# Lattice Methods

## Goals

- To show applicability of the one-factor and two-factor binomial options pricing model
- Gain some programming experience
- Have an option calculator
- Create lattice datastructures in C++/learn reuse/flexible design

# Background

- Generalisable numerical method for option pricing
- It uses a discrete time lattice model that describes the underlying and price over time
- Useful method for American and Bermudan options
- Simple method, easy to implement

# “Input”

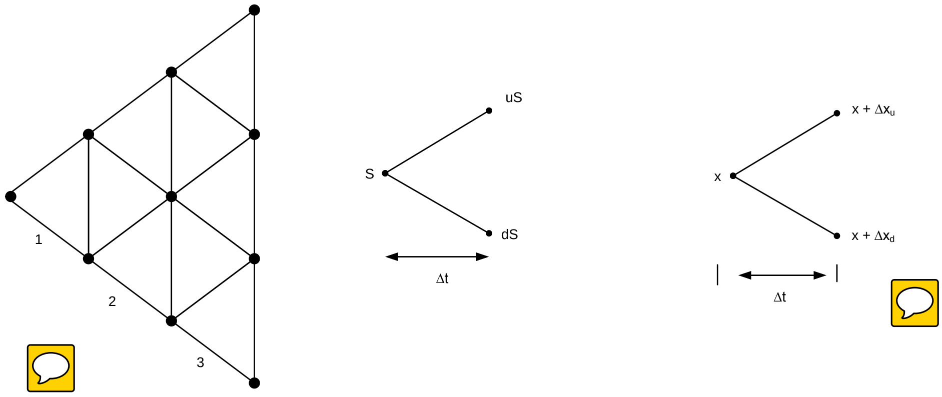
- SDE (additive, multiplicative models) that describes underlying S or x = log(S)
- Determine up and down jumps in discrete lattice
- Forward induction: create the binomial price tree
- Backward induction: compute option price, starting at t = T and navigating to t = 0
- As we navigate, we can ‘test’ various conditions, e.g. early exercise, has a barrier been hit etc.

## SDEs for Lattice Models

- Multiplicative and additive versions

$$
\begin{aligned}
&dS = \mu S dt + \sigma S dW \\
&\text{where} \\
&\mu = \text{drift (constant)} \\
&\sigma = \text{volatility (constant)} \\
&dW = \text{Wiener (Brownian motion) process} \\
&u = \text{'up' jump value} \\
&d = \text{'down' jump value} \\
&p_u = \text{probability that asset price is } uS \\
&p_d = \text{probability that asset price is } dS \\
&(p_d = 1 - p_u)
\end{aligned}
$$

## Lattice



## Up and Down Jumps

- CRR

$$\begin{aligned} u &= exp((r - \tfrac{1}{2}\sigma^2)\Delta t + \sigma\sqrt{\Delta t}) \\ d &= exp((r - \tfrac{1}{2}\sigma^2)\Delta t - \sigma\sqrt{\Delta t}) \\ p_u &= \tfrac{1}{2}, \quad p_d = 1 - p_u \end{aligned}$$

- JR

$$\begin{aligned} u &= exp(\sigma\sqrt{\Delta t}) \\ d &= exp(-\sigma\sqrt{\Delta t}) \\ p_u &= \tfrac{1}{2} + \tfrac{r - \frac{1}{2}\sigma^2}{2\sigma}\sqrt{\Delta t}, \quad p_d = 1 - p_u \end{aligned}$$

## Backwards Induction

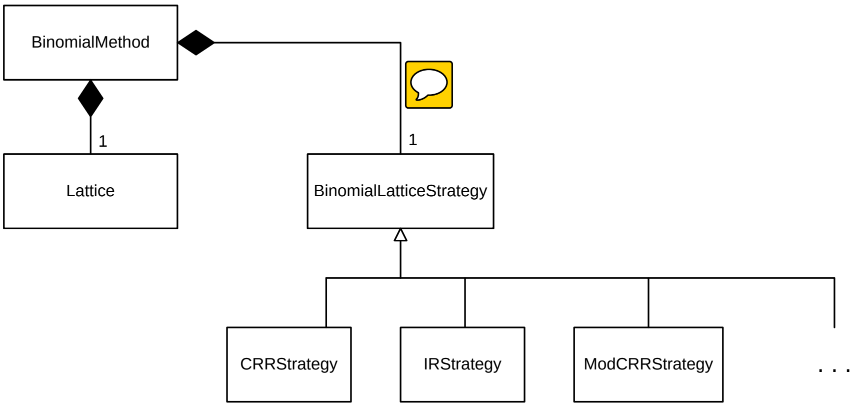
- For American option

$$V_j^n = \max\left(e^{-rk}\left(pV_{j+1}^{n+1} + (1-p)V_j^{n+1}\right), K - S_j^n\right)$$

## Design Goals

- Flexible binomial method solver
- Using appropriate data structures and design patterns
- Learn to understand someone else’s (well-documented J) code
- Focus on flexibility; efficiency not the issue here

## UML Class Diagram



BinomialMethod

1

1

Lattice

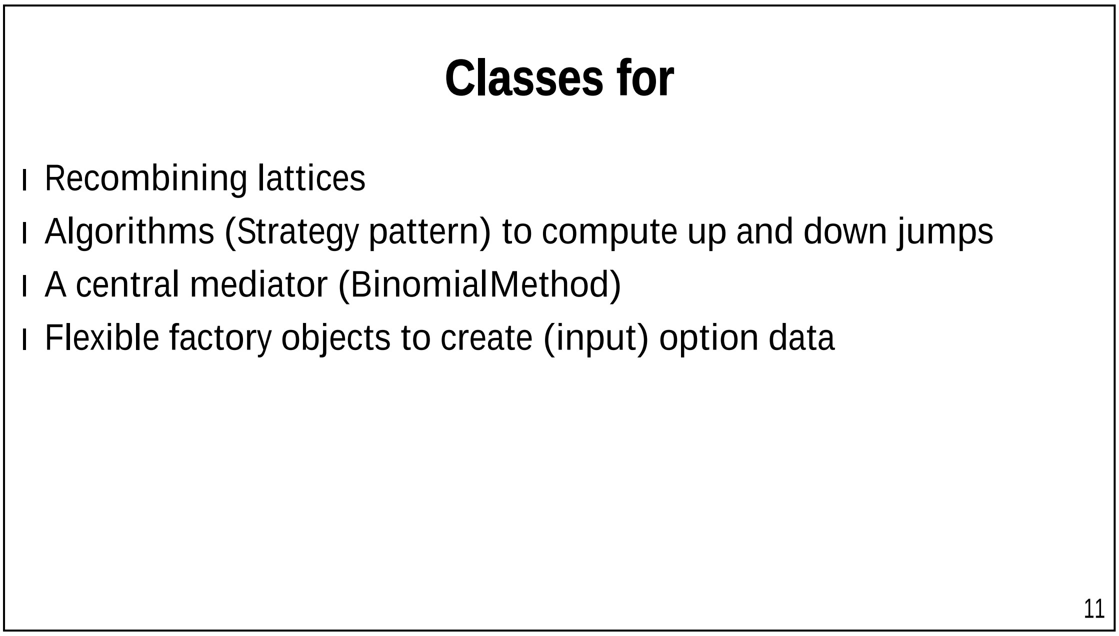
BinomialLatticeStrategy

CRRStrategy

IRStrategy

ModCRRStrategy

. . .



## Classes for

- Recombining lattices
- Algorithms (Strategy pattern) to compute up and down jumps
- A central mediator (BinomialMethod)
- Flexible factory objects to create (input) option data

11