# No-Arbitrage Constraints

Stefano Nicoli

April 1, 2020

## 1 What do we want to do?

To clean options, we also need to know which constraints we have to apply before the download and which ones after. Here, I want to specify both of them, related to the no-arbitrage conditions. To look at them, see the financial engineering script on page 88 or the class "Week 8" of financial engineering on page 40.
In this pdf, I do not consider the other constraints (liquidity and so on).

## 2 What do we have to do before the download?

An important remark is that dividends are taken into consideration in these constraints. So, in order to avoid them, we have to download as a preliminary step the futures prices, exploiting that:

$$F_t(T)exp(-r_{t,T}(T-t)) = S_t exp(-q_{t,T}(T-t)), \tag{1}$$

where $r_{t,T}$ is the value of the interest rates at time t. In this way, we can use the left part instead of the right one.
Once we have the futures prices, we have to apply the following conditions:

1. $0 \geq \Delta C$;
2. $0 \leq \Delta P(t,T)$;
3. $0 \leq \Gamma$ for both calls and puts.

So far, we cannot set other constraints because "Conditional Statements" on Wharton only allows us to compare a variable to a real number and not to another variable.

## 3 What do we have to do after the download?

First of all, we have to set the "more complex" constraints:

1. $-exp(-r_{t,T}(T-t)) \leq \Delta C(t,T)$;

2. $exp(-r_{t,T}(T-t)) \geq \Delta P(t,T)$;

3. $(F_t(T)exp(-r_{t,T}(T-t)) - exp(-r_{t,T}(T-t))K)^+ \leq C(t,T,K)$;

4. $C(t,T,K) \leq F_t(T)exp(-r_{t,T}(T-t))$;

5. $(exp(-r_{t,T}(T-t))K - F_t(T)exp(-r_{t,T}(T-t)))^+ \leq P(t,T,K)$;

6. $P(t,T,K) \leq F_t(T)exp(-r_{t,T}(T-t))$.

Last but not least, we have to avoid the calendar spread arbitrage (the last one both in the slides and in the script). Honestly, I do not know a I direct way to avoid it. So, I thought we can try to code something like this:

# I am only pretending to write a python code, this is just to give a naive idea
# option.k is the strike of the option
# option.T is the maturity of the option
# we give to every option an "arbitrage flag" equal to zero
for day in days:
—— define a void set for strikes and a void list for options
—— for option in day:
—— —— if option.k not in set:
—— —— —— add k to strikes_set
—— —— else:
—— —— —— for j in option_list:
# I want to signal as "arbitrage options" every couple of options with the same
# strike, and in which one has a more distant maturity and is cheaper than the
# other
—— —— —— —— if option.k == j.k and (option.T-j.T)*(option.price-j.price)<0:
—— —— —— —— —— option.flag=1
—— —— —— —— —— j.flag=1
—— —— add option to option_list
—— delete every option with flag=1
—— delete option_list
—— delete strikes_set

# 4 What do I ask you?

I am almost sure about the "call constraints" since I copied them from other sources, while the "put ones" are made by me. They should be easy to make, but I do not want to make mistakes at this point. Moreover, I am sorry about the code, it's terrible, but it was the easiest and fastest way for me to share how I would deal with calendar arbitrage. If you don't understand the idea, if you have better ones or if you simply think that it doesn't work, please tell me.

Finally, I would like to know if I am taking everything into consideration or if I am missing something.