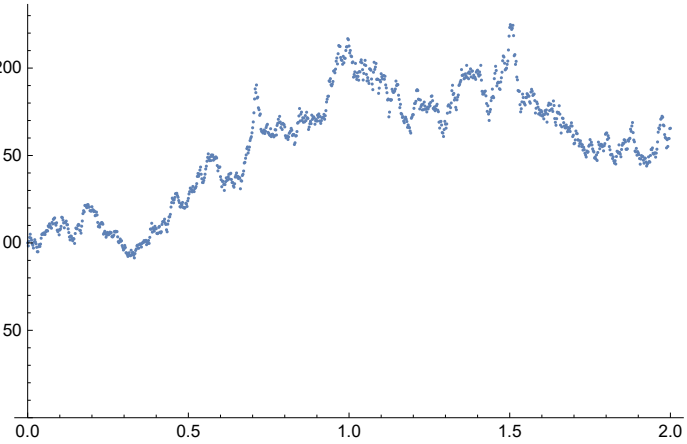
Now we proceed to apply such solver into financial context. Assumption: the stock has the price S(0) = p, it grows with constant rate a (deterministic part of the model) and volatility σ. Thus we have the model:

$dS(t) = aS(t)dt + \sigma S(t)dB(t)$

In[ ]:=
```
StockModel[a_, σ_, p_, T_] := SDEsolver[a #2 &, σ #2 &, p, 0, T, 1000]
```

In[•]:=
```
With[{T = 2}, price = StockModel[.3, .4, 100, T];
  ListPlot[price, PlotRange → {0, Automatic}]];
```

Out[•]=



SDEsolver for m-dimensional data and General SDEsolver:

After seeing the application of such solver for modeling 1 stock price evolution, we attempt to expand the solver to be able to solve for multivariate SDEs:

In[•]:=
```
SDEsolver2[f_, σ_, y0_, t0_, t1_, K_] := Module[{n, dt, dB, G}, n = Length[y0];
  dt = N[(t1 - t0)/K];
  dB = Sqrt[dt] Table[Random[NormalDistribution[]], {K}, {n}];
  FF[{t_, y_}, db_] := {t + dt, y + b[t, y] dt + c[t, y].db};
  FoldList[FF, {t0, y0}, dB]];
```

Having constructed the SDEsolver2, it would be useful for use to combine the 2 solvers into an if statement so the program can decide which solver to use based on whether the b function is a list or not.

In[•]:=
```
GeneralSDEsolver[f_, σ_, y0_, t0_, t1_, K_] := If[Head[f[t0, y0]] === List,
  SDEsolver2[f, σ, y0, t0, t1, K], SDEsolver[f, σ, y0, t0, t1, K]]
```

To conclude the presentation, let's apply the GeneralSDEsolver to model 2 stock price evolutions with multiple independent Brownian motions (sources of randomness)

$dS(t) = S(t)a\,dt + S(t)\sigma\,dB(t)$

Where:

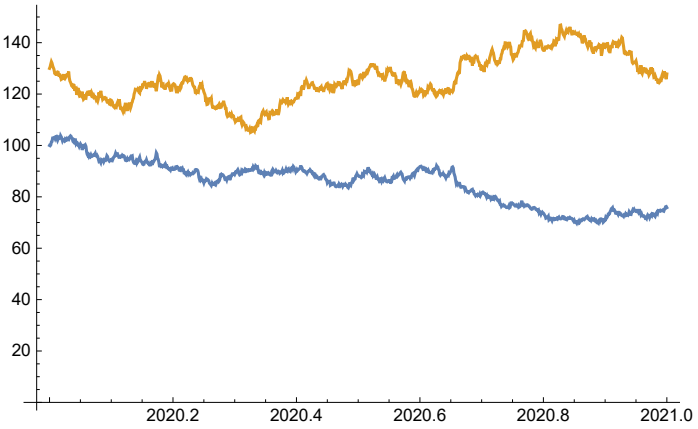
In[•]:=
```
𝕡1 = .2; 𝕡2 = .25; ρ = -.4;
```

In[•]:=
```
b[t_, {y1_, y2_}] := {.1 y1, -.15 y2}
```

In[•]:=
```
c[t_, {y1_, y2_}] := {y1 𝕡1, y2 𝕡2} {{1, 0}, {ρ, Sqrt[1 - ρ^2]}}
```

In[ ]:=
```
t0 = 2020; t1 = 2021; K = 1000; dt = N[(t1 - t0)/K]; y0 = {100, 130};
SDEx = GeneralSDEsolver[b, c, y0, t0, t1, K];
ListPlot[Table[{#[[1]], #[[2, k]]} & /@ SDEx, {k, 2}], Joined → True]
```

Out[ ]=



Deriving Forward Formula from Black-Scholes-Merton Partial Differential Equation

Khoa Tran - MATH 5111 - Professor Stojanovic

First, we start with the Black-Scholes-Merton PDE for the log-normal market dynamics, that is:

$V_t(t,S)+\frac{1}{2}S^2\mathbb{p}^2 V_{SS}(t,S)+S(r-\mathbb{D})\,V_S(t,S)-rV(t,S)=-\mathcal{D}$

and the terminal condition V[T,S] = φ[S] where φ[S] is the payoff function. We implement such equation with the following code:

In[•]:=
```
LogNormBSPDE =
 {D[V[t, S], t] + 1/2 S^2 ℙ^2 D[V[t, S], {S, 2}] + S (r - 𝔻) D[V[t, S], S] - r V[t, S] == -𝒟[t, S],
  V[T, S] == φ[S]}
```

Out[•]=
$$\left\{-r\,V[t,S]+S\,(r-\mathbb{D})\,V^{(0,1)}[t,S]+\frac{1}{2}S^2\,\mathbb{p}^2\,V^{(0,2)}[t,S]+V^{(1,0)}[t,S]==-\mathcal{D}[t,S],\ V[T,S]==\varphi[S]\right\}$$

In the forward contract, the party entering the agreement has the obligation to buy or sell the underlying asset at the strike price K in a future date. This dictates the payoff function φ[S] = S - K = V[T,S]. In English language, the payoff (or loss) is the difference between the asset price and the strike price at maturity. We also assume that the contract pay no dividend, i.e. $\mathcal{D}$ = 0. Thus, we derive the PDE for forward contract:

In[•]:=
```
ForwardPDE = LogNormBSPDE /. {𝒟 → (0 &), φ → (S - K &)} // Simplify
```

Out[•]=
$$\left\{S\,(r-\mathbb{D})\,V^{(0,1)}[t,S]+\frac{1}{2}S^2\,\mathbb{p}^2\,V^{(0,2)}[t,S]+V^{(1,0)}[t,S]==r\,V[t,S],\ K+V[T,S]==S\right\}$$

Next, we need to define the form of V[t,S]. Looking the payoff function and with the reasoning that the value of the contract at any given time is the discounted value of the payoff, let V[t,S] = $b_0(t) + b_1(t)\,S$. We implement this into codes:

In[•]:=
```
ForwardPDE2 = ForwardPDE /. V → (b0[#1] + #2 b1[#1] &) // Simplify
```

Out[•]=
$$\{r\,b_0[t]+S\,\mathbb{D}\,b_1[t]==b_0'[t]+S\,b_1'[t],\ K+b_0[T]+S\,b_1[T]==S\}$$

Then, we proceed to solve the Black-Scholes PDE for $b_0(t)$ and $b_1(t)$. Here, we can solve this system of 2 equations algebraically by matching the coefficients of each level of power of S, that is:

$r\,b_0[t]=b_0'[t];\ \mathbb{D}\,b_1[t]=b_1'[t];\ b_0[T]+K=0;\ b_1[T]=S$

We can also implement codes to solve this system as the following:

```
In[•]:= Sol = (0 == # & /@ CoefficientList[Subtract @@ #, S]) & /@ (ForwardPDE2) // Simplify
Out[•]= {{r b0[t] == b0'[t], 𝔻 b1[t] == b1'[t]}, {K + b0[T] == 0, b1[T] == 1}}
```

We plug this result into a Differential Equation Solver to find $b_0(t)$ and $b_1(t)$. But first, we need to flatten the list of 2 list down to just one list.

```
In[•]:= FlatSol = Flatten[(0 == # & /@ CoefficientList[Subtract @@ #, S]) & /@ (ForwardPDE2)]
Out[•]= {0 == r b0[t] - b0'[t], 0 == 𝔻 b1[t] - b1'[t], 0 == K + b0[T], 0 == -1 + b1[T]}

In[•]:= DSolve[FlatSol, {b0, b1}, t]
Out[•]= {{b0 → Function[{t}, -ℯ^(r t-r T) K], b1 → Function[{t}, ℯ^(t 𝔻-T 𝔻)]}}
```

Having solved for $b_0(t)$ and $b_1(t)$, we arrive at our formula for forward contract pricing V(t,S), which is:

```
In[•]:= FW_k_,r_,𝔻_,T_[t_] =
          b0[t] + S b1[t] /. {b0 → Function[{t}, -ℯ^(r t-r T) k], b1 → Function[{t}, ℯ^(t 𝔻-T 𝔻)]} // Simplify
Out[•]= -ℯ^(r (t-T)) k + ℯ^((t-T) 𝔻) S
```

Graphing The Implied Volatility Smile

Khoa Tran - MATH 5111 - Professor Stojanovic

In order to graph and observe the implied volatility smile, we use the data for option quotes of Google. First, we import the data from the csv file and clean it in Mathematica:

```
In[•]:= SetDirectory["C:\\Users\\binbe\\Desktop\\lesson\\computational finance"];
fn = FileNames["2020-10-27*"][[1]];
DecimalTime[x_] :=
  x[[1]] + (FromDate[x] - FromDate[{x[[1]], 1, 1, 0, 0, 0}]) /
    (FromDate[{x[[1]] + 1, 1, 1, 0, 0, 0}] - FromDate[{x[[1]], 1, 1, 0, 0, 0}]) // N;
UnderlyingLegend = {"LAST", "LX", "Net Chng", "BID", "BX",
   "ASK", "AX", "Size", "Volume", "Open", "High", "Low"};
AllOptionsData = Import[fn, "Data"];
timeOfRecord = With[{x = AllOptionsData[[1, 1]]},
   DecimalTime[DateList[StringTake[x, {StringPosition[x, " on"][[1, 2]] + 2, -1}]]]];
lbau = Position[UnderlyingLegend, #][[1, 1]] & /@ {"LAST", "BID", "ASK"};
OptionsLegend = {"", "", "LAST", "LX", "Net Chng", "BID", "BX", "ASK", "AX",
   "Exp", "Strike", "BID", "BX", "ASK", "AX", "LAST", "LX", "Net Chng", "", ""};
lbacp = Transpose[Flatten /@ (Position[OptionsLegend, #] & /@ {"LAST", "BID", "ASK"})];
lbacp = Insert[lbacp, Flatten[Position[OptionsLegend, #] & /@ {"Exp", "Strike"}], 2];
UnderlyingLBAPrices =
  AllOptionsData[[Position[AllOptionsData, {"LAST", "LX", "Net Chng", "BID", "BX", "ASK",
          "AX", "Size", "Volume", "Open", "High", "Low"}][[1, 1]] + 1]][[#]] & /@ lbau;
OptionsSeparators = Flatten[Position[AllOptionsData, {"", "", "LAST", "LX",
     "Net Chng", "BID", "BX", "ASK", "AX", "Exp", "Strike", "BID",
     "BX", "ASK", "AX", "LAST", "LX", "Net Chng", "", ""}]];
OptionsSeparators2 = {1, -1} + # & /@ Partition[OptionsSeparators, 2, 1];
recordLength = Length[{"", "", 0, "", 0, 629.6`, "X",
    637.3`, "X", "4 DEC 20", 960, 0, "H", 0.6`, "Z", 0.45`, "N", 0, "", ""}];
AllOptionsData2 = Select[Take[AllOptionsData, #], Length[#] == recordLength &] & /@
   OptionsSeparators2;
AllRelevantOptionsData = Function[record, record[[#]] & /@ # & /@ lbacp] /@ # & /@
   AllOptionsData2;
AllRelevantOptionsData2 =
  Function[y, ReplacePart[y, {2, 1} → DecimalTime[DateList[y[[2, 1]]]]] /@ # & /@
   AllRelevantOptionsData;
AllRelevantOptionsData3 =
  Function[y, ReplacePart[ReplacePart[y, 1 → Mean[Drop[y[[1]], 1]]],
      3 → Mean[Drop[y[[3]], 1]]]] /@ # & /@ AllRelevantOptionsData2;
```

After cleaning the data, our data is stored in AllRelevantOptionsData3, which contains info on options as following:

```
In[•]:= AllRelevantOptionsData3[[1, 1]]
Out[•]= {772.95, {2020.83, 820}, 1.8}
```

with the first element is the mean of bid and ask prices for call option [772.95], the second element contains the maturity date [2020.83] and the strike price [820] respectively, and the third element is the mean of bid and ask prices for the put option [1.8].

Next, we define the value functions for the call options, VC, and put options, VP, as following:

In[•]:=
```
VC[t_, S_, T_, k_, a_, b_, σ_] := 1/2 ℯ^(b (t-T)) (ℯ^(a (t-T)) S (Erf[(2 Log[S/k] - (t - T) (σ^2 + 2 a))/(2 Sqrt[2] Sqrt[(T - t) σ^2])] + 1) +
    k (Erfc[(2 Log[S/k] - (t - T) (2 a - σ^2))/(2 Sqrt[2] Sqrt[(T - t) σ^2])] - 2));
VP[t_, S_, T_, k_, a_, b_, σ_] := 1/2 ℯ^(b (t-T))
  (k Erfc[(2 Log[S/k] - (t - T) (2 a - σ^2))/(2 Sqrt[2] Sqrt[(T - t) σ^2])] - ℯ^(a (t-T)) S Erfc[(2 Log[S/k] - (t - T) (σ^2 + 2 a))/(2 Sqrt[2] Sqrt[(T - t) σ^2])])
```

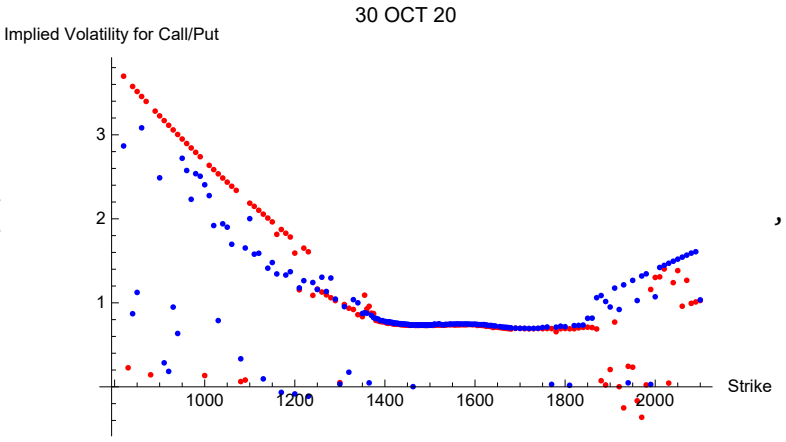
Then, we determine the implied volatility by solving for σ as VC[σ] = call price or VP[σ]= put price in the market data above. In order to graph the "smile", we'll calculate such implied volatility for all the strike price K at each maturity date. Note that here we loop through the strike prices and the maturity date by iterating through the column (col) and the row (row) of the data respectively (note, we iterate through each row first and then through columns of that row because the number of columns for each row might be different) , as following:

In[•]:=
```
row = Length[AllRelevantOptionsData3];
put[r_, c_] := FindRoot[AllRelevantOptionsData3[[r, c]][[3]] ==
    VP[timeOfRecord, Mean[Drop[UnderlyingLBAPrices, 1]],
     AllRelevantOptionsData3[[r, c]][[2, 1]], AllRelevantOptionsData3[[r, c]][[2, 2]],
     0.0025 - 0.0001, 0.0025, 𝕡], {𝕡, Random[Real, {0, 2}]}];
putvol = Table[{AllRelevantOptionsData3[[r, c, 2, 2]], 𝕡} /. put[r, c],
   {r, 1, row}, {c, 1, Length[AllRelevantOptionsData3[[r]]]}];
call[r_, c_] := FindRoot[AllRelevantOptionsData3[[r, c]][[1]] ==
    VC[timeOfRecord, Mean[Drop[UnderlyingLBAPrices, 1]],
     AllRelevantOptionsData3[[r, c]][[2, 1]], AllRelevantOptionsData3[[r, c]][[2, 2]],
     0.0025 - 0.0001, 0.0025, 𝕡], {𝕡, Random[Real, {0, 2}]}];
callvol = Table[{AllRelevantOptionsData3[[r, c, 2, 2]], 𝕡} /. call[r, c],
   {r, 1, row}, {c, 1, Length[AllRelevantOptionsData3[[r]]]}];
```

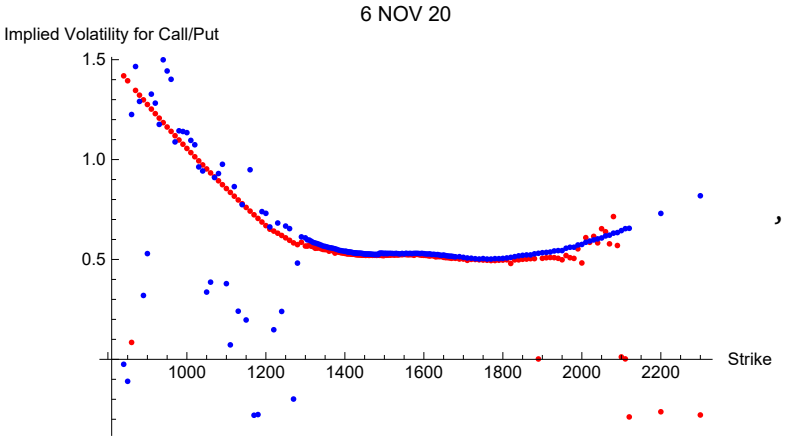
Finally, we graph both of the put and call implied volatilities on 6 graphs (for 6 different maturity date in our data, according to 6 rows) to observe the smiles (red = put, blue = call):

In[•]:=
```
Show[Show[ListPlot[#[[1]], PlotStyle → Red,
     AxesLabel → {"Strike", "Implied Volatility for Call/Put"}, PlotRange → All],
    ListPlot[#[[2]], PlotStyle → Blue], ImageSize → 400], PlotLabel → #[[3]]] & /@
 Transpose[{putvol, callvol, #[[1]] & /@ (#[[2, 1]] & /@ # & /@ AllRelevantOptionsData)}]
```

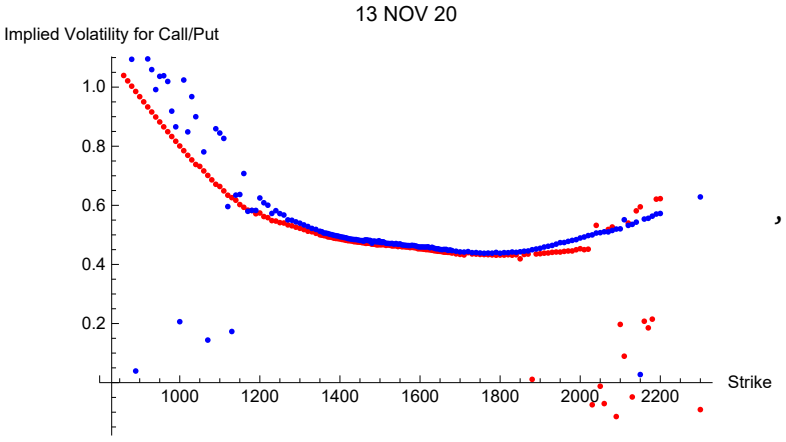
Out[•]= {


30 OCT 20
Implied Volatility for Call/Put
3
2
1
1000
1200
1400
1600
1800
2000
Strike

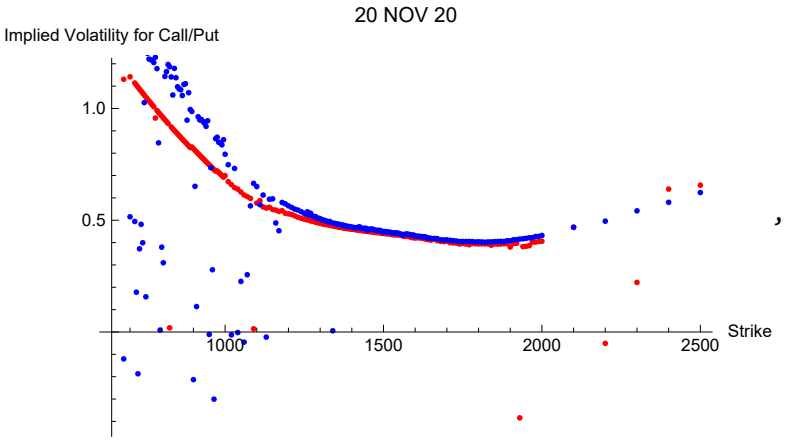
,

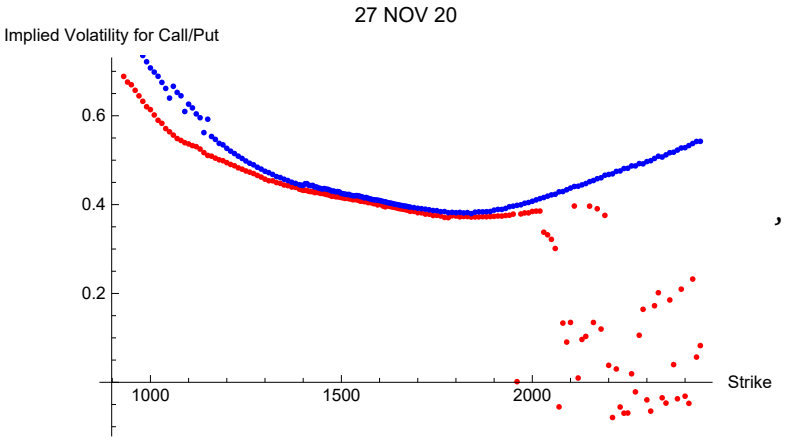

6 NOV 20
Implied Volatility for Call/Put
1.5
1.0
0.5
1000
1200
1400
1600
1800
2000
2200
Strike

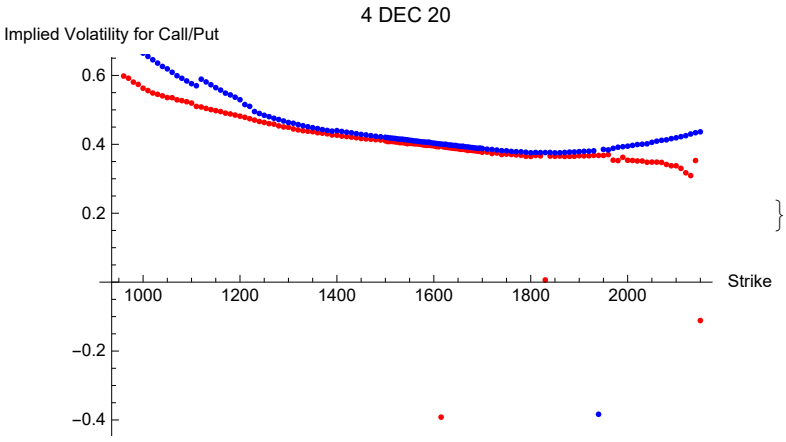
,


13 NOV 20
Implied Volatility for Call/Put
1.0
0.8
0.6
0.4
0.2
1000
1200
1400
1600
1800
2000
2200
Strike

,







These smiles imply that volatilities in the market are not constant.