Local Volatility FX Basket Option written by Monte Carlo

We consider a basket call option written on 10 FX rates given by

\[ \frac{dS^{(i)}_t}{S^{(i)}_t} = \left( r - r^{(i)}_f \right) dt + \sigma^{(i)}(S^{(i)}_t, t) dW^{(i)}_t \]

for \( i = 1, \ldots , 10 \) where \( r_d \) is the domestic rate, \( r^{(i)}_f \) is the foreign rate and \( \{W^{(i)}_t\}_{t \geq 0} \) is a correlated 10-dimensional Brownian motion with \( \langle W^{(i)} \rangle, W^{(i)}(0) = \rho^{(i,j)} \). The local volatility function \( \sigma^{(i)} \) is calibrated from market implied volatility data. The price of the option is

\[ P = e^{-r_d T} \sum_{i=1}^{10} w^{(i)} (S^{(i)}_T - K)^+ \]

where \( K > 0 \) is the strike and the \( w^{(i)} \)'s are a set of weights summing to one. \( P \) is computed by Monte Carlo simulation.

Input Parameters to the Model

Input parameters are the strike, maturity, weights, rates, correlation structure, and the 10 market implied volatility surfaces. In total there are 438 inputs and the task is to compute sensitivities of the price with respect to all.

GPU Accelerated Code: CPU → GPU → CPU

We developed a 3 stage code to price the basket option as follows:

- Stage 1: Setup (on CPU, double precision). Process implied vol surfaces into local vol surfaces
- Stage 2: Monte Carlo (GPU, single precision). Compute sample paths
- Stage 3: Payoff (CPU, double precision). Compute the payoff

The aim was to create a GPU accelerated adjoint version of this code using AD to compute the gradient.

Algorithmic Differentiation (AD) in a Nutshell

AD is a program transformation technique. It yields derivative code of up to arbitrary order. AD software tools have been developed to provide support to the application programmer. Refer to [1] for further details.

References