Define an method is\_path\_dependent() which returns bollean for each Option, so that for instance in the MC solver, if option.is\_path\_dependent(), then using the calc path spot prices method (or function) if it is true, otherwise just compute the payoff as option.calc\_payoff()

To differientiate between particular types of options, for instance path\_dependent or not, we can specify private attributes for each option, and then in the Solvers or Models for instance using if else statement in order to run the specific implementation.

For options that use multiple parameter values such as multiple underlyings for basket or rainbow, or multiple maturities for knock-in options, we may call the constructor of the base class Option for other paramters such as r and sigma, and setting to 0 the ones that will be replaced with multiple values. And then, in the constructor taking a vector of double values for instance. We could also mask the get\_S, set\_S, and get\_T, set\_T to access to the new defined S and T which will now be vectors. Maybe we can also redefine the get\_S and set\_S inherted and return a std::except (This methoid is not avainable for this class). Maybe we could also delte some methods or constrctutor we delete keyword (see if it is possible to do it).

For some path dependent options, like Asian for instance, could use the constructor of the the Option base class to initiliaze the attributes, but could use them to specifiy more complicated attributes, like for instance std::vector<double> spot prices(num intervals, S) where num\_intervals will be a parameter in the Asian’s constructer.

Maybe can specify one file for each option, each payoff, each solver, and using in those files some functions. Maybe defining a kind of utils cpp file in order to speiciufy some useful functions. We also have to see to which extend we can use things like pointer to member functions, template functions, classes, generic programming maybe.

Can define a Model class which will take an option, a solver, and provide the price. For instance BlackScholes asbtarct base class and specific ones like BlackScholeAnalytical, BlackScholePDE, BlackScholeMC … which will access the option’s paramters and payoff through the option objects getters and payoff\_calc.

Maybe some models, or some solvers will be available for specific Options or kind of options. The list of available options can be passed as an enumerate type, or eneularte class, but using attributes like is\_path\_depdentn can also be very effective. This attribute will be set automatically for each option (not user defined). This approach with if else statement will enable use to define specific implementions.