
# EXCHANGE RATE TRIANGLE CALIBRATION

## 3A Research Project

March 19, 2017

Authors : Lucas Furquim
Felipe García
Tutor : Benjamin Jourdain

ÉCOLE POLYTECHNIQUE
UNIVERSITÉ PARIS-SACLAY

# CONTENTS

# 1
# PROJECT DESCRIPTION

In this project we describe a calibration method based on [Guyon, 2013] for three exchange rates with local volatility. for example between USD, EUR and GBP. We already know how to calibrate the two exchange rates $S^1$ = USD/EUR and $S^2$ = EUR/GBP. To calibrate the third exchange rate we need to take into account that $S^{12} = S^1/S^2$. In this project we present a model on how to calibrate the third exchange rate.

In this context, this project will provide a general theoretical analysis volatility regimes, considering implied volatility, local volatility models and models on exchange rates. We last show an equation **(4.0.14)** that shows us how to calibrate this three exchange rate model.

# 2
# GENERAL BACKGROUND

## 2.1 Implied Volatility

In the Black-Scholes model, options' price is uniquely determined by the only unobservable parameter: the volatility. Knowing Put and Call market prices, we can invert Black-Scholes equation and extract this parameter, named in this case "implied volatility"(IV).

$$C = SN(d_+) - e^{-r(T-t)}KN(d_-),\ t \in [0,T] \tag{2.1.1}$$

$$d_\pm = \frac{\ln(\frac{S}{K} + (r \pm \sigma^2/2)(T-t))}{\sigma\sqrt{T-t}} \tag{2.1.2}$$

In this framework, IV should be constant across all strikes and maturities and equal to the historical volatility (standard deviation of annualized log returns) of the underlying. However, when IV is computed from market quoted option prices, one observes that:

- IV and historical volatility are generally different. When IV is greater than historical volatility, options are thought to be overvalued, and when IV is less than historical volatility, options are considered to be undervalued;

- IV of different options on the same underlying depend on their maturities and strikes.

The implied volatility, then, becomes a non-constant function of strike, and the shape of the curve denotes a phenomenon known as volatility smile or skew (in equity speak).

Volatility skew can be used to identify trading opportunities. In practice, implied volatility allows traders to compare options with different strikes and expiration dates. The implied volatility enables traders to gain a better perspective of derivatives' market. Since supply and demand ultimately drive prices, traders can learn which options are "cheap" or "expensive", relative to others, as measured by the implied volatility of each option.

The parameter $\sigma$ in **(2.1.2)** corresponds to the average volatility of the underlying asset during the lifetime of the contract. This is the only parameter in the Black & Scholes model that is not directly observable in the financial markets. In order to find the market price of a publicly traded option by using the Black & Scholes model, it is necessary to find a matching value for $\sigma$. Generally, the value of $\sigma$ which produces a market price is called *implied volatility*.

**Defintion.** *The implied volatility is the volatility which makes **(2.1.2)** generate a price consistent with the price of a market quoted call option.*

The implied volatility is found by inverting **(2.1.2)** with respect to the volatility parameter, using an option price quoted on the financial markets with known strike and maturity. Since **(2.1.2)** is not analytically invertible with respect to $\sigma$,the implied volatility has to be found using numerical techniques.

The implied volatility for a grid of market quoted options with different maturities and strikes is generally not constant. The strike and maturity dependency in implied volatility is caused by the financial markets attaching higher probabilities to extreme movements in log-returns compared to the normal distribution. As a function of strike, the implied volatility for FX-options usually assumes the **shape of a smile**. This curve is generally denoted the implied volatility smile.

It is possible to add maturity dependency in $\sigma$ while keeping a closed-form solution similar to **(2.1.2)**. Assume that the volatility is a deterministic function of time, $\sigma(t)$. Assuming $\sigma(t)$ to be a step function allows for arbitrage-free calibration to market prices on options with any set of maturities. This simple approach is sadly not applicable in the strike dimension, which leads us to the next section.

## 2.2 Local Volatility and Dupire Formula

The local volatility model is a generalization of the Black & Scholes model. The model was first proposed by Dupire (1994) and has been further developed by Derman and Kani (1998) among others. The model is based on the assumption that the volatility is a general deterministic function dependent on time and the contemporaneous value of the underlying asset. This generalization makes it possible to create a risk-neutral probability distribution of the underlying asset which is consistent with an entire market quoted implied volatility surface.

To emphasize one of the strengths of the local volatility model we need the following definition.

**Definition.** *A model is complete if all contingent claims can be perfectly hedged.*

Since the local volatility model does not introduce any further sources of risk, the model is a complete market model where assets theoretically can be perfectly hedged by using a continuously updated $\delta$-hedging strategy.

Assume that the underlying asset follows a stochastic process with the dynamics

$$dS_t = S_t\sigma(S_t, t)dW_t^{\mathbb{Q}}, t \in [0, T] \tag{2.2.1}$$

where $(W_t^{\mathbb{Q}})_{t\in[0,T]}$ is a standard Brownian motion under the risk-neutral probability measure. The central equation in the local volatility model, the Dupire equation, is presented below together with a proof similar to the one by Dupire (1994).

**Theorem.** *(Dupire Equation for European Call Options) Let $C(K, T)$ denote European call options with strikes $K \in \kappa$ , maturities $T \in \theta$ , and under-lying asset $S$. Assume that the price of $S$ follows the dynamics in **(2.2.1)**, with initial condition $S_t = S$. The prices of the call options at time $t$ will then satisfy the equation:*

$$\frac{\partial C}{\partial T}(K, T) = \frac{1}{2}\sigma^2(K, T)K^2\frac{\partial^2 C}{\partial K^2}(K, T) \tag{2.2.2}$$

*with boundary condition:*

$$\lim_{T\to t} C(K, T) = (S - K)^+. \tag{2.2.3}$$

While this equation looks very similar to the Black & Scholes equation, the two equations have fundamentally different meaning. The Black & Scholes equation describes the evolution of the

price of any European contingent claim over time, holding claim-specific parameters such as strike $K$ and maturity $T$ constant. The Dupire equation describes the dynamics in a grid of European call option prices, holding the contemporaneous value of the underlying asset $S$ and time $t$ constant. The two equations are often referred to as the forward and backward equations. This convention originates from the fact that the Black & Scholes equation has boundary conditions at the terminal date $t = T$, and hence needs to be solved backwards in time, while the Dupire equation has boundary conditions at $T = t$ and needs to be solved forward in maturity. The remaining part of this section consists of a proof of the Dupire equation.

Consider a European call option $C(S, t)$ with some underlying asset $S$, following the dynamics in **(2.2.1)**. Assume further that the call option has maturity $T$ and strike $K$. Denote the risk-neutral density of the underlying at time $t$ as $\varphi(S, t)$. The price of the option can then be calculated as the expected payoff under the risk-neutral probability measure:

$$C(S,T) = E^Q[(S_t - K)\mathbf{1}_{S_t-K>0}|S_t = S] \tag{2.2.4}$$

Using the definition of expected value, this expression can be formulated as the integral

$$C(S,T) = \int_0^\infty (x-K)\mathbf{1}_{x>K}\varphi(x,T)dx = \int_K^\infty (x-K)\varphi(x,T)dx, \tag{2.2.5}$$

where $\varphi(x, T)$ is the probability density function of the underlying asset at maturity, conditioned on $S_t = S$. In order to continue the derivation, the following theorem, usually called the Fokker-Planck theorem, is needed.

**Theorem.** (Fokker-Planck) *Let $X_t$ be a N-dimensional stochastic process with uncertainty driven by a M-dimensional standard Brownian motion $W_t$:*

$$dX_t = \mu(X_t, t)dt + \sigma(X_t, t)dW_t, \tag{2.2.6}$$

where $\mu(X_t, t) = (\mu(X_t, t) \cdots \mu_n(X_t, t))$ is a N-dimensional drift vector and $\sigma(X_t, t)$ is a diffusion tensor. Then the joint probability function $f(x, t)$ satisfies the Fokker-Planck equation

$$\frac{\partial f(x,t)}{\partial t} = -\sum_{i=1}^{N}\frac{\partial}{\partial x_i}[\mu_i(x,t)f(x,t)] + \sum_{i=1}^{N}\sum_{j=1}^{N}\frac{\partial}{\partial x_i \partial x_j}[D_{i,j}(x,t)f(x,t)], \tag{2.2.7}$$

where $D_{i,j} = \frac{1}{2}\sum_{k=1}^{M}\sigma_{i,k}(x,1)\sigma_{j,k}(x,1), 1 \le i,j \le N$.
Applying this theorem to the driftless one-factor framework which is of interest yields the Fokker-Planck equation

$$\frac{\partial}{\partial t}(\varphi(S,t)) = \frac{1}{2}\frac{\partial^2}{\partial S^2}(S^2\sigma^2(S,t)\varphi(S,t)). \tag{2.2.8}$$

The solution of this equation will (in accordance with The Second Fundamental Theorem of Asset Pricing) be assumed to be unique provided that the probability distribution is restricted to the risk neutral one. In order to proceed, some differentials of the option price with respect to the strike $K$ and maturity $T$ are needed. We will from now on let $K$ and $T$ be variables and $S$ and $t$ static parameters. The notation for a call option is also changed to $C(K,T)$, but keep in mind that the price is still conditioned on $S_t = S$. Using the Leibniz integral rule on **(2.2.5)** and assuming that $\lim_{S\to\infty} \varphi(S,T) = 0$ yields:

$$\frac{\partial C}{\partial K}(K,T) = -\int_K^\infty \varphi(x,T)dx, \tag{2.2.9}$$

$$\frac{\partial^2 C}{\partial K^2}(K,T) = \varphi(K,T), \tag{2.2.10}$$

$$\frac{\partial C}{\partial T}(K,T) = \int_K^\infty (x-K)\frac{\partial}{\partial T}(\varphi(x,T))dx. \tag{2.2.11}$$

Substituting **(2.2.8)** at time $t = T$ into **(2.2.11)** yields:

$$\frac{\partial C}{\partial T}(S,T) = \int_K^\infty (x-K)(\frac{1}{2}\frac{\partial^2}{\partial x^2}(x^2\sigma^2(x,T)\varphi(x,T))dx. \tag{2.2.12}$$

Integrating this expression by parts two times and using the equations **(2.2.9)** and **(2.2.10)** yields:

$$\int_K^\infty (x-K)(\frac{1}{2}\frac{\partial^2}{\partial x^2}(x^2\sigma^2(x,T)\varphi(x,T))dx = \frac{1}{2}\sigma^2(K,T)K^2\frac{\partial^2 C}{\partial K^2}. \tag{2.2.13}$$

Substituting **(2.2.13)** into **(2.2.12)** yields the Dupire PDE for European call options. The boundary condition of the above equation simply states that an option with time to maturity $\tau = T - t = 0$ will have the value equal to the call option payoff function:

$$\lim_{T\to t} C(K,T) = (S-K)^+. \tag{2.2.14}$$

# 3
# MODEL FOR EXCHANGE RATE OPTIONS

Assume that there are riskless assets USD and GBP in dollars and British pounds sterling, respectively, with riskless rates of return $r^d$ and $r^f$. Because of uncertainty about future exchange rates, the asset GBP does not appear riskless to the dollar investor, nor does the asset USD appear riskless to the pound sterling investor; the choice of numeraire (dollar or pound sterling) determines which asset is riskless. Let's take the point of view of the pound-sterling investor. Let $Y_t$ be the rate of exchange at time $t$ (that is, $Y_t$ is the number of British pounds that one dollar will buy at time $t$). In the simplest model, $Y_t$ behaves like a geometric Brownian motion, that is, it follows a stochastic differential equation of the form

$$dY_t = \mu Y_t dt + \sigma Y_t dW_t \tag{3.0.1}$$

where $W_t$ is a Brownian motion. Let $A_t$ and $B_t$ denote the share prices of the assets USD and GBP, reported in units of dollars and British pounds, respectively, and normalized so that the time-zero share prices are both 1. Then

$$A_t = e^{r^d t} \tag{3.0.2}$$

$$B_t = e^{r^f t} \tag{3.0.3}$$

The share price of US dolar at time $t$ in pounds sterling is $A_t Y_t$. Solving the stochastic differential equation **(3.0.1)** gives the explicit formula

$$A_t Y_t = Y_0 \exp\left\{r^d t + \mu t - \sigma^2 t/2 + \sigma W_t\right\}. \tag{3.0.4}$$

Now let $\mathbb{Q}^f$ be the risk neutral measure of the english pound. Under $\mathbb{Q}^f$, the discounted price of the asset USD is $\exp\left\{-r^f t\right\} A_t Y_t$, and therefore by equation **(3.0.4)** equals

$$\exp\left\{-r^f t\right\} A_t Y_t = Y_0 \exp\left\{(r^d - r^f)t + \mu t - \sigma^2 t/2 + \sigma W_t\right\} \tag{3.0.5}$$

The second exponential is by itself a martingale, and the first exponential is nonrandom. Thus, in order that the product of the two exponentials be a martingale it must be that $r^d - r^f + \mu = 0$. So $\mu = r^f - r^d$. Therefore the exchange rate for $Y_t$ is given by

$$dY_t = (r^f - r^d)Y_t dt + \sigma Y_t dW_t. \tag{3.0.6}$$

# 4

# THE FX TRIANGLE CALIBRATION PROBLEM

Let $S^1$, $S^2$ be two FX rates, and $S^1/S^2$. For example $S^1$ = EUR/USD, $S_2$ = GBP/USD, and $S^{12} = S^1/S^2$ = EUR/GBP (the cross rate). Assuming that the implied volatility $S^1$, $S^2$ and $S^{12}$ until some maturity $T$ are known from the market, and that those surfaces are jointly arbitrage-free. They correspond to three local volatility surfaces that we denote by $\sigma_1(t, S^1)$, $\sigma_2(t, S^2)$, and $\sigma_{12}(t, S^{12})$. Assume the following model for the dynamics of $S^1$ and $S^2$:

$$\begin{aligned} dS_t^1 &= (r_t^d - r_t^1)S_t^1\, dt + \sigma_1(t, S_t^1)S_t^1\, dW_t^1 \\ dS_t^2 &= (r_t^d - r_t^2)S_t^2\, dt + \sigma_2(t, S_t^2)S_t^2\, dW_t^2 \\ d\left\langle W^1, W^2\right\rangle_t &= \rho(t, S_t^1, S_t^2)\, dt \end{aligned} \tag{4.0.1}$$

We know from the Ito's quotient rule that:

$$\frac{dS_t^{12}}{S_t^{12}} = \frac{dS_t^1}{S_t^1} - \frac{dS_t^2}{S_t^2} + \left(\frac{dS_t^2}{S_t^2}\right)^2 - \frac{dS_t^1}{S_t^1}\frac{dS_t^2}{S_t^2} \tag{4.0.2}$$

So, from (**4.0.1**):

$$\frac{dS_t^{12}}{S_t^{12}} = (r_t^2 - r_t^1)\, dt + \sigma_1\, dW_t^1 - \sigma_2\, dW_t^2 + \sigma_2^2\, dt - \sigma_1\sigma_2\, d\left\langle W^1, W^2\right\rangle_t \tag{4.0.3}$$

Since $d\left\langle W^1, W^2\right\rangle_t = \rho(t, S_t^1, S_t^2)\, dt$:

$$\frac{dS_t^{12}}{S_t^{12}} = (r_t^2 - r_t^1)\, dt + \sigma_1\, dW_t^1 - \sigma_2\, dW_t^2 + \sigma_2^2 dt - \rho\sigma_1\sigma_2 dt \tag{4.0.4}$$

$$\frac{dS_t^{12}}{S_t^{12}} = (r_t^2 - r_t^1)dt + \sigma_1\, \left(dW_t^1 - \rho\sigma_2\, dt\right) - \sigma_2(dW_t^2 - \sigma_2 dt) \tag{4.0.5}$$

$$\frac{dS_t^{12}}{S_t^{12}} = (r_t^2 - r_t^1)\, dt + \sigma_1\left(t, S_t^1\right) dW_t^{1,f} - \sigma_2(t, S_t^2) dW_t^{2,f} \tag{4.0.6}$$

where:

$$W_t^{1,f} = W_t^1 - \int_0^t \rho(s, S_s^1, S_s^2)\sigma_2(s, S_s^2)\, ds \tag{4.0.7}$$

$$W_t^{2,f} = W_t^2 - \int_0^t \sigma_2(s, S_s^2)\, ds \tag{4.0.8}$$

We can see by Girsanov Theorem (using the Novikov's condition) that $W_t^{1,f}$ and $W_t^{2,f}$ are Brownian motions under the risk neutral measure $\mathbb{Q}^f$ defined by

$$\frac{d\mathbb{Q}^f}{d\mathbb{Q}} = \frac{S_T^2}{S_0^2} \exp\left(\int_0^T (r_t^2 - r_t^d)\, dt\right) \tag{4.0.9}$$

Now as $W_t^{1,f}$ and $W_t^{2,f}$ are Brownian motions under the risk neutral measure $V$ we conclude that also is

$$W_t^f = \int_0^t \frac{\sigma_1(s, S_s^1) dW_s^{1,f} - \sigma_2(s, S_s^2) dW_s^{2,f}}{a_s}\, ds \tag{4.0.10}$$

where $a_t^2 = \sigma_1^2(t, S_t^1) + \sigma_2^2(t, S_t^2) - \rho(t, S_t^1, S_t^2)\sigma_1^2(t, S_t^1)\sigma_2^2(t, S_t^2)$. Being a normalized linear combination of two Brownian Motions. Thus we have

$$\frac{dS_t^{12}}{S_t^{12}} = (r_t^2 - r_t^1)dt + a_t dW_t^f \tag{4.0.11}$$

But we recall that we also have via Garman-Kohlhagen for the third exchange rate that

$$\frac{dS_t^{12}}{S_t^{12}} = (r_t^2 - r_t^1)dt + \sigma_{12}(t, S_t^{12}) dW_t^f \tag{4.0.12}$$

to end the proof we use the Gyongy's theorem from the appendix. Thus we have:

$$\mathbb{E}(a_t^2 | S_t^{12}) = \sigma_{12}^2 \tag{4.0.13}$$

which is equivalent to the calibration requirement

$$\mathbb{E}_\rho^{\mathbb{Q}^f}\left[\sigma_1^2(t, S_t^1) + \sigma_2^2(t, S_t^2) - 2\rho(t, S_t^1, S_t^2)\sigma_1^2(t, S_t^1)\sigma_2^2(t, S_t^2) | S_t^{12}\right] = \sigma_{12}^2(t, S_t^{12}) \tag{4.0.14}$$

From now on, we will denote $\mathcal{C}$ the set of functions $\rho : [0, T] \times \mathbb{R}_+^* \times \mathbb{R}_+^* \to [-1, 1]$. And any $\rho \in \mathcal{C}$ satisfying **(4.0.14)** will be called "**admissible correlation**".

Let us now pick two functions $a(t, S^1, S^2)$ and $b(t, S^1, S^2)$ such that b does not vanish and

$$a(t, S^1, S^2) + b(t, S^1, S^2)\rho(t, S^1, S^2) \equiv f\left(t, \frac{S^1}{S^2}\right) \tag{4.0.15}$$

is local in cross. Then:

$$\begin{aligned}
\sigma_{12}^2(t, \frac{S_t^1}{S_t^2}) &= \mathbb{E}_\rho^{\mathbb{Q}^f}\left[\sigma_1^2(t, S_t) + \sigma_2^2(t, S_t^2) - 2\rho(t, S_t^1, S_t^2)\sigma_1(t, S_t^1)\sigma_2(t, S_t^2) | \frac{S_t^1}{S_t^2}\right] \\
&= \mathbb{E}_\rho^{\mathbb{Q}^f}\left[\sigma_1^2(t, S_t^1) + \sigma_2^2(t, S_t^2) + 2\frac{a(t, S_t^1, S_t^2)}{b(t, S_t^1, S_t^2)}\sigma_1(t, S_t^1)\sigma_2(t, S_t^2) | \frac{S_t^1}{S_t^2}\right] \\
&\quad - 2(a + b\rho)\left(t, \frac{S_t^1}{S_t^2}\right) \mathbb{E}_\rho^{\mathbb{Q}^f}\left[\frac{\sigma_1(t, S_t^1)\sigma_2(t, S_t^2)}{b(t, S_t^1, S_t^2)} | \frac{S_t^1}{S_t^2}\right]
\end{aligned} \tag{4.0.16}$$

As consequence $\rho = \rho_{(a,b)}$ satisfies $\rho_{(a,b)} \in \mathcal{C}$ and

$$\rho_{(a,b)}(t, S_t^1, S_t^2) = \frac{1}{b(t, S_t^1, S_t^2)} \left( \frac{\mathbb{E}^{\mathbb{Q}^f}_{\rho(a,b)} \left[ \sigma_1^2(t, S_t^1) + \sigma_2^2(t, S_t^2) + 2\frac{a(t,S_t^1,S_t^2)}{b(t,S_t^1,S_t^2)} \sigma_1(t, S_t^1)\sigma_2(t, S_t^2) | \frac{S_t^1}{S_t^2} - \sigma_{12}^2(t, \frac{S_t^1}{S_t^2}) \right]}{\mathbb{E}^{\mathbb{Q}^f}_{\rho} [\frac{\sigma_1(t,S_t^1)\sigma_2(t,S_t^2)}{b(t,S_t^1,S_t^2)} | \frac{S_t^1}{S_t^2}]} - a(t, S_t^1, S_t^2) \right) \tag{4.0.17}$$

# 5

# THE PARTICLE METHOD FOR LOCAL CORRELATION

We have thus proved, at the last section, that any admissible correlation is of the above type. Conversely, **if a function $\rho_{(a,b)} \in \mathcal{C}$ satisfies (4.0.17), then it is an admissible correlation**. We call **(4.0.17)** the "**local in cross $a+b\rho$ representation**" of admissible correlations. Thus we can rewrite **(4.0.1)** as:

$$\begin{aligned} dS_t^1 &= (r_t^d - r_t^1)S_t^1 dt + \sigma_1(t, S_t^1)S_t^1 dW_t^1 \\ dS_t^2 &= (r_t^d - r_t^2)S_t^2 dt + \sigma_2(t, S_t^2)S_t^2 dW_t^2 \end{aligned}$$

$$d\left\langle W^1, W^2 \right\rangle_t = \frac{dt}{b(t, S_t^1, S_t^2)} \tag{5.0.1}$$

$$\left( \frac{\mathbb{E}^{\mathbb{Q}^f}_{\rho(a,b)}\left[\sigma_1^2(t,S_t^1) + \sigma_2^2(t,S_t^2) + 2\frac{a(t,S_t^1,S_t^2)}{b(t,S_t^1,S_t^2)}\sigma_1(t,S_t^1)\sigma_2(t,S_t^2)|\frac{S_t^1}{S_t^2} - \sigma_{12}^2(t,\frac{S_t^1}{S_t^2})\right]}{\mathbb{E}^{\mathbb{Q}^f}_{\rho}\left[\frac{\sigma_1(t,S_t^1)\sigma_2(t,S_t^2)}{b(t,S_t^1,S_t^2)}|\frac{S_t^1}{S_t^2}\right]} - a(t,S_t^1,S_t^2) \right) \tag{5.0.2}$$

In practice, one may try to build a solution $\rho_{(a,b)} \in \mathcal{C}$ using the **particle method**, that can be described as follows. Let $t_k$ denote a time discretization of $[0,T]$. We simulate $N$ processes $(S_t^{1,i}, S_t^{2,i})_{1\le i\le N}$ starting from $(S_0^1, S_0^2)$ at time 0 using $N$ independent Brownian motions under the domestic measure $\mathbb{Q}$ as follows:

1. Initialize k = 1 and set $\rho_{(a,b)}(t, S_t^1, S_t^2) = \frac{\sigma_1^2(0,S^1)+\sigma_1^2(0,S^1)-\sigma_{12}^2(0,\frac{S^1}{S^2})}{2\sigma_1(0,S^1)\sigma_2(0,S^2)}$ for all $t \in [t_0 = 0; t_i]$
2. Simulate $(S_t^{1,i}, S_t^{2,i})_{1\le i\le N}$ from $t_{k-1}$ to $t_k$ using a discretization scheme - say a **log-Euler scheme**
3. For all $S^{12}$ in a grid $G_{t_k}$ of cross rate values, compute

$$E_{t_k}^{num}(S^{12}) = \frac{\sum_{i=1}^N S_{t_k}^{2,i}\left(\sigma_1^2(t_k, S_{t_k}^{1,i}) + \sigma_2^2(t, S_{t_k}^{2,i}) + 2\frac{a(t,S_{t_k}^{1,i},S_{t_k}^{2,i})}{b(t,S_{t_k}^{1,i},S_{t_k}^{2,i})}\sigma_1(t,S_{t_k}^{1,i})\sigma_2(t,S_{t_k}^{2,i})\right)\delta_{t_k,N}\left(\frac{S_{t_k}^{1,i}}{S_{t_k}^{2,i}} - S^{12}\right)}{\sum_{i=1}^N S_{t_k}^{2,i}\delta_{t_k,N}\left(\frac{S_{t_k}^{1,i}}{S_{t_k}^{2,i}} - S^{12}\right)}$$

$$E_{t_k}^{den}(S^{12}) = \frac{\sum_{i=1}^N S_{t_k}^{2,i}\frac{\sigma_1^2(t_k,S_{t_k}^{1,i}\sigma_2^2(t,S_{t_k}^{2,i})}{b(t,S_{t_k}^{1,i},S_{t_k}^{2,i})}\delta_{t_k,N}\left(\frac{S_{t_k}^{1,i}}{S_{t_k}^{2,i}} - S^{12}\right)}{\sum_{i=1}^N S_{t_k}^{2,i}\delta_{t_k,N}\left(\frac{S_{t_k}^{1,i}}{S_{t_k}^{2,i}} - S^{12}\right)}$$

$$f(t_k, S^{12}) = \frac{E_{t_k}^{num}(S^{12}) - \sigma^{12}(t_k, S^{12})}{2E_{t_k}^{den}(S^{12})}$$

interpolate and extrapolate $f(t_k, .)$, for instance using cubic splines, and, for all $t \in [t_k, t_{k+1}]$, set

$$\rho_{(a,b)}(t, S^1, S^2) = \frac{1}{b(t, S^1, S^2)} \left( f\left(t_k, \frac{S^1}{S^2}\right) - a(t, S^1, S^2) \right)$$

4. Set $k := k + 1$. Iterate steps 2 and 3 up the maturity date $T$.

Where $\delta_{t_k,N}$ is an approximation of the Delta dirac function.

## 5.1 Volatility models

Having the algorithm the next step was the numerical implementation, sadly, we did not have access to real data. Nevertheless with the recommendation of our Professor we simulated our own data. For this simulation we refer to [Tankov, ], [Gatheral and Jacquier, 2014] and [Mooney, 1997].

Knowing the volatilities of our system ($\sigma_1$, $\sigma_2$ and $\sigma_{12}$) we can easily simulate $N$ paths of the FX rates ($S^1$, $S^2$ and $S^{12}$) by using $N$ independent Brownian motions and **(4.0.1)**. We took extra care in simulation the tuple $(S^1, S^2)$ because we needed to simulated correlated Brownian motions, for that purpose we used the formula :

$$\begin{aligned} dW_t^1 &= Z_1 \\ dW_t^2 &= \rho dW^1 + \sqrt{1-\rho^2} Z_2 \end{aligned}$$

with $Z_1, Z_2$ two independent centered Gaussian variables.

Next, for the choice of $\sigma_1$, $\sigma_2$ and $\sigma_{12}$, we analyze three different models that will be detailed later.

Finally for the implementation of our algorithm we took different approaches to compare the particle calibration method.

1. Constant volatility : In this model we take just $\sigma_1$, $\sigma_2$ and $\sigma_{12}$ to be constants, here we expect $\rho$ to be constant in each of the calibration methods.

2. "Exponential Volatility" with fixed $\rho$ : Here we take a more complicated model of the exponential form $\sigma(t, x) = \sigma_0(1 + e^{-(\mu x + \lambda t)})$ so that the volatilities are not constants nor $\rho$. This model is an approximation of the volatility smile. In the numerical methods we compare this model also with the model before. In this approach we fix a specific $\rho$ to be calibrated, and we calibrate this model to see the error of each calibration method.

For the calibration methods we will take those given in [Guyon, 2013] which are :

1. Local in cross correlation : $a = 0$ and $b = 1$. In this case, we can assume that the correlation itself is local in cross:

$$\rho_{(0,1)}(t, S_t^1, S_t^2) = \frac{\mathbb{E}^{\mathbb{Q}^f}_{\rho(0,1)}\left[\sigma_1^2(t, S_t^1) + \sigma_2^2(t, S_t^2)|\frac{S_t^1}{S_t^2}\right] - \sigma_{12}^2(t, \frac{S_t^1}{S_t^2})}{2\mathbb{E}^{\mathbb{Q}^f}_{\rho(0,1)}[\sigma_1(t, S_t^1)\sigma_2(t, S_t^2)|\frac{S_t^1}{S_t^2}]} \tag{5.1.1}$$

2. Local in cross volatility : $a = \sigma_1^2 + \sigma_2^2$ and $b = -2\sigma_1\sigma_2$. In this case, we can assume that the instantaneous variance of the cross rate is local in cross:

$$\rho^*(t, S_t^1, S_t^2) = \frac{\sigma_1^2(t, S_t^1) + \sigma_2^2(t, S_t^2) - \sigma_{12}^2(t, \frac{S_t^1}{S_t^2})}{2\sigma_1(t, S_t^1)\sigma_2(t, S_t^2)} \tag{5.1.2}$$

3. Local in cross covariance : $a = 0$ and $b = \sigma_1\sigma_2$. In this case we can assume that the local covariance $\rho(t, S_t^1, S_t^2)\sigma_1(t, S^1)\sigma_2(t, S^2)$ of increments of $S^1$ and $S^2$ is local in cross:

$$\rho_{(0,\sigma_1\sigma_2)}(t, S_t^1, S_t^2) = \frac{\mathbb{E}^{\mathbb{Q}^f}_{\rho(0,\sigma_1\sigma_2)}\left[\sigma_1^2(t, S_t^1) + \sigma_2^2(t, S_t^2)|\frac{S_t^1}{S_t^2}\right] - \sigma_{12}^2(t, \frac{S_t^1}{S_t^2})}{2\sigma_1(t, S_t^1)\sigma_2(t, S_t^2)} \tag{5.1.3}$$

For all those cases, if at some date $t < T$, $\rho(t, S_t^1, S_t^2) \notin [-1, 1]$ for some FX values $S^1, S^2$, then the trial is a failure and this $\rho$ is not admissible.

## 5.2 Model Simulation

For the numerical simulation of our model we took $N$ paths and wee simulated according to the log-Euler scheme :

$$\frac{dS_t^1}{S_t^1} = (r_t^d - r_t^1)\, dt + \sigma_1(t, S_t^1)\, dW_t^1 \tag{5.2.1}$$

$$\frac{dS_t^2}{S_t^2} = (r_t^d - r_t^2)\, dt + \sigma_2(t, S_t^2)\, dW_t^2 \tag{5.2.2}$$

$$\frac{dS_t^{12}}{S_t^{12}} = (r_t^2 - r_t^1)\, dt + \sigma_1\left(t, S_t^1\right) dW_t^{1,f} - \sigma_2(t, S_t^2) dW_t^{2,f} \tag{5.2.3}$$

$$dW_t^1 = Z_1 \tag{5.2.4}$$

$$dW_t^2 = \rho dW^1 + \sqrt{1 - \rho^2} Z_2 \tag{5.2.5}$$

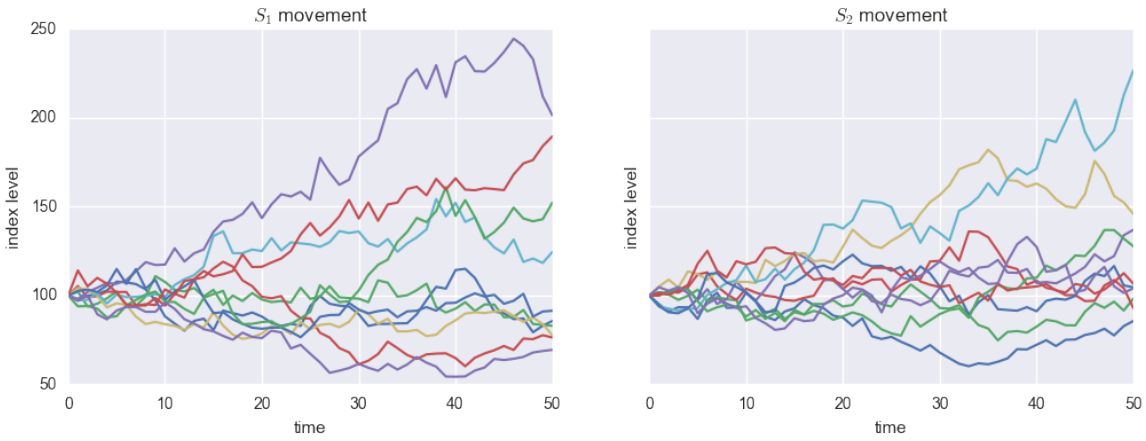



**Figure 5.1:** Sample simulations of $(S_t^1, S_t^2)$

with $Z_1, Z_2$ two independent centered Gaussian variables. In the **Figure 5.1** we can see some of the generated paths :

Next for the simulation of the volatilities, we can see in the **Figure 5.2** and **Figure 5.3** the plot of $\sigma_1, \sigma_2$ following the exponential form given before, and the $\rho$ obtained with the exponential volatility model and adjusted as $\rho = \dfrac{\sigma_1^2 + \sigma_2^2 - \sigma_{12}^2}{2\sigma_1\sigma_2}$ :

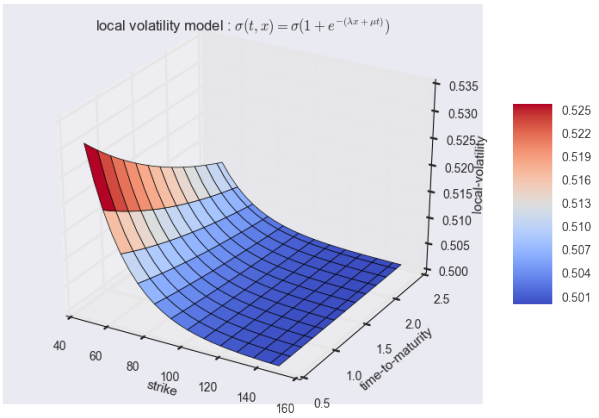



**Figure 5.2:** exponential volatility $\sigma(t,x) = \sigma_0(1 + e^{-(\mu x + \lambda t)})$

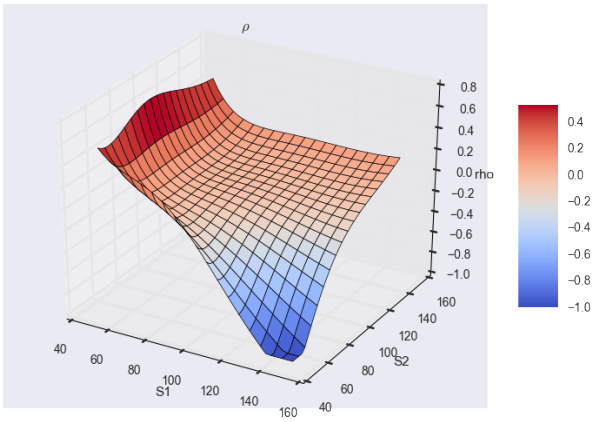



**Figure 5.3:** sample correlation

After constructing our own data system, we can finally begin the implementation of our algorithm. For that, we need first to define our Delta dirac approximation function $\delta_{t_k,N}$ and the interpolation method that will use.

For the function $\delta_{t_k,N}$ we used:

$$\delta_{t_k,N}(x) = \frac{n}{\pi((1 + (nx)^2))} \tag{5.2.6}$$

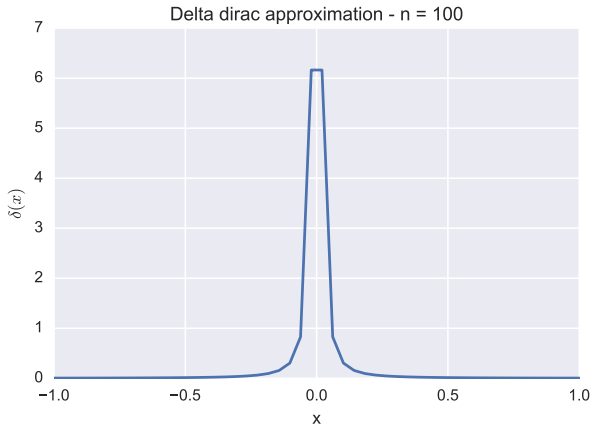



**Figure 5.4:** Delta Dirac Approximation

where $n$ is a natural constant. For our proposes, $n = 100$ gave great results, as showed bellow:

For the interpolation method, we followed the paper [Guyon, 2013] suggestion and used the cubic splines interpolation.

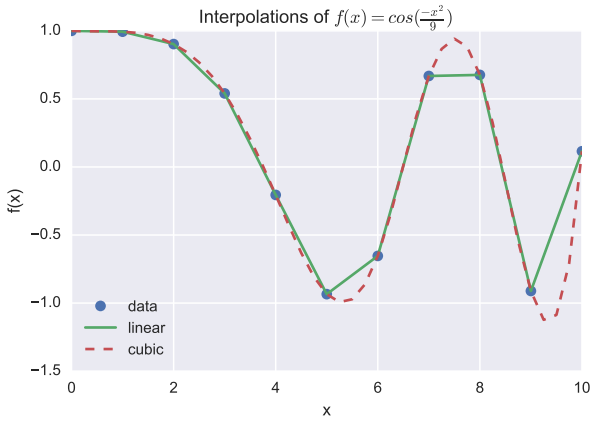



**Figure 5.5:** Linear and cubic interpolations of the function $f(x) = cos(\frac{-x^2}{9})$

This way, at our code, the function $particle\_method(S_1, S_2, S_12, \sigma_1, \sigma_2, \sigma_{12}, a, b, grilha)$ follow the steps explained at the last section and return the list of the cubic spline interpolated functions $f(t_k, .)$ as it can be seen at the appendix.

With this list, we can easily calculate $\rho_{(a,b)(t,S^1,S^2)}$ by finding first the interval $[t_k, t_{k+1}[$ in which $t$ belongs and then using the equation:

$$\rho_{(a,b)}(t, S^1, S^2) = \frac{1}{b(t, S^1, S^2)} \left( f\left(t_k, \frac{S^1}{S^2}\right) - a(t, S^1, S^2) \right)$$

With all this background knowledge and functions, we are finally able to analyze the results given by the particle method to each one of our studies cases.

# 6
# NUMERICAL EXAMPLES

Here, at this section, we expose and discuss the results obtained.

## 6.1 Constant Volatility

We noticed the method worked, as expected, really well to constant volatilities and returned also almost constant $\rho$. The results observed were obtained for:

$$\begin{aligned} r^1 &= 0.5 \\ r^2 &= 0.8 \\ r^d &= 0.6 \\ \sigma_1 &= 0.25 \\ \sigma_2 &= 0.25 \\ \rho_{theoretic}(t, S_t^1, S_t^2) &= -1 \\ T &= 2.5 \\ \Delta t &= \frac{1}{100} \\ N &= 5000 \end{aligned}$$

where $T$ is the maturity, $N$ is the number of Monte Carlo paths used and $\Delta t$ is the time step. We can see the results of this method in the **Figure 6.7**, **Figure 6.8** and **Figure 6.9**.

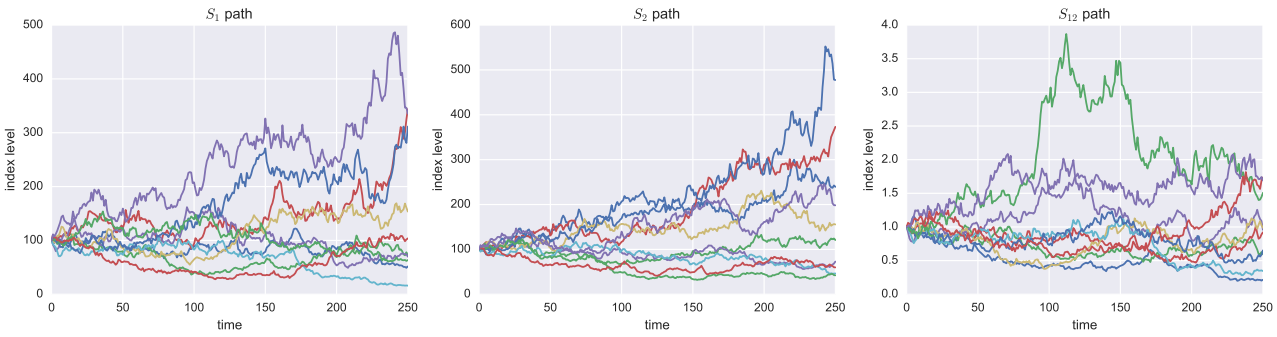



**Figure 6.6:** FX rates paths

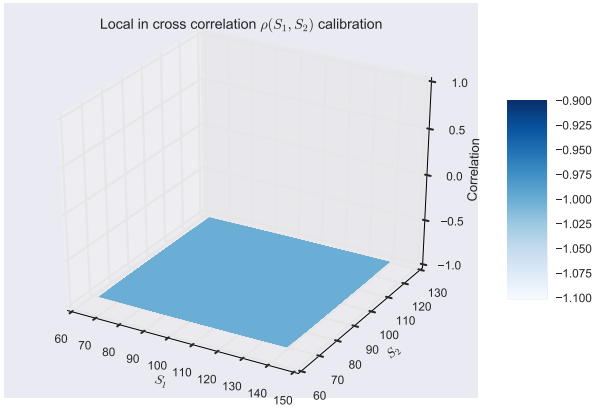



**Figure 6.7:** Local in cross correlation model

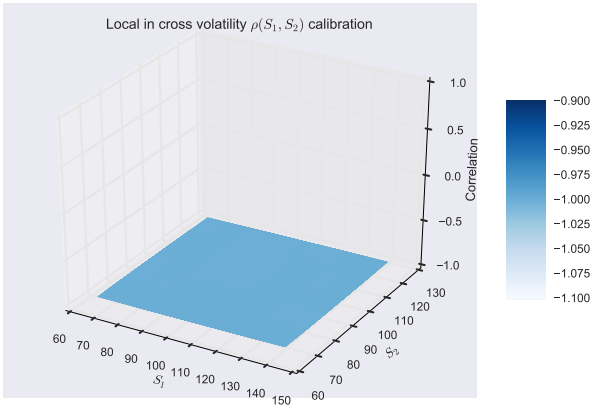



**Figure 6.8:** Local in cross volatility model

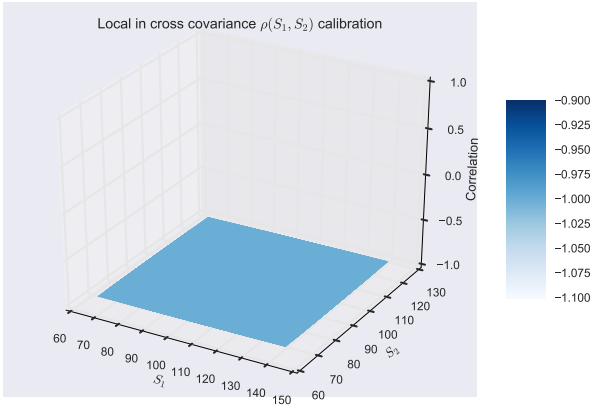



**Figure 6.9:** Local in cross covariance model

From the results we see that the calibration works and calibrates well the value $\rho = -1$ in all the elections of local correlation parameters $a$ and $b$. We see that all methods work because we have considered a very simple model with constant correlation.

## 6.2 Exponential Volatility model with fixed $\rho$

In this model we take the same parameters as before, but this time:

$$\begin{aligned}
\sigma_1(t,x) &= 0.5(1+e^{-0.05x-0.5t}) \\
\sigma_2(t,x) &= 0.4(1+e^{-0.1x-0.45t}) \\
\sigma_{12}(t,x) &= 0.3(1+e^{-0.2x-0.9t}) \\
\rho_{theoretic}(t,S_t^1,S_t^2) &= \frac{\sigma_1^2+\sigma_2^2-\sigma_{12}^2}{2\sigma_1\sigma_2} \\
Grid &= linspace(1,4,500)
\end{aligned}$$

We can see the results of this method in the **Figure 6.15**, **Figure 6.16** and **Figure 6.17**.

In this implementation we have considered non constant volatilities shown in **Figure 6.10**, **Figure 6.11**, **Figure 6.12** and the theoretical correlation that we want to calibrate in **Figure 6.13**.

If we compare this last plot with the results of **Figure 6.15**, **Figure 6.16** and **Figure 6.17** we see that although the method explodes at certain parts, nevertheless we approximate the correlation in between 0.5 and 0.4. We also state that when the calibration gave a correlation outside the interval [-1,1] (which is possible if we are close in the dirac function to zero) we truncated this result to -1 or 1 in each case to avoid that the plot lost it's meaning.

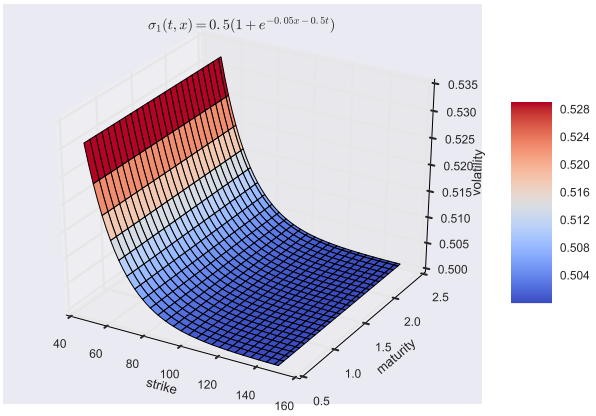



**Figure 6.10:** Volatility of $S_1$

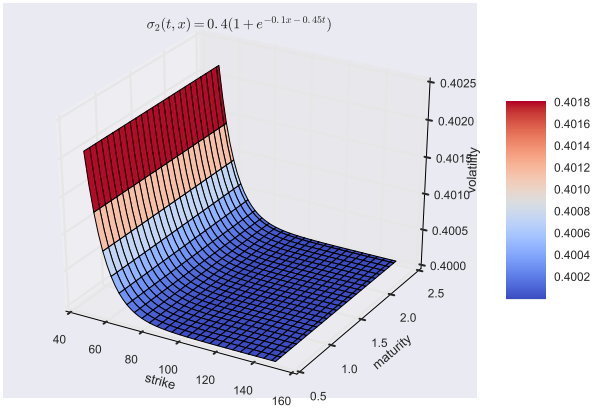



**Figure 6.11:** Volatility of $S_1$

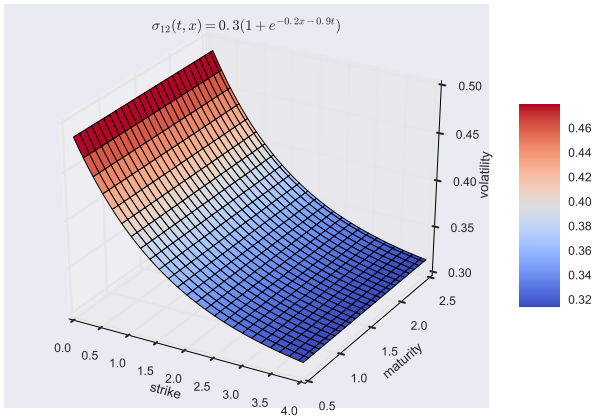



**Figure 6.12:** Volatility of $S_{12}$

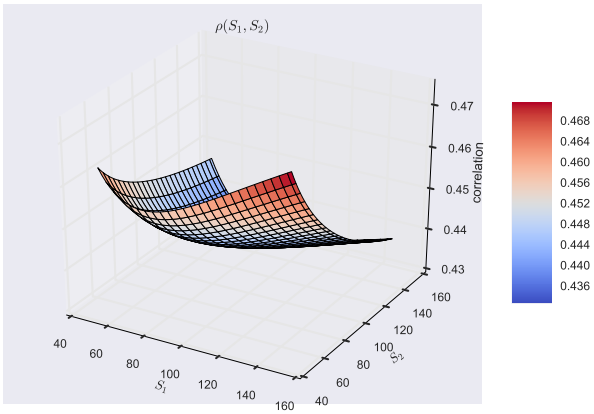



**Figure 6.13:** Local correlation

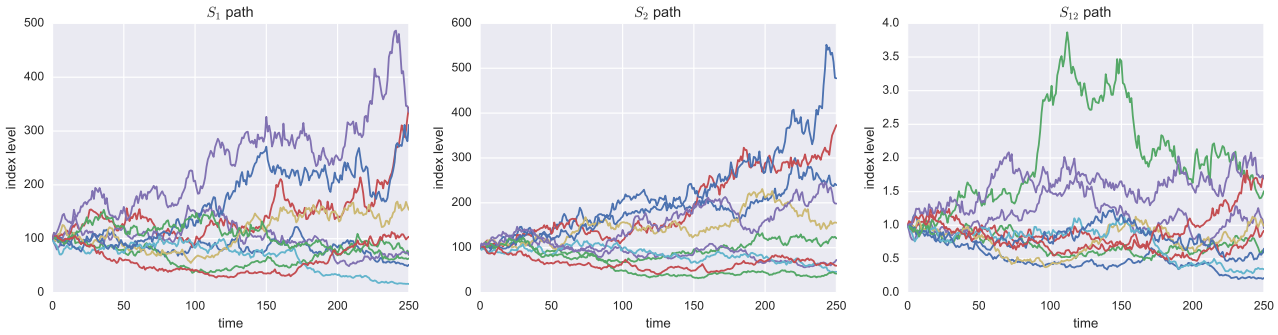



**Figure 6.14:** FX rates paths

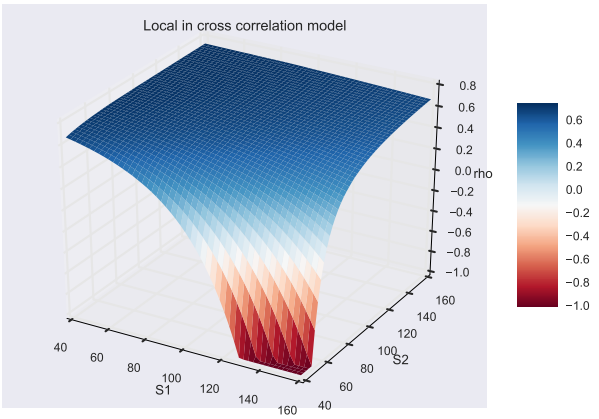



**Figure 6.15:** Local in cross correlation model

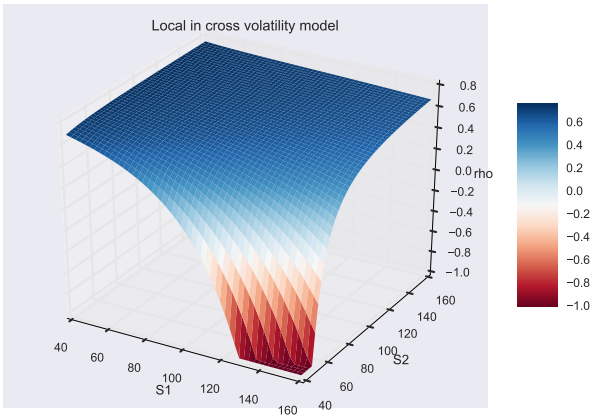



**Figure 6.16:** Local in cross volatility model

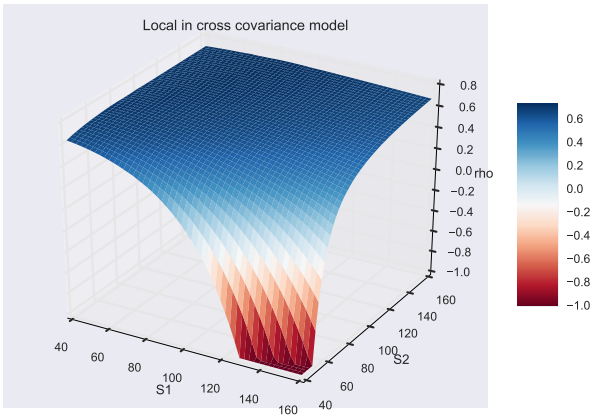



**Figure 6.17:** Local in cross covariance model

# 7
# CONCLUSION

With the general theoretical analysis on volatility regimes and models on exchange rates discussed at the **section 2** and **section 3** of this project, we were able to derivate at **section 4** the equation **(4.0.14)** that shows us how to calibrate this three exchange rate model.

Besides, we analyzed at **section 5** how to build a solution using the particle method and exposed at **section 6** the results obtained.This last section verified the efficiency and consistency of the method and that the model considered to the pair of functions (a,b) has a direct impact over the quality of the system calibration.

With all this study we were thus able to develop a solid academic background about the volatility regimes, the models on exchange rates and the calibration of an exchange rate triangle models and theory.

Furthermore, it was an extraordinary opportunity to improve our coding and algorithms skills, since all the computational analyses and implementation is an extra challenge of the project.

This project, then, furnishes a good mathematical overview of exchange rates triangle modeling and calibration. Applying the concepts of stochastic calculus, numerical calculus and general applied mathematics fundamentals, we were able to unveil this non-trivial topic using a practical approach. Albeit the analysis presented here provides a good idea of exchange rates triangle modeling and calibration, we highlight that this is a huge and complex domain.

To finish, we emphasize that the initial idea of our project was to implement and study the particle method for local correlation for simulated and real data. Sadly, we did not have access to real data and our study finished to be developed with our simulated systems. Nevertheless, we are confident that with the appropriate data and minors changes in the code we would be able to study real cases scenarios with consistency and credibility.

END

# 8
# BIBLIOGRAPHY


[Dawson and Vaillancourt, 1995] Dawson, D. and Vaillancourt, J. (1995). Stochastic mckean-vlasov equations. *NoDEA: Nonlinear Differential Equations and Applications*, 2(2):199–229.

[Dupire et al., 1994] Dupire, B. et al. (1994). Pricing with a smile. *Risk*, 7(1):18–20.

[Gatheral, 2011] Gatheral, J. (2011). *The volatility surface: a practitioner's guide*, volume 357. John Wiley & Sons.

[Gatheral and Jacquier, 2014] Gatheral, J. and Jacquier, A. (2014). Arbitrage-free svi volatility surfaces. *Quantitative Finance*, 14(1):59–71.

[Guyon, 2013] Guyon, J. (2013). A new class of local correlation models. *Available at SSRN 2283419*.

[Guyon and Henry-Labordere, 2010] Guyon, J. and Henry-Labordere, P. (2010). From spot volatilities to implied volatilities. *Available at SSRN 1663878*.

[Guyon and Henry-Labordere, 2011] Guyon, J. and Henry-Labordere, P. (2011). The smile calibration problem solved.

[Guyon and Henry-Labordère, 2013] Guyon, J. and Henry-Labordère, P. (2013). *Nonlinear option pricing*. CRC Press.

[Gyöngy, 1986] Gyöngy, I. (1986). Mimicking the one-dimensional marginal distributions of processes having an itô differential. *Probability theory and related fields*, 71(4):501–516.

[Itô and McKean, 1996] Itô, K. and McKean, H. (1996). *Diffusion processes and their sample paths*. Springer.

[Langnau, 2010] Langnau, A. (2010). A dynamic model for correlation. *Risk*, 23(4):74.

[Mooney, 1997] Mooney, C. Z. (1997). *Monte carlo simulation*, volume 116. Sage Publications.

[Reghai, 2010] Reghai, A. (2010). Breaking correlation breaks. *Risk*, 23(10):92.

[Tankov, ] Tankov, P. Surface de volatilité.

[Tankov, 2006] Tankov, P. (2006). Calibration de modeles et couverture de produits derives.

# 9 APPENDIX

For the following theorems we denote

$$Z_t^{(a)} = exp\left(\int_0^t a\phi_s dW_s - \frac{a^2}{2}\int_0^t |\phi_s|^2 ds\right), 0 \leq t \leq T. \tag{9.0.1}$$

**Novikov's criterion.** Suppose that

$$E[e^{\frac{1}{2}\int_0^T |\phi_s|^2 ds}] < \infty \tag{9.0.2}$$

Then $E[Z_t] = 1$ and the process $Z_t, 0 \leq t \leq T$ is a martingale.

**Girsanov's Theorem.** Suppose that $E[Z_t] = 1$. Then the process

$$\tilde{W} := W_t - \int_0^t \phi_s ds, t \leq T \tag{9.0.3}$$

is a Brownian motion under the probability $\mathbb{Q} = Z_t\mathbb{P}$ on $\mathcal{F}_T$

**Gyongy's Theorem.** Suppose

$$dX(t) = \mu_t dt + \sigma_t dW(t), \tag{9.0.4}$$

where $\mu_t$ and $\sigma_t$ are adapted random processes and $W(t)$ is a Brownian motion. Define (non-random) functions

$$\hat{\mu}(t,x) = \mathbb{E}[\mu_t | X(t) = x], \tag{9.0.5}$$

$$\hat{\sigma}(t,x) = \left(\mathbb{E}[\sigma_t^2 | X(t) = x]\right)^{\frac{1}{2}} \tag{9.0.6}$$

Then there exists a solution of the stochastic differential equation

$$dY(t) = \hat{\mu}(t, Y(t))dt + \hat{\sigma}(t, Y(t))dW(t) \tag{9.0.7}$$

with initial condition $Y(0) = X(0)$ such that at each fixed time t, $Y(t) \overset{\mathcal{D}}{=} X(t)$.

**Garman-Kohlhagen model.** We consider a model with a domestic and a foreign country with domestic and foreign interest rates $r^d$ and $r^f$ and let $S$ be the exchange rate of the foreign under the domestic country. Then the following holds:

$$dS_t/S_t = (r^f - r^d)\, dt + \sigma\, dW_t \tag{9.0.8}$$

**Proposition.** Let us consider the following dynamics for an asset S, where the volatility $a_t$, the interest rate $r_t$, and the repo $q_t$, inclusive of the dividend yield, are all stochastic processes:

$$\frac{dS_t}{S_t} = (r_t - q_t)dt + a_t dW_t \tag{9.0.9}$$

This model is exactly calibrated to the market smile of $S$ if and only if

$$\frac{\mathbb{E}[D_{0t}a_t^2|S_t = K]}{\mathbb{E}[D_{0t}|S_t = K]} = \sigma_D up(t,K)^2 - \frac{\mathbb{E}[D_{0t}(r_t - q_t - (r_t^0 - q_t^0))\mathbf{1}_{S_t>K}]}{\frac{1}{2}K\partial_K^2\mathcal{C}(t,K)} + \frac{\mathbb{E}[D_{0t}(q_t - q_t^0)(S_t - K)^+]}{\frac{1}{2}K^2\partial_K^2\mathcal{C}(t,K)} \tag{9.0.10}$$

for all $(t, K)$, where $D_{0t} = exp\left(-\int_0^t r_s ds\right)$ is the discount factor, $r_t^0$ and $q_t^0$ are deterministic rates and repos, and

$$\sigma_{Dup}(t,K)^2 = \frac{\partial_t\mathcal{C}(t,K) + (r_t^0 - q_t^0)K\partial_K\mathcal{C}(t,K) + q_t^0\mathcal{C}(t,K)}{\frac{1}{2}K^2\partial_K^2\mathcal{C}(t,K)} \tag{9.0.11}$$

with $\mathcal{C}(t, K)$ market price of the call option on $S$ with strike $K$ and maturity $t$.

**REMARK** The deterministic rate $r_t^0$ is typically taken to be equal to $-\partial_t \ln P_{0t}$, with $P_{0t}$ the price at time 0 of a zero-coupon bond maturing at time $t$. Then one can infer a deterministic repo rate $q_{0t}$ from the forward price $f_0^t$:

$$q_t^0 = r_t^0 - \partial_t \ln \frac{f_0^t}{S_0} \tag{9.0.12}$$

# 10
# JUPYTER CODE

# FX_calibration

March 19, 2017

## 1 FX Triangle Calibration

*Lucas Furquim, Felipe Garcia*

```
In [1]: # Imports
        import numpy as np
        import numpy.random as npr
        import pandas as pd
        import matplotlib.pyplot as plt
        import seaborn
        %matplotlib inline
```

Next we are simulation two GBM with initial parameters $S_0$ and $S_{0\_}$ growing at the same rates

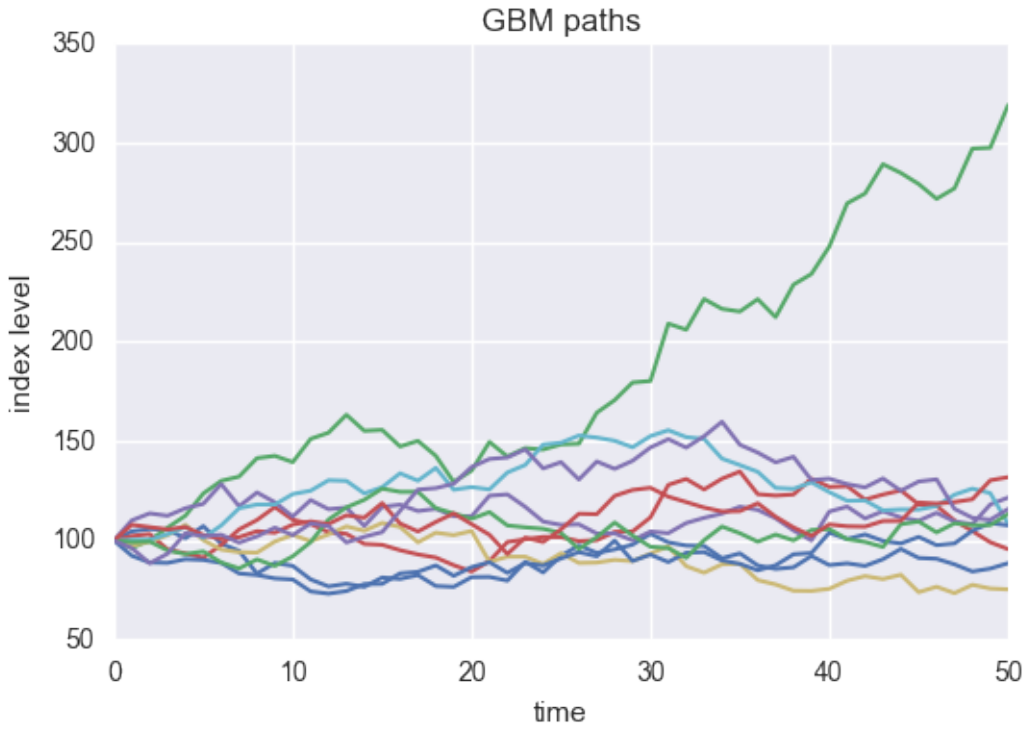
```
In [2]: S0, S0_ = 100, 100 # initial value
        r = 0.05  # constant short rate
        sigma = 0.25  # constant volatility
        T = 2.0  # time in years
        M = 50 # maturity
        I = 100  # number of random draws

        dt = T / M
        S_1 = np.zeros((M + 1, I))
        S_2 = np.zeros((M + 1, I))

        S_1[0] = S0
        S_2[0] = S0_

        for t in range(1, M + 1):
            S_1[t] = S_1[t - 1] * np.exp((r - 0.5 * sigma ** 2) * dt
                    + sigma * np.sqrt(dt) * npr.standard_normal(I))
            S_2[t] = S_2[t - 1] * np.exp((r - 0.5 * sigma ** 2) * dt
                    + sigma * np.sqrt(dt) * npr.standard_normal(I))
```

Plotting the GBM

```
In [3]: plt.plot(S_1[:, :10], lw=1.5)
        plt.xlabel('time')
```

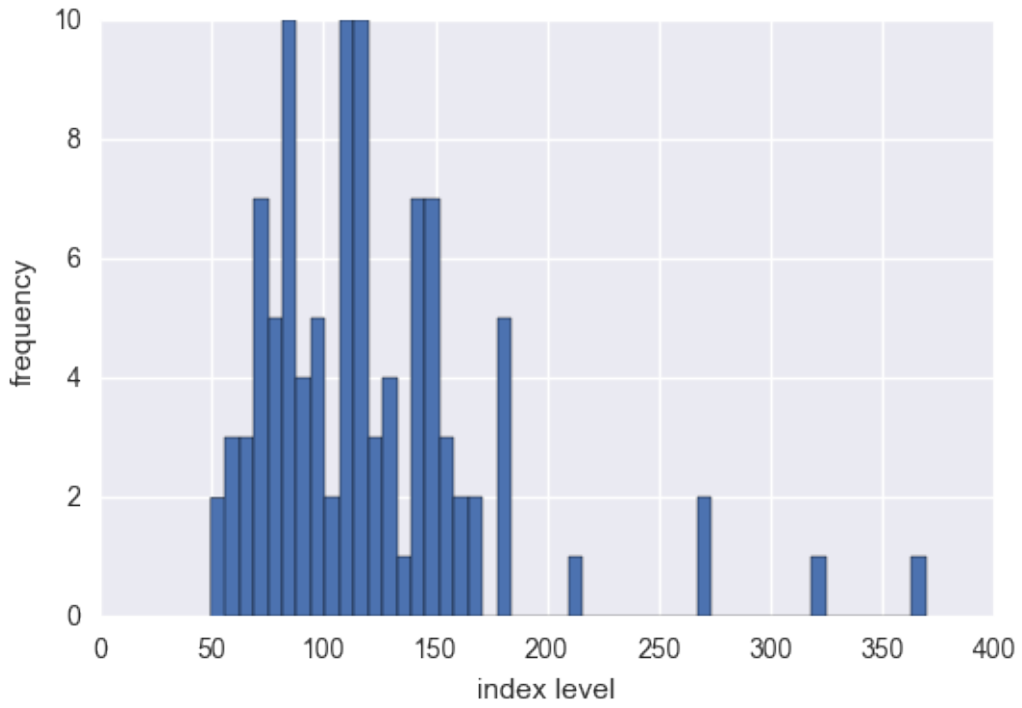
```
plt.ylabel('index level')
plt.grid(True)
plt.title("GBM paths")
# tag: gbm_dt_paths
# title: Simulated geometric Brownian motion paths
# size: 60
```

```
Out[3]: <matplotlib.text.Text at 0x116d245d0>
```




Simulating geometric Brownian motion at maturity

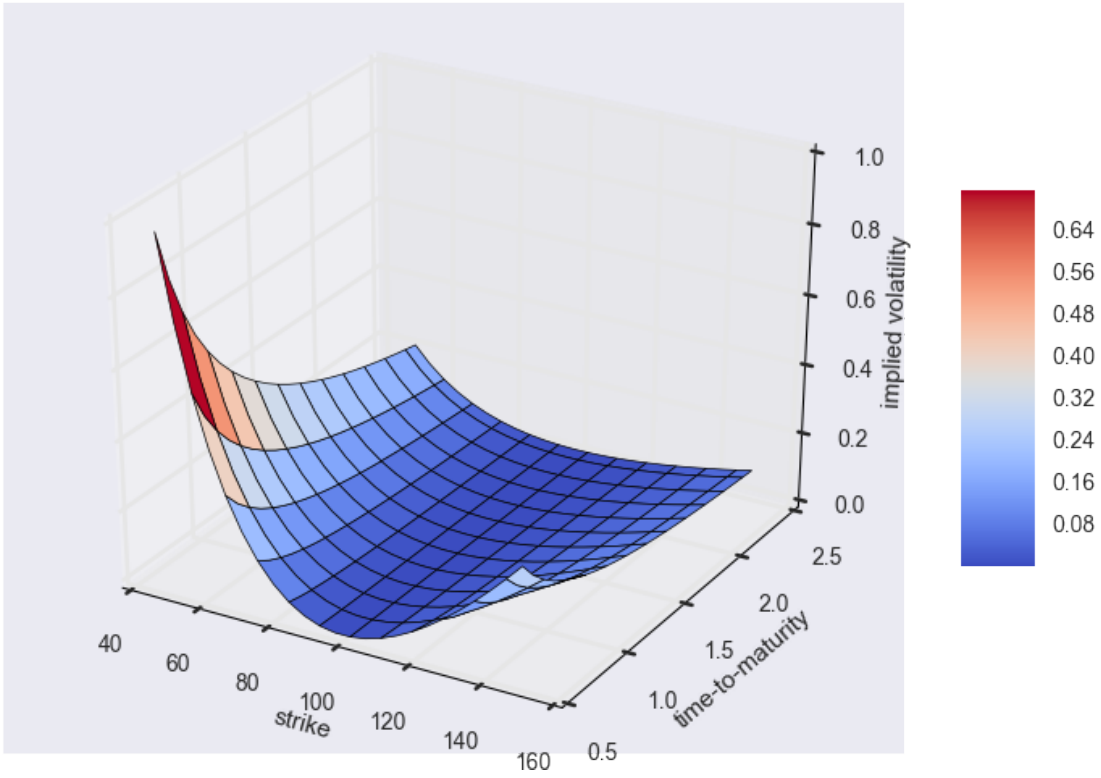
```
In [4]: plt.hist(S_1[-1], bins=50)
plt.xlabel('index level')
plt.ylabel('frequency')
plt.grid(True)
# tag: gbm_dt_hist
# title: Simulated geometric Brownian motion at maturity
# size: 60
```




```
In [5]: strike = np.linspace(50, 150, 24)
        ttm = np.linspace(0.5, 2.5, 24)
        strike, ttm = np.meshgrid(strike, ttm)


        iv = (strike - 100) ** 2 / (100 * strike) / ttm
          # generate fake implied volatilities
```

```
In [6]: from mpl_toolkits.mplot3d import Axes3D

        fig = plt.figure(figsize=(9, 6))
        ax = fig.gca(projection='3d')

        surf = ax.plot_surface(strike, ttm, iv, rstride=2, cstride=2,
                               cmap=plt.cm.coolwarm, linewidth=0.5,
                               antialiased=True)

        ax.set_xlabel('strike')
        ax.set_ylabel('time-to-maturity')
        ax.set_zlabel('implied volatility')

        fig.colorbar(surf, shrink=0.5, aspect=5)
        # tag: matplotlib_17
        # title: 3d surface plot for (fake) implied volatilities
        # size: 70
```

```
Out[6]: <matplotlib.colorbar.Colorbar at 0x117b15fd0>
```




## 2 Simulation of FX

Here we simulate paths for the FX triangle $S_1$, $S_2$ and $S_{12}$

```
In [7]: # Data simulation
        S0, S0_, S0__ = 100.0, 100.0, 1.0 # initial value
        r1, r2, rd = 0.5, 0.8, 0.6  # constant short rate
        sigma1, sigma2 = 0.25, 0.25  # constant volatility
        T = 2.0  # time in years
        M = 50 # maturity
        I = 100  # number of random draws

        dt = T / M
        S_1 = np.zeros((M + 1, I))
        S_2 = np.zeros((M + 1, I))

        S_1[0] = S0
        S_2[0] = S0_

        for t in range(1, M + 1):
```

```
        S_1[t] = S_1[t - 1] * np.exp((rd - r1 ) * dt
                + sigma1 * np.sqrt(dt) * npr.standard_normal(I))
        S_2[t] = S_2[t - 1] * np.exp((rd - r2) * dt
                + sigma2 * np.sqrt(dt) * npr.standard_normal(I))

    S_12 = S_1 / S_2
```

Here we plot the paths obtained via the Log-Euler scheme for $S_1, S_2$ and $S_{12}$ :

```
In [8]: f, (ax1, ax2, ax3) = plt.subplots(1, 3, sharey=False)
        f.set_size_inches(16, 4)

        ax1.plot(S_1[:, :10], lw=1.5)
        ax1.set_xlabel('time')
        ax1.set_ylabel('index level')
        ax1.set_title('$S_1$ movement')
        ax1.grid(True)

        ax2.plot(S_2[:, :10], lw=1.5)
        ax2.set_xlabel('time')
        ax2.set_ylabel('index level')
        ax2.set_title('$S_2$ movement')
        ax2.grid(True)

        ax3.plot(S_12[:, :10], lw=1.5)
        ax3.set_xlabel('time')
        ax3.set_ylabel('index level')
        ax3.set_title('$S_{12}$ movement')
        ax3.grid(True)
```


## 2.1 Implementation of The Algorithm

Here we implement the particle method. We first define our delta dirac kernel: $f_n(x) = \frac{n}{\pi(1+(nx)^2)}$ for $n = 100$

```
In [9]: from scipy.interpolate import interp1d
        # Approximation of Delta dirac function
        def delta(x):
            n = 100
            return n / (1 + (n * x) ** 2) / np.pi
```

### 2.1.1 Local in cross volatility model

We are taking the case for constant volatility $\sigma\_1 = \sigma\_2\$and(\rho = \frac{\sigma_1^2 + \sigma_2^2 - \sigma_{12}^2}{2\sigma_1\sigma_2}$ as described in the paper. $a = \sigma_1^2 + \sigma_2^2$ and $b = -2\sigma_1\sigma_2$

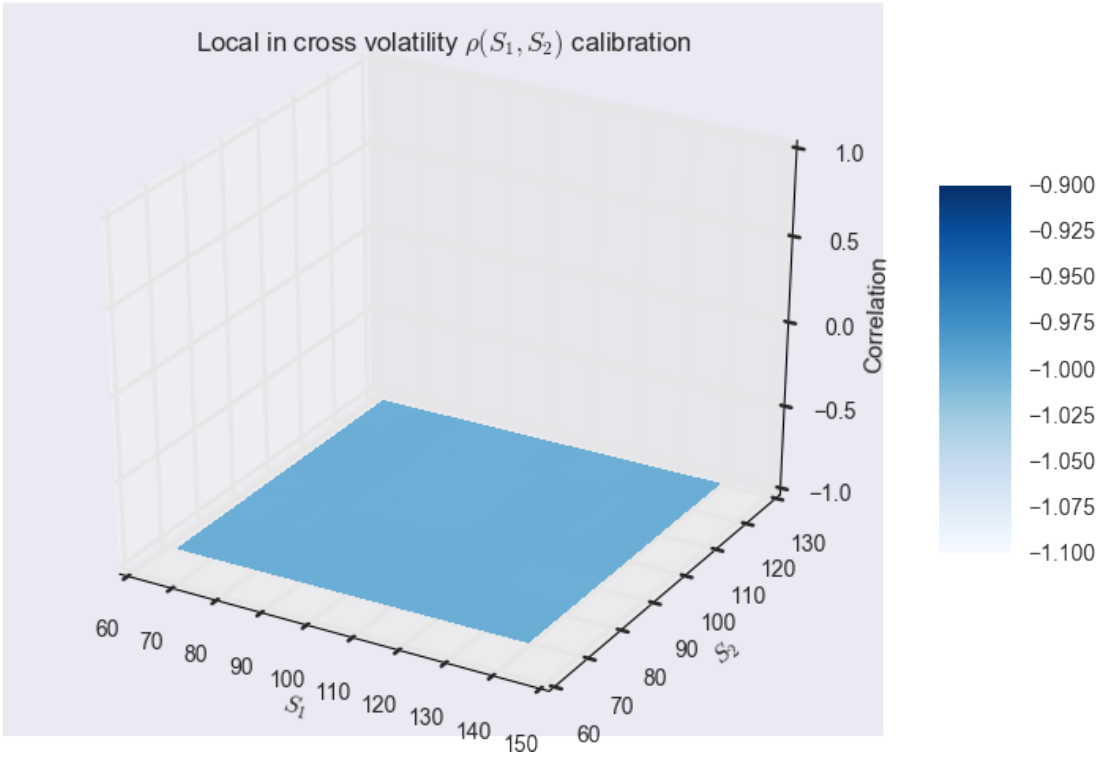
```
In [10]: # Definition of Parameters
         k = 1
         sigma_12 = 0.5
         sigma1, sigma2 = 0.25, 0.25
         a = sigma1 ** 2 + sigma2 ** 2
         b = -2 * sigma1 * sigma2

         N = 10 # grid
         grilha = np.linspace(0,100, N)
```

```
In [11]: # interpolation formula
         def interp(p, f):
             x = p
             y = f
             f2 = interp1d(x, y, kind='cubic')
             return f2
```

```
In [12]: # Particle Method
         def calibrate(S_1, S_2, sigma1, sigma2, a, b, grilha):
             ind = 0
             rho = (sigma1**2 + sigma2**2 - sigma_12**2) / (2 * sigma1 * sigma2)
             Enum = np.zeros((N, M+1))
             Eden = np.zeros((N, M+1))
             f = np.zeros((N, M+1))
             for p in grilha:
                 Enum[ind] = np.sum((S_2*(sigma1**2 + sigma2**2 + 2*(a/b)*sigma1*si
                 Eden[ind] = np.sum((S_2*(sigma1*sigma2)/b) * delta(S_1/S_2 - p), a
                 f[ind] = (Enum[ind] - sigma_12**2)/(2*Eden[ind])
                 ind+=1

             f2 = []
             for t in range(1, M + 1):
                 f2.append(interp(grilha, f[:, t]))
             return f2
```

```
In [13]: flist = calibrate(S_1, S_2, sigma1, sigma2, a, b, grilha)
         t = 10 # time where to calculate rho
```

```
s1, s2 = np.meshgrid(S_1[t], S_2[t])
z = (1/b) * (flist[t](s1/s2) - a)

fig = plt.figure(figsize=(9, 6))
ax = fig.gca(projection='3d')

surf = ax.plot_surface(s1, s2, z, rstride=1, cstride=1,
                       cmap=plt.cm.Blues, linewidth=0,
                       antialiased=False)

ax.set_xlabel(r'$S_1$')
ax.set_ylabel(r'$S_2$')
ax.set_zlabel(r'Correlation')
ax.set_zlim([-1,1])
ax.set_title(r'Local in cross volatility $\rho(S_1,S_2)$ calibration')

fig.colorbar(surf, shrink=0.5, aspect=5)
fig.savefig("images/constant_local_in_cross_vol.pdf", bbox_inches='tight')
```

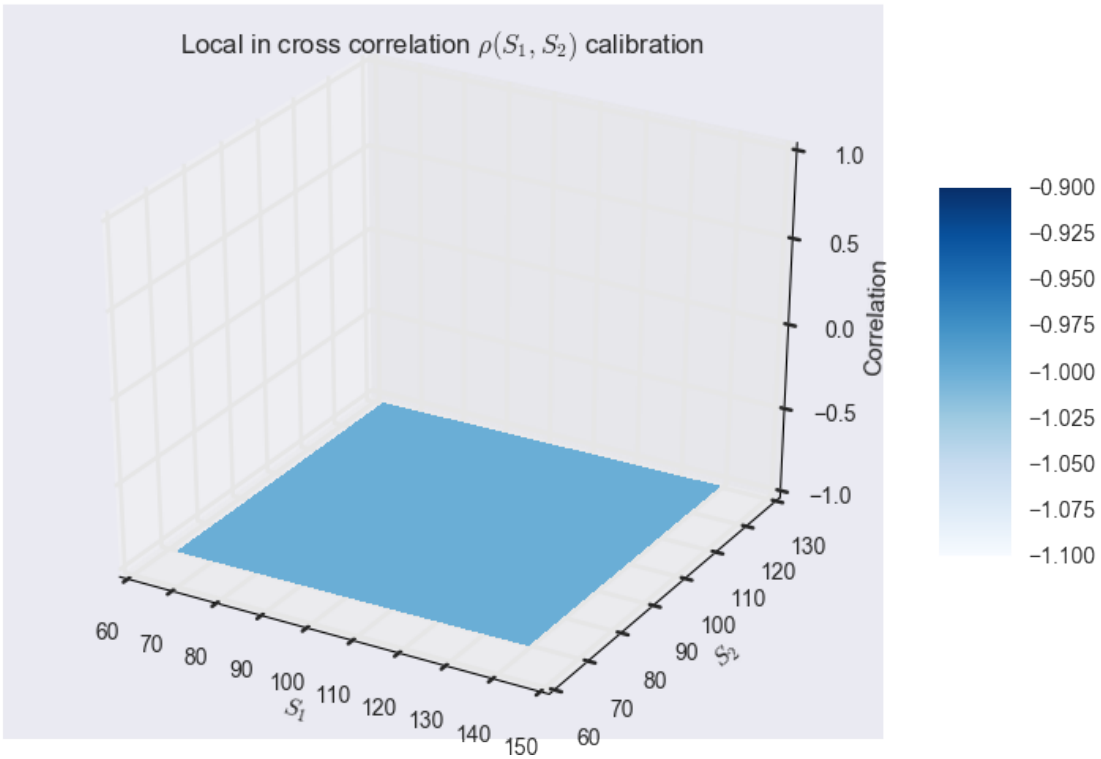



We see a costant correlation as expected with value $\rho = \dfrac{\sigma_1^2 + \sigma_2^2 - \sigma_{12}^2}{2\sigma_1\sigma_2} = -1$ as expected.

### 2.1.2 Local in cross correlation model

We are taking $a = 0$ and $b = 1$

```
In [14]: a,b = 0, 1
         flist = calibrate(S_1, S_2, sigma1, sigma2, a, b, grilha)
         t = 10
         s1, s2 = np.meshgrid(S_1[t], S_2[t])
         z = (1/b) * (flist[t](s1/s2) - a)

         fig = plt.figure(figsize=(9, 6))
         ax = fig.gca(projection='3d')

         surf = ax.plot_surface(s1, s2, z, rstride=1, cstride=1,
                                cmap=plt.cm.Blues, linewidth=0,
                                antialiased=False)

         ax.set_xlabel(r'$S_1$')
         ax.set_ylabel(r'$S_2$')
         ax.set_zlabel(r'Correlation')
         ax.set_zlim([-1,1])
         ax.set_title(r'Local in cross correlation $\rho(S_1,S_2)$ calibration')

         fig.colorbar(surf, shrink=0.5, aspect=5)
         fig.savefig("images/constant_local_in_cross_corr.pdf", bbox_inches='tight'
```

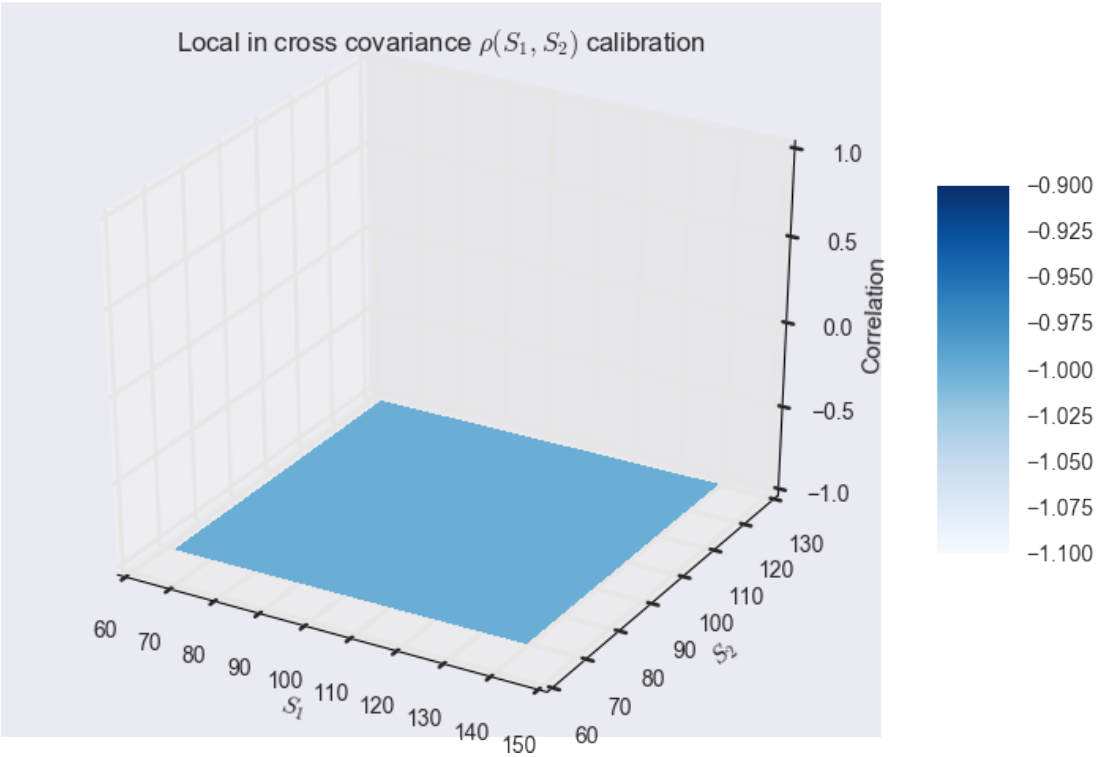



We see a costant correlation as expected with value $\rho = \dfrac{\sigma_1^2 + \sigma_2^2 - \sigma_{12}^2}{2\sigma_1\sigma_2} = -1$ as expected.

### 2.1.3 Local in cross covariance model

We are taking $a = 0$ and $b = \sigma_1\sigma_2$

```
In [15]: a, b = 0, sigma1 * sigma2
         flist = calibrate(S_1, S_2, sigma1, sigma2, a, b, grilha)
         t = 10
         s1, s2 = np.meshgrid(S_1[t], S_2[t])
         z = (1/b) * (flist[t](s1/s2) - a)

         fig = plt.figure(figsize=(9, 6))
         ax = fig.gca(projection='3d')

         surf = ax.plot_surface(s1, s2, z, rstride=1, cstride=1,
                                cmap=plt.cm.Blues, linewidth=0,
                                antialiased=False)

         ax.set_xlabel(r'$S_1$')
         ax.set_ylabel(r'$S_2$')
         ax.set_zlabel(r'Correlation')
         ax.set_zlim([-1,1])
         ax.set_title(r'Local in cross covariance $\rho(S_1,S_2)$ calibration')

         fig.colorbar(surf, shrink=0.5, aspect=5)
         fig.savefig("images/constant_local_in_cross_cov.pdf", bbox_inches='tight')
```



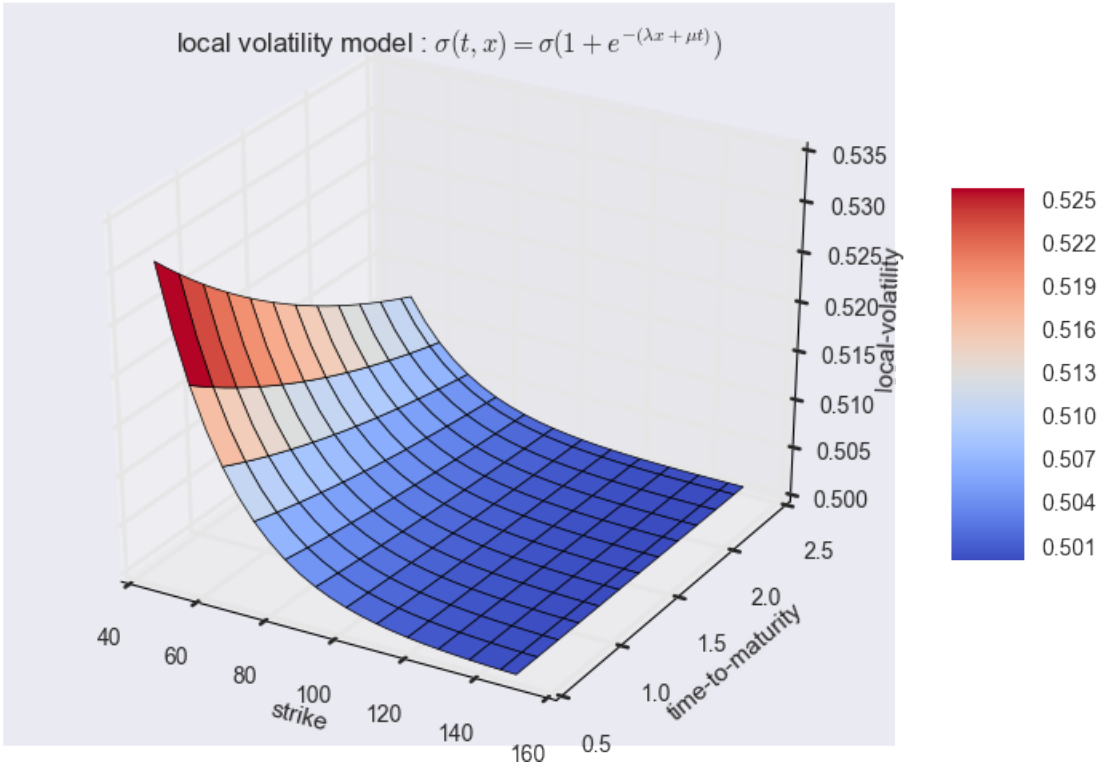
We see a costant correlation as expected with value $\rho = \frac{\sigma_1^2 + \sigma_2^2 - \sigma_{12}^2}{2\sigma_1\sigma_2} = -1$ as expected.

## 3 Local Volatility Model

Here we discuss the model with exponential volatilities of the form:

$$\sigma(t, x) = \sigma(1 + e^{-(\lambda x + \mu x)}).$$

```
In [16]: strike = np.linspace(50, 150, 24)
         ttm = np.linspace(0.5, 2.5, 24)
         strike, ttm = np.meshgrid(strike, ttm)


         iv = 0.50 * (1 + np.exp(- 0.05 * strike - 0.5 * ttm))
           # generate fake implied volatilities

         from mpl_toolkits.mplot3d import Axes3D

         fig = plt.figure(figsize=(9, 6))
         ax = fig.gca(projection='3d')

         surf = ax.plot_surface(strike, ttm, iv, rstride=2, cstride=2,
                                cmap=plt.cm.coolwarm, linewidth=0.5,
                                antialiased=True)

         ax.set_xlabel('strike')
         ax.set_ylabel('time-to-maturity')
         ax.set_zlabel('local-volatility')
         ax.set_title(r'local volatility model : $\sigma(t,x) = \sigma (1 + e^{-(\l

         fig.colorbar(surf, shrink=0.5, aspect=5)
         fig.savefig("images/exponential_volatility.pdf", bbox_inches='tight')
         # tag: matplotlib_17
         # title: 3d surface plot for (fake) implied volatilities
         # size: 70
```




For the Triangle model :

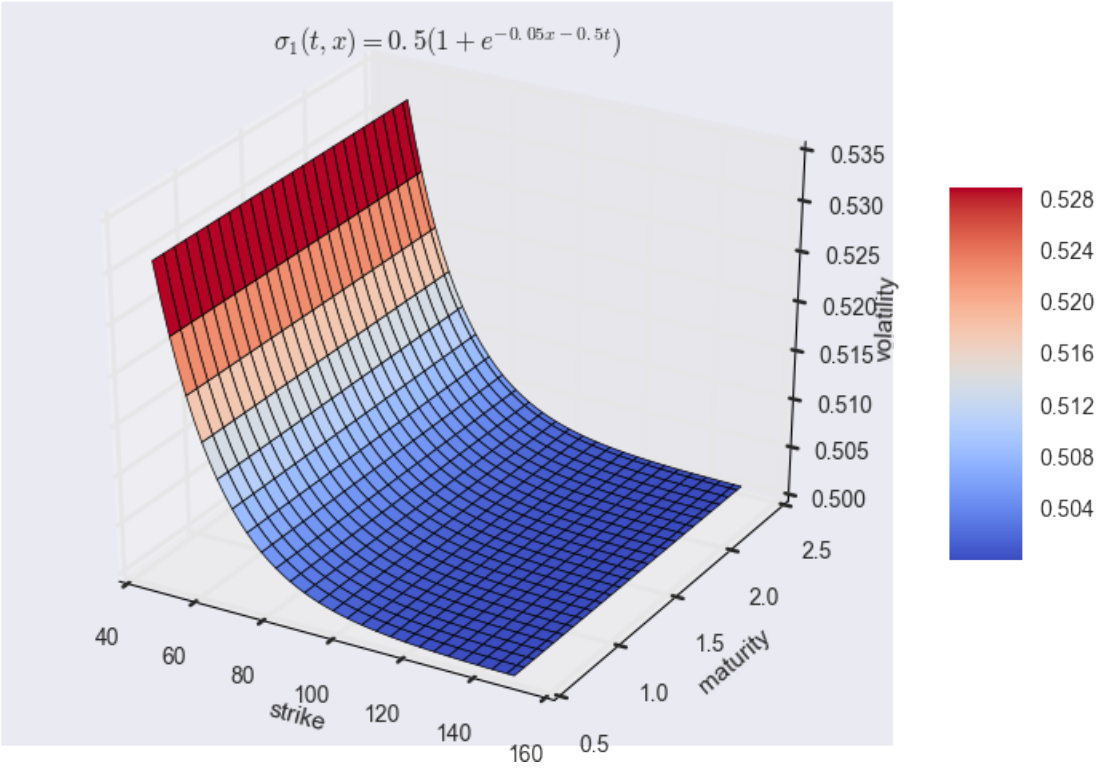
$$\begin{aligned} dS_t^1 &= (r_t^d - r_t^1)S_t^1\, dt + \sigma_1(t, S_t^1)S_t^1\, dW_t^1 \\ dS_t^2 &= (r_t^d - r_t^2)S_t^2\, dt + \sigma_2(t, S_t^2)S_t^2\, dW_t^2 \\ d\left\langle W^1, W^2 \right\rangle_t &= \rho(t, S_t^1, S_t^2)\, dt \end{aligned} \tag{1}$$

We start by simulating $S_1, S_2$ and $S_{12}$ We will now plot the functions $\sigma_1, sigma_2, \sigma_{12}$ and $\rho$ that we will use to compare with the calibration method

### 3.1 Plot for $\sigma_1(t,x)$

```
In [17]: fig = plt.figure(figsize=(9, 6))
         ax = fig.gca(projection='3d')

         def sigma1(t,x):
             aux = 0.50 * (1 + np.exp(- 0.05 * x - 0.5 * t))
             return np.minimum(np.maximum(aux, -1), 1)

         x = np.linspace(50, 150, 50)
         t = np.linspace(0.5, 2.5, 50)
         s1, s2 = np.meshgrid(x, t)
```

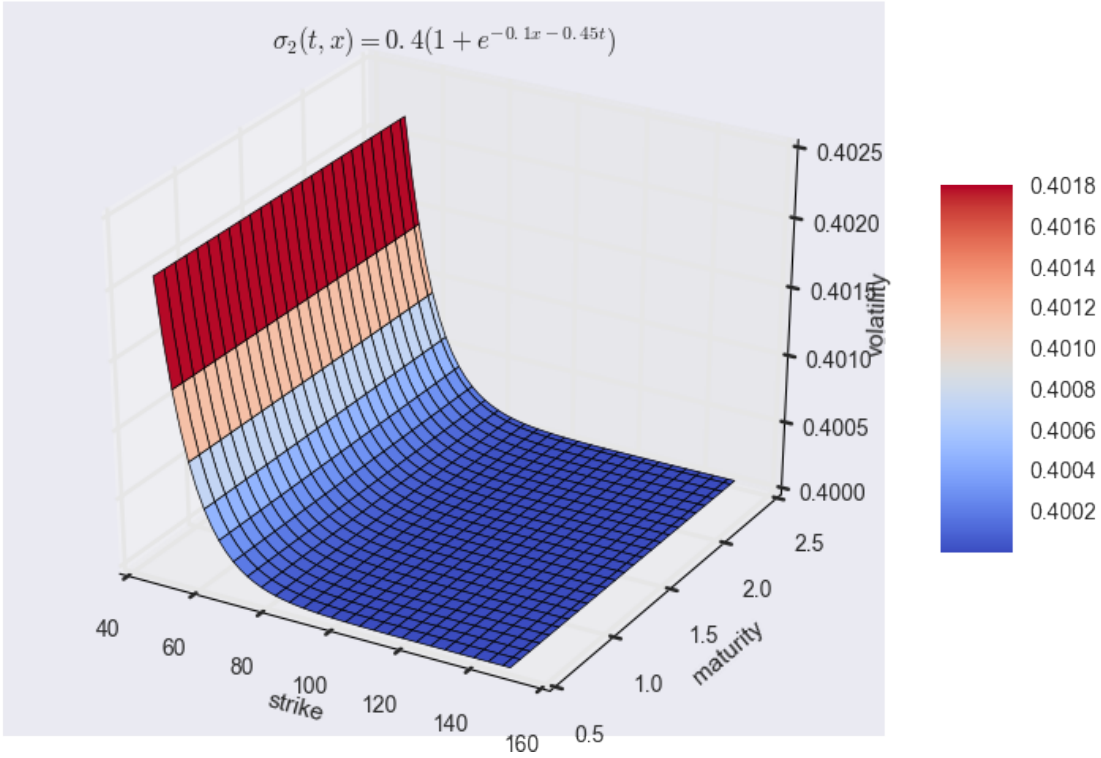
```
    z = sigma1(t, x)

    surf = ax.plot_surface(s1, s2, z, rstride=2, cstride=2,
                           cmap=plt.cm.coolwarm, linewidth=0.5,
                           antialiased=True)

    ax.set_title(r'$\sigma_1(t,x) =  0.5 (1 + e^{- 0.05  x - 0.5  t})$')
    ax.set_xlabel('strike')
    ax.set_ylabel('maturity')
    ax.set_zlabel(r'volatility')

    fig.colorbar(surf, shrink=0.5, aspect=5)
    fig.savefig("images/sigma1_exp.pdf", bbox_inches='tight')
```




## 3.2 Plot for $\sigma_2(t,x)$

In [18]:
```
    fig = plt.figure(figsize=(9, 6))
    ax = fig.gca(projection='3d')

    def sigma2(t,x):
        aux = 0.40 * (1 + np.exp(- 0.1 * x - 0.45 * t))
        return np.minimum(np.maximum(aux, -1), 1)
```

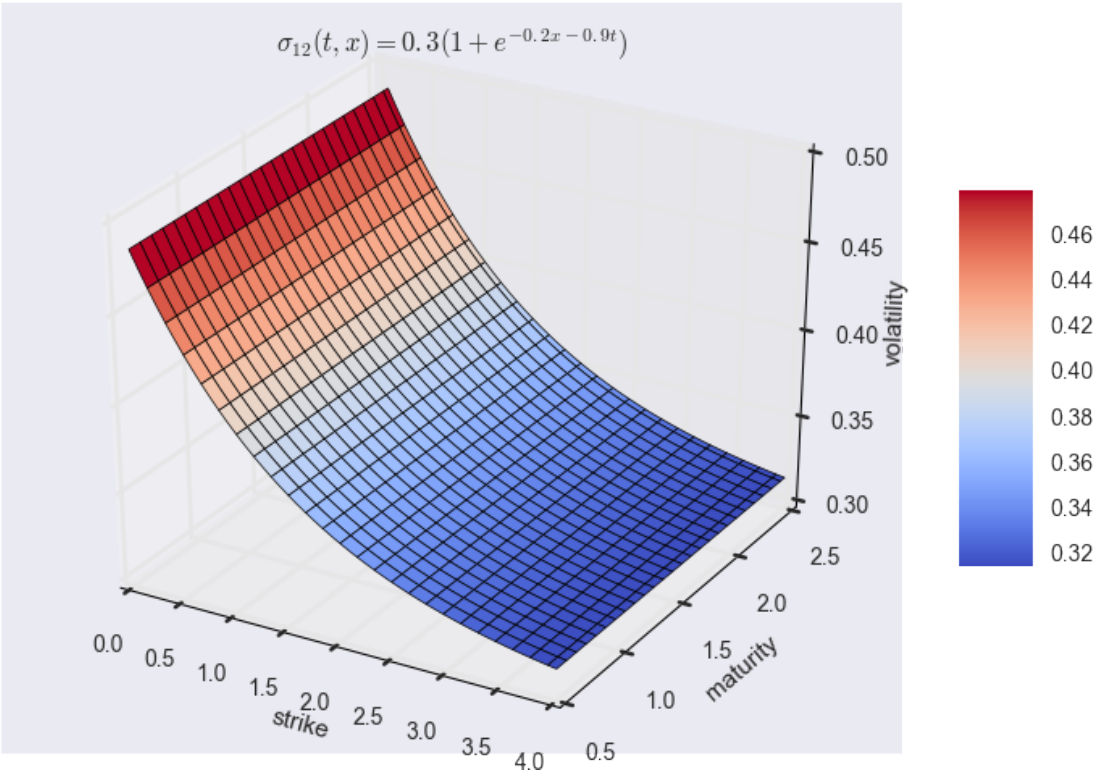
```
x = np.linspace(50, 150, 50)
t = np.linspace(0.5, 2.5, 50)
s1, s2 = np.meshgrid(x, t)
z = sigma2(t, x)

surf = ax.plot_surface(s1, s2, z, rstride=2, cstride=2,
                       cmap=plt.cm.coolwarm, linewidth=0.5,
                       antialiased=True)

ax.set_title(r'$\sigma_2(t,x) =  0.4 (1 + e^{- 0.1  x - 0.45  t})$')
ax.set_xlabel('strike')
ax.set_ylabel('maturity')
ax.set_zlabel(r'volatility')

fig.colorbar(surf, shrink=0.5, aspect=5)
fig.savefig("images/sigma2_exp.pdf", bbox_inches='tight')
```




### 3.3 Plot for $\sigma_{12}(t,x)$

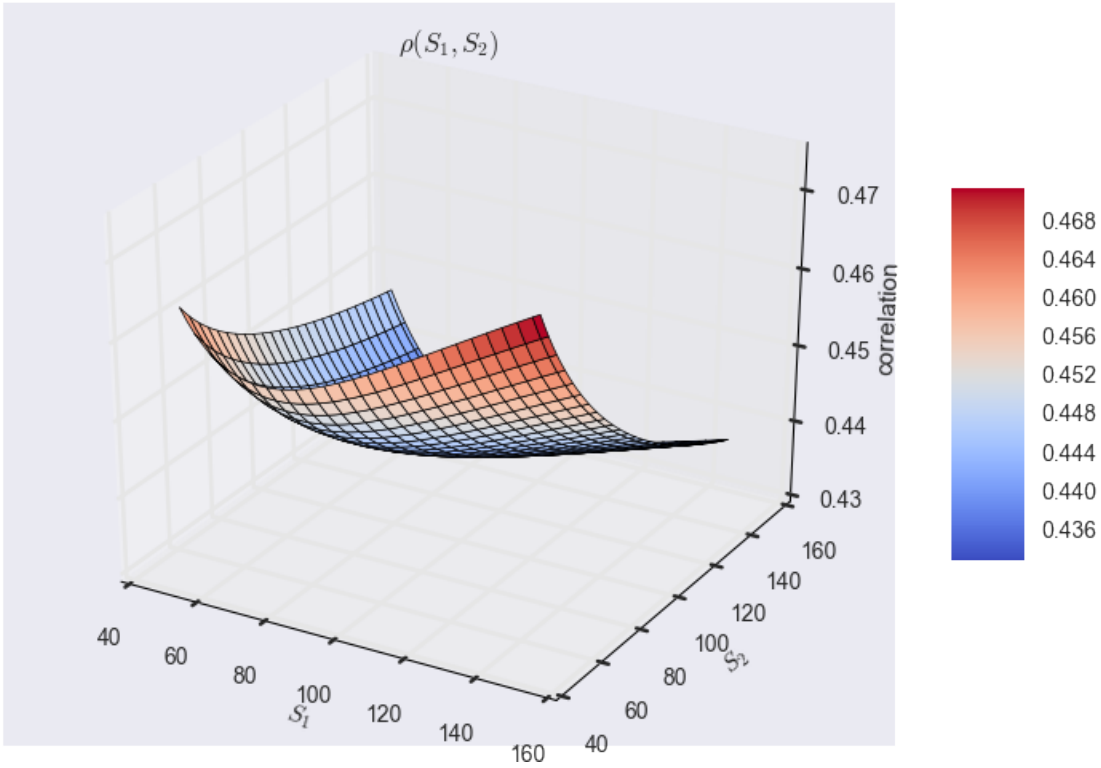
In [19]:
```
fig = plt.figure(figsize=(9, 6))
ax = fig.gca(projection='3d')
```

```
def sigma12(t,x):
    aux = 0.30 * (1 + np.exp(- 0.2 * x - 0.9 * t))
    return np.minimum(np.maximum(aux, -1), 1)

x = np.linspace(0.1, 4, 50)
t = np.linspace(0.5, 2.5, 50)
s1, s2 = np.meshgrid(x, t)
z = sigma12(t, x)

surf = ax.plot_surface(s1, s2, z, rstride=2, cstride=2,
                       cmap=plt.cm.coolwarm, linewidth=0.5,
                       antialiased=True)

ax.set_title(r'$\sigma_{12}(t,x) =  0.3 (1 + e^{- 0.2  x - 0.9  t})$')
ax.set_xlabel('strike')
ax.set_ylabel('maturity')
ax.set_zlabel(r'volatility')

fig.colorbar(surf, shrink=0.5, aspect=5)
fig.savefig("images/sigma12_exp.pdf", bbox_inches='tight')
```



### 3.4 Plot for $\rho(S_1, S_2)$

```
In [20]: fig = plt.figure(figsize=(9, 6))
         ax = fig.gca(projection='3d')

         def ro(t,x1,x2):
             aux = (sigma1(t,x1)**2 + sigma2(t,x2)**2 - sigma12(t,x1/x2)**2) / (2 *
             return np.minimum(np.maximum(aux, -1), 1)

         t = 20
         ttm = (M - t) * T / M
         x = np.linspace(50, 150, 50)
         y = np.linspace(50, 150, 50)
         s1, s2 = np.meshgrid(x, y)
         z = ro(ttm, s1, s2)

         surf = ax.plot_surface(s1, s2, z, rstride=2, cstride=2,
                                cmap=plt.cm.coolwarm, linewidth=0.5,
                                antialiased=True)

         ax.set_title(r'$\rho(S_1, S_2)$')
         ax.set_xlabel(r'$S_1$')
         ax.set_ylabel(r'$S_2$')
         ax.set_zlabel('correlation')

         fig.colorbar(surf, shrink=0.5, aspect=5)
         fig.savefig("images/rho_exp.pdf", bbox_inches='tight')
```



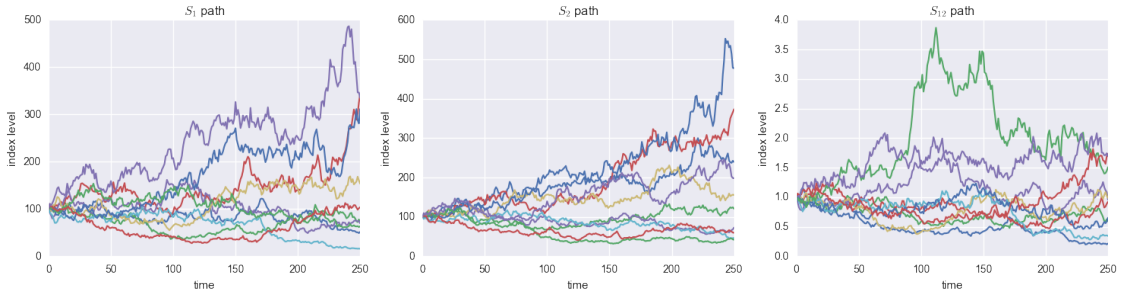

### 3.5 Path Simulation: $S_1^t, S_2^t, S_{12}^t$

```
In [21]: # Data simulation
         S0, S0_, S0__ = 100.0, 100.0, 1.0 # initial value
         r1, r2, rd = 0.5, 0.4, 0.6  # constant short rate

         T = 2.5  # time in years
         M = 250 # maturity
         I = 5000  # number of random draws

         dt = T / M
         S_1 = np.zeros((M + 1, I))
         S_2 = np.zeros((M + 1, I))
         S_12 = np.zeros((M + 1, I))

         S_1[0] = S0
         S_2[0] = S0_
         S_12[0] = S0__

         for t in range(1, M + 1):
             ttm = (M - t) * T / M
             rh = ro(ttm, S_1[t - 1], S_2[t - 1])
             W1 = npr.standard_normal(I)
```

```
        W2 = rh * W1 + np.sqrt(1-rh**2) * npr.standard_normal(I)

        S_1[t] = S_1[t - 1] * np.exp((rd-r1) * dt + sigma1(ttm, S_1[t - 1]) *
        S_2[t] = S_2[t - 1] * np.exp((rd-r2) * dt + sigma2(ttm, S_2[t - 1]) *
        S_12[t] = S_12[t - 1] * np.exp((r2-r1) * dt + sigma1(ttm, S_1[t - 1])
```

### 3.6 Plot of the trajectories

```
In [22]: fig, (ax1, ax2, ax3) = plt.subplots(1, 3, sharey=False)
         fig.set_size_inches(18, 4)

         ax1.plot(S_1[:, :10], lw=1.5)
         ax1.set_xlabel('time')
         ax1.set_ylabel('index level')
         ax1.set_title('$S_1$ path')
         ax1.grid(True)

         ax2.plot(S_2[:, :10], lw=1.5)
         ax2.set_xlabel('time')
         ax2.set_ylabel('index level')
         ax2.set_title('$S_2$ path')
         ax2.grid(True)

         ax3.plot(S_12[:, :10], lw=1.5)
         ax3.set_xlabel('time')
         ax3.set_ylabel('index level')
         ax3.set_title('$S_{12}$ path')
         ax3.grid(True)
         fig.savefig("images/exp_paths.pdf", bbox_inches='tight')
```




### 3.7 Particle Method Implementation

Here we improve our old algorithm to apply it to non constant volatilities

```
In [23]: # Algorithm
         N_g = 500 # grid
```

```
        grilha = np.linspace(0.01, 4, N_g)

        def particle_method(S_1, S_2, S_12, sigma1, sigma2, sigma12, a, b, grilha)
            ind = 0
            Enum = np.zeros((N_g, M+1))
            Eden = np.zeros((N_g, M+1))
            f = np.zeros((N_g, M+1))
            for p in grilha:
                ttm = (M - ind) * T / M
                Enum[ind] = np.sum((S_2*(sigma1(ttm, S_1)**2 + \
                                         sigma2(ttm, S_2)**2 + 2*(a(ttm, S_1, S_2)
                                         *sigma1(ttm, S_1)*sigma2(ttm, S_2)))  \
                                   * delta(S_1/S_2 - p), axis = 1) / np.sum(S_2 *
                Eden[ind] = np.sum((S_2*(sigma1(ttm, S_1)*sigma2(ttm, S_2))/b(ttm,
                                   delta(S_1/S_2 - p), axis = 1)/np.sum(S_2 * delt
                f[ind] = (Enum[ind] - sigma12(ttm, p)**2 * np.ones(M+1))/(2*Eden[i
                ind+=1

            f2 = []
            for time in range(1, M + 1):
                f2.append(interp(grilha, f[:, time]))
            return f2
```

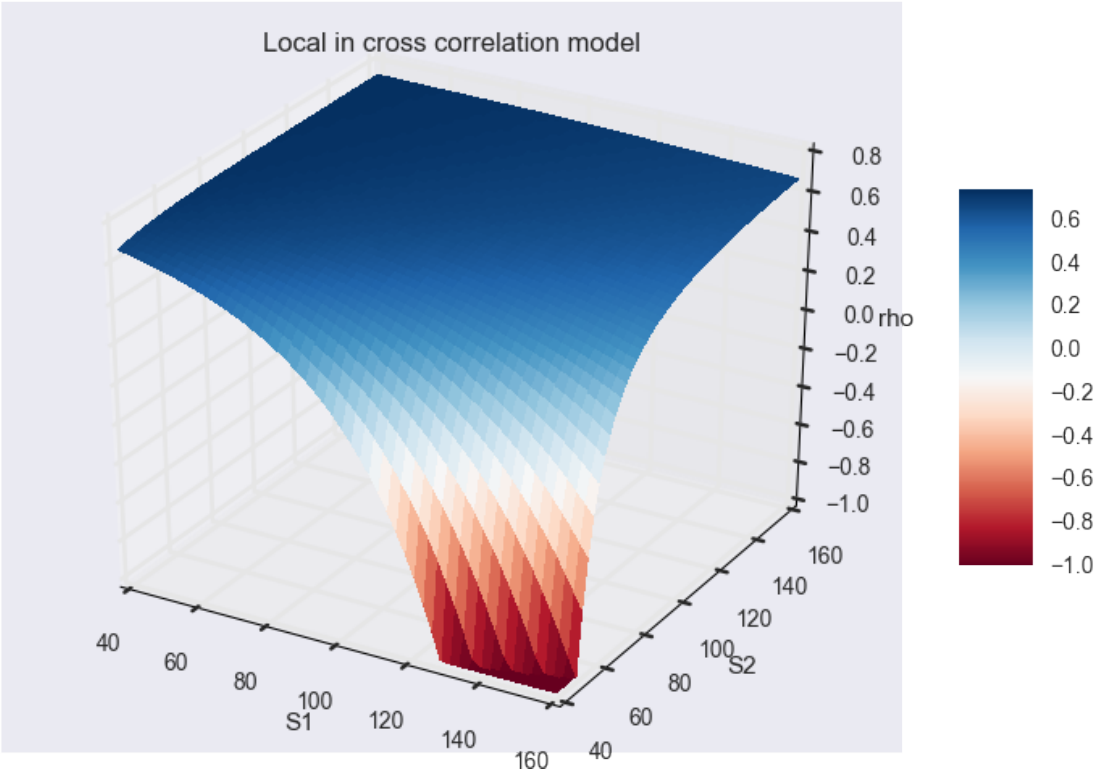
In [24]:
```
        # Auxiliary variables
        t = 20
        ttm = (M - t) * T / M

        x = np.linspace(40, 160, 50)
        y = np.linspace(40, 160, 50)
        s1, s2 = np.meshgrid(x, y)
```

### 3.8 Model local in cross correlation

We take $a = 0$ and $b = 1$

In [25]:
```
        def a(t,x1,x2):
            return 0
        def b(t,x1,x2):
            return 1

        flist = particle_method(S_1, S_2, S_12, sigma1, sigma2, sigma12, a, b, gri

        z = (1 / b(ttm, s1, s2)) * (flist[t](s1 / s2) - a(ttm, s1, s2))
        z = np.minimum(np.maximum(z, -1), 1)

        fig = plt.figure(figsize=(9, 6))
        ax = fig.gca(projection='3d')
```

```
surf = ax.plot_surface(s1, s2, z, rstride=1, cstride=1,
                       cmap=plt.cm.RdBu, linewidth=0,
                       antialiased=False)
ax.set_title(r'Local in cross correlation model')
ax.set_xlabel('S1')
ax.set_ylabel('S2')
ax.set_zlabel('rho')

fig.colorbar(surf, shrink=0.5, aspect=5)
fig.savefig("images/exp_local_corr.pdf", bbox_inches='tight')
```

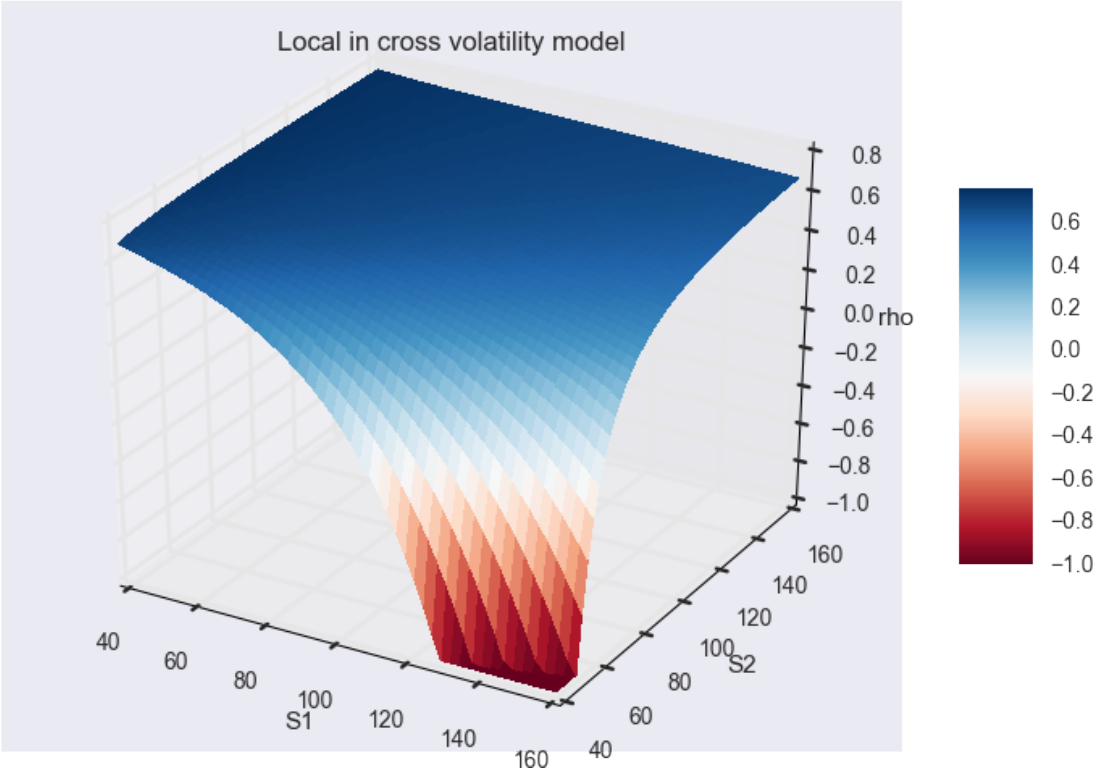



### 3.9 Model local in cross volatility

We take $a = \sigma_1^2 + \sigma_2^2$ and $b = -2\sigma_1\sigma_2$

```
In [26]: def a(t,x1,x2):
             return sigma1(t,x1) ** 2 + sigma2(t,x2)**2
         def b(t,x1,x2):
             return - 2 * sigma1(t,x1) * sigma2(t,x2)

         flist = particle_method(S_1, S_2, S_12, sigma1, sigma2, sigma12, a, b, gri

         z = (1/b(ttm, s1, s2)) * (flist[t](s1/s2) - a(ttm, s1, s2))
```

```
z = np.minimum(np.maximum(z, -1), 1)

fig = plt.figure(figsize=(9, 6))
ax = fig.gca(projection='3d')

surf = ax.plot_surface(s1, s2, z, rstride=1, cstride=1,
                       cmap=plt.cm.RdBu, linewidth=0,
                       antialiased=False)
ax.set_title(r'Local in cross volatility model')
ax.set_xlabel('S1')
ax.set_ylabel('S2')
ax.set_zlabel('rho')

fig.colorbar(surf, shrink=0.5, aspect=5)
fig.savefig("images/exp_local_vol.pdf", bbox_inches='tight')
```

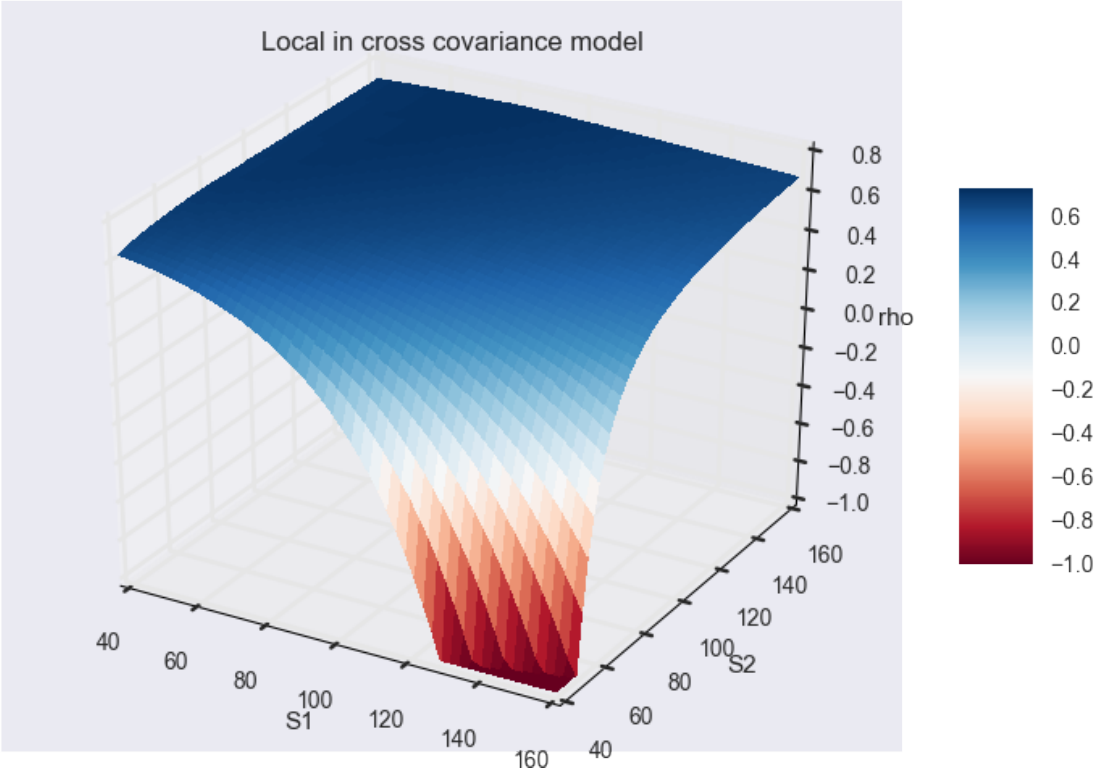



### 3.10 Model local in cross covariance

We take $a = 0$ and $b = \sigma_1\sigma_2$

```
In [27]: def a(t,x1,x2):
             return 0
         def b(t,x1,x2):
```

```
    return sigma1(t,x1) * sigma2(t,x2)

flist = particle_method(S_1, S_2, S_12, sigma1, sigma2, sigma12, a, b, gri

z = (1/b(ttm, s1, s2)) * (flist[t](s1/s2) - a(ttm, s1, s2))
z = np.minimum(np.maximum(z,-1),1)

fig = plt.figure(figsize=(9, 6))
ax = fig.gca(projection='3d')

surf = ax.plot_surface(s1, s2, z, rstride=1, cstride=1,
                       cmap=plt.cm.RdBu, linewidth=0,
                       antialiased=False)
ax.set_title(r'Local in cross covariance model')
ax.set_xlabel('S1')
ax.set_ylabel('S2')
ax.set_zlabel('rho')

fig.colorbar(surf, shrink=0.5, aspect=5)
fig.savefig("images/exp_local_cov.pdf", bbox_inches='tight')
```

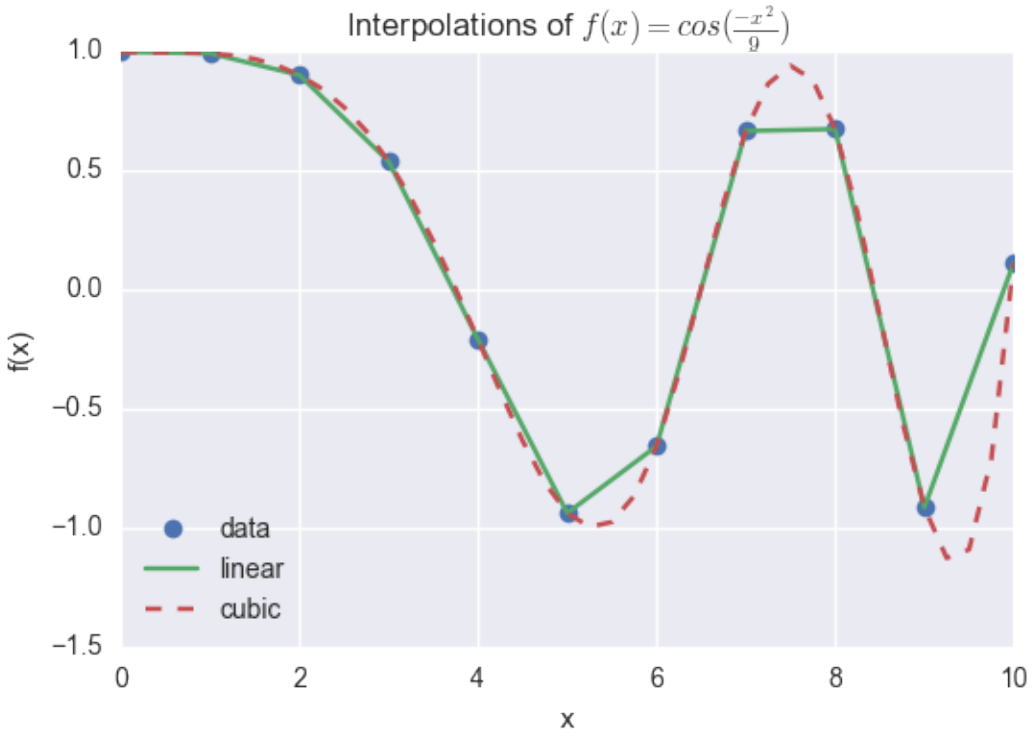


# 4 APPENDIX

## 4.1 Spline interpolation

Here we demostrate the interpolation methods

```
In [28]: from scipy.interpolate import interp1d

         x = np.linspace(0, 10, num=11, endpoint=True)
         y = np.cos(-x**2/9.0)
         f = interp1d(x, y)
         f2 = interp1d(x, y, kind='cubic')

         xnew = np.linspace(0, 10, num=41, endpoint=True)

         plt.plot(x, y, 'o', xnew, f(xnew), '-', xnew, f2(xnew), '--')
         plt.xlabel('x')
         plt.ylabel('f(x)')
         plt.legend(['data', 'linear', 'cubic'], loc='best')
         plt.title(r'Interpolations of $f(x) = cos(\frac{-x^2}{9})$')
         plt.savefig("images/spline.pdf", bbox_inches='tight')
         plt.show()
```



## 4.2 Delta Dirac approximation

Here we plot the delta dirac approximation function considered in the project

```
In [29]: x = np.linspace(-1, 1, num=50)
         plt.plot(x,delta(x),'-')
         plt.xlabel('$x$')
         plt.ylabel(r'$\delta (x)$')
         plt.title("Delta dirac approximation - n = 100")
         plt.savefig("images/dirac.pdf", bbox_inches='tight')
         plt.show()
```