COMPUTATIONAL FINANCE

Lecture 3: Pricing Futures and Futures Options
A Futures Option Pricing Program

Philip H. Dybvig
Washington University
Saint Louis, Missouri

Commodity Spot Market (Cash) Prices

- One-way arb in many storable commodities
  - can go long physical commodity
  - cannot go short
- Adjust return by:
  - storage cost
  - use value

$$E^*[Return] \leq r - use\ value + storage\ cost$$

Dependence of Futures Price on Maturity

- Futures price = risk-adjusted expected spot rate
- If storable and the stock is ample, cannot jump up or down
  - gold and Silver
- If storable with stockouts, can jump down
  - agriculturals: at expected harvest
  - oil: at anticipated easing of supply
  - paper (interesting mkt, futures but no spot)
- If not storable (except at high cost), can jump up or down
  - electricity and natural gas

Forwards versus Futures

A Forward Contract is a commitment to buy a fixed amount at a fixed price at the end

A Futures Contract is defined to have the commodity price at the end (cash or physical settlement) and to be priced correctly at each earlier date so that receiving the change in the futures price is a fair gamble with no initial investment

If the interest rate is nonrandom and there is no arbitrage, then the futures price equals the forward price. This is usually a good approximation even when interest rates are random.

One futures contract gives "more bang for the buck" than one forward contract, since the change in value is received immediately rather than at the end.

Economic Advantages of Futures over Forwards

- Less margin or credit verification is required, since only one day's worth of information arrival needs to be covered.
- The contract is the same every day.

Valuing Commodity Futures

If the futures price moves as

$$F \quad \begin{cases} \nearrow F(1+\delta) \\ \searrow F(1-\delta) \end{cases}$$

this means that an investment of 0 paying the change (called variation) is a fair trade:

$$0 \quad \begin{cases} \nearrow F\delta \\ \searrow -F\delta \end{cases}$$

which is to say that the expected change is zero in the risk-neutral probabilities. For many commodities, this is a good model with delta constant and risk-neutral probabilities of 1/2 and 1/2. (We could take the spacing to be unequal to have volatility fall as the price falls, with the necessary adjustment in the probabilities.)

Valuing and Hedging Futures Options

Valuing commodity futures options is performed exactly like valuing equity options or interest options, once we have established the risk-neutral probabilities.

The simple assumption about risk-neutral probabilities is the one described in the previous slide, although it may be more natural to usual approach for the underlying asset for bond futures options or stock index futures options.

The number of futures to hold is the number needed to give the same difference in value across the two states tomorrow as holding the futures would.

In-class Exercise: Futures Options

Consider a two-period binomial model. The short riskless interest rate is fixed at 10% per period. The two-period corn futures price is \$200 today and will go up or down by \$50 each period, with risk-neutral probabilities 1/2 and 1/2. What is the price today of a futures call option with an exercise price of \$50 and maturity one period from now?

User and File Interface *FutOpStart.java*

```
//
// Binomial futures option pricing program
//
import java.awt.*;
import java.net.*;
import java.io.*;

public class FutOpStart {
  public static void main(String[] args) {
    FutOpFrame f1 = new FutOpFrame();
    f1.setTitle("Futures Option Pricing Program");
    f1.pack();
    f1.show();}}

class FutOpFrame extends Frame {
    int i,j,k;
    FutOp c1;
    ValHedge x;
    double futuresP,strikeP,optionP;
```

```
  String maturity, inputLine, futuresPs;
  URL optionsPage, futuresPage;
  BufferedReader futin, optin;
  TextField r, sigma, nper;
  Label[] tableBody;
  Label north;
public FutOpFrame() {
  setLayout(new BorderLayout());
  setBackground(new Color(245,255,245));
  add("North",north = new Label(
    "Japanese Yen Futures Option Pricing Program: Call",Label.CENTER));
  north.setForeground(Color.blue);
    Panel centr = new Panel();
    centr.setLayout(new GridLayout(15,6));
    centr.add(new Label("maturity",Label.CENTER));
    centr.add(new Label("futuresP",Label.CENTER));
    centr.add(new Label("strikeP",Label.CENTER));
    centr.add(new Label("call:mkt",Label.CENTER));
    centr.add(new Label("call:model",Label.CENTER));
    centr.add(new Label("delta",Label.CENTER));
    tableBody = new Label[84];
```

```
      for(i=0,k=0;i<14;i++) {
        for(j=0;j<6;j++,k++) {
          centr.add(tableBody[k]=new Label("********",Label.CENTER));}}
    add("Center",centr);
      Panel south = new Panel();
      south.setLayout(new GridLayout(1,6));
      south.add(new Label("sigma (%) ="));
      south.add(sigma = new TextField("17",8));
      south.add(new Label("r (%) ="));
      south.add(r = new TextField("5",8));
      south.add(new Label("#periods ="));
      south.add(nper = new TextField("100",8));
    add("South",south);
    c1 = new FutOp();
    recalc();}
  void recalc() {
//
// Read in futures price for near contract
//
  try {
    futuresPage = new URL(
```

```
      "http://dybfin.wustl.edu/teaching/compufinj/homework/3/r_jy.html");
/*    futuresPage = new URL(
        "http://www.cme.com/cgi-bin/prices.cgi?prices/r_jy.html"); */
    futin = new BufferedReader(new InputStreamReader(
      futuresPage.openStream()));
    while(!(futin.readLine().startsWith("STRIKE"))) {};
    futin.readLine();                    // skip one line after table header
    inputLine = futin.readLine();        // read the near futures line
    /* System.out.println(inputLine); */
    maturity = inputLine.substring(0,5);  // extract maturity date as string
    /* System.out.println(maturity); */
    futuresPs = inputLine.substring(33,39);
    futuresP = Double.valueOf(futuresPs).doubleValue(); // futures price
    /* System.out.println(futuresP); */
    futin.close();
  } catch(MalformedURLException e) {
  System.err.println("Error: bad URL");
  System.exit(0);
  } catch(IOException e) {
  System.err.println("Error: Trouble opening or reading futuresPage");
  System.exit(0);}
```

```
//
// Set parameters for the option pricing engine.  Hardwiring days to
// maturity is a cheat!  See the Challenger for more information.
//
    c1.newPars((double) 28.0/365.25, (int) text2double(nper),
      (double) text2double(r)/100.0, (double) text2double(sigma)/100.0);
//
// Read in corresponding futures options prices, compute theoretical value
// and delta, and print
//
  try {
      optionsPage = new URL(
        "http://dybfin.wustl.edu/teaching/compufinj/homework/3/r_oj.html");
/*      optionsPage = new URL(
          "http://www.cme.com/cgi-bin/prices.cgi?prices/r_oj.html"); */
      optin = new BufferedReader(
        new InputStreamReader(optionsPage.openStream()));
    while(!(optin.readLine().startsWith("OJ "+maturity))) {}; // find maturity
    i=0;k=0;
    while( ((inputLine = optin.readLine()) != null) &&
      ((inputLine.length()>40) && (k<14)) ) {
```

```
/* System.out.println(inputLine);
System.out.println("\"" + inputLine.substring(35,39) + "\"");
System.out.println("\"" + inputLine.substring(32,39) + "\"");
System.out.println("\"" + inputLine.substring(39,40) + "\""); */
      try {
      optionP = Double.valueOf(inputLine.substring(32,40)).doubleValue();
      strikeP = Double.valueOf(inputLine.substring(0,7)).doubleValue();
      if(Math.abs(futuresP-strikeP)<futuresP/12.0) {
        (tableBody[6*k]).setText(maturity);
        (tableBody[6*k+1]).setText(futuresPs);
        (tableBody[6*k+2]).setText(inputLine.substring(0,4));
        (tableBody[6*k+3]).setText(inputLine.substring(32,39));
        x = c1.eurcall(futuresP,strikeP);
/* System.out.println(futuresP);
System.out.println(strikeP);
System.out.println(x.value/100.0);
System.out.println(x.delta); */
        (tableBody[6*k+4]).setText(String.valueOf((float)
          Math.floor(1000.0 *(x.value/100.0) + 0.5)/1000.0));
        (tableBody[6*k+5]).setText(String.valueOf((float)
          Math.floor(1000.0 *(x.delta) + 0.5)/1000.0));
```

```
    (tableBody[6*k+3]).setForeground(Color.black);
    if(optionP > 1.05 *(x.value/100.0))
      (tableBody[6*k+3]).setForeground(Color.red);
    if(optionP < .95 * (x.value/100.0))
      (tableBody[6*k+3]).setForeground(Color.green);
      k++;}
    } catch(NumberFormatException e) {}}
  optin.close();
  } catch(MalformedURLException e) {
  System.err.println("Error: bad URL");
  System.exit(0);
  } catch(IOException e) {
  System.err.println("Error: Trouble opening optionsPage");
  System.exit(0);}}
public boolean handleEvent(Event event) {
  if(event.id == Event.WINDOW_DESTROY) {
    System.exit(0);}
  return super.handleEvent(event);}
double text2double(TextField tf) {
  return Double.valueOf(tf.getText()).doubleValue();}
public boolean action(Event ev, Object arg) {
```

```
if(ev.target instanceof TextField) {
  recalc();
  return true;}
return false;}}
```

Computational Engine *FutOp.java*

```
//
// Binomial futures option pricing model: computational engine
//

public class FutOp {

  int nper;
  double tinc,disc,up,down,prcup,prcdn;
  double val[];

  public FutOp() {}


  public void newPars(double ttm,int npers,double r,double sigma) {
    nper = npers;
    tinc = ttm/(double) nper;
    disc = Math.exp(-r * tinc);
    up = 1.0 + sigma * Math.sqrt(tinc);
    down = 1.0 - sigma * Math.sqrt(tinc);
```

```
    prcup = 0.5*disc;
    prcdn = 0.5*disc;
    val = new double[npers+1];}

  public ValHedge eurcall(double f0,double X) {
    int i,j;
    double futprice;
    ValHedge x1 = new ValHedge();
// initialize terminal payoffs
// i is the number of up moves over the whole life
    for(i=0;i<=nper;i++) {
      futprice = f0 * Math.pow(up,(double) i)
        * Math.pow(down,(double) (nper-i));
      val[i] = Math.max(futprice - X,0.0);}
// compute prices back through the tree
// j+1 is the number of periods from the end
// i is the number of up moves from the start
    for(j=0;j<nper-1;j++) {
      for(i=0;i<nper-j;i++) {
        val[i] = prcdn * val[i] + prcup * val[i+1];}}
    x1.value = prcdn * val[0] + prcup * val[1];
```

```
x1.delta = (val[1]-val[0]) / (f0*(up-down));
return(x1);}}
```

ValHedge Object Definition *ValHedge.java*

```
public class ValHedge {
  public double value;
  public double delta;
  public ValHedge() {}}
```