Modelling Financial Derivatives

Dr. William T. Shaw
Quantitative Analysis Group
Nomura International plc
London

in conjunction with

Wolfram Research;
Cambridge University Press.

Email: william.shaw@nomura.co.uk

Präsentiert von:
ADDITIVE Soft- und Hardware für Technik und Wissenschaft GmbH
Rohrwiesenstraße 2
61381 Friedrichsdorf
Telefon: 06172-5905-0 Fax: 06172-77613
mathematica@additive-net.de http://www.additive-net.de

Plan of Presentation

Part 1: General Principles

Explaining how and why you use a system such as *Mathematica* in the particular context of financial modellin

Part 2: Issues with standard models

Flaws in standard algorithms, and what you can do about it, and how to use *Mathematica* to do new things eas

Acknowledgements

Nomura International, and numerous colleagues there. In particular Reza Ghassemieh, head of Quantitative An traders and modellers with infinite patience, and the Risk Management teams;

Cambridge University Press, for commissioning a text on the matters raised;

Wolfram Research, of course, for the software, and for arranging these presentations;

Numerous authors, for giving me something curious to think about in the numerical analysis - I do not wish to with the luxury of having something in place to criticise - but we do need to consider carefully whether the firs should continue using, especially in the light of considerations of numerical analysis, and of detailed comparis

Principles of Derivatives Modelling in a Symbolic Algebra Environment

1. Symbolic Calculus and the Greeks

Binary Options - a Quickie

In advanced symbolic systems, many of the ingredients to define basic options are already present. For examp function removes the need to code up some form of rational or other approximation to the cumulative normal definition:

```
Norm@x_D := 1/2 (1 + Erf[x/Sqrt[2]]); Norm@z_?NumberQD := N[0.5 (1 + Erf[z/Sqrt[2]]
```

Two lines of code then suffice to define a standard Binary Call option:

```
dtwo@S_, s_, K_, t_, r_, q_D := (Log[S/K] + (r - q) t)/(s Sqrt[t]) - (s Sqrt[t])/2;

BCall@B_, S_, K_, s_, r_, q_, t_D := B Exp@-r tD Norm@dtwo@S, s, K, t, r,
```

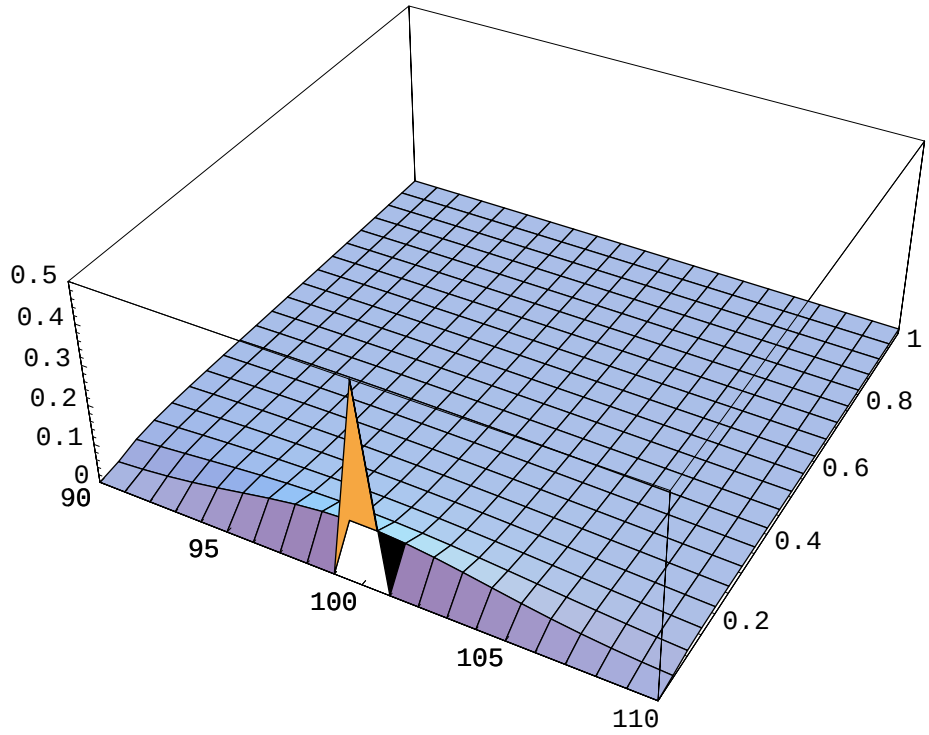
The delta and other Greeks can be computed by asking the system to do the differentiation - the system output

```
BDelta@B_, S_, K_, s_, r_, q_, t_D = ∂s BCall@B, S, K, s, r, q, tD
```

$$B\,E^{-r\,t-\frac{1}{2}\left(\frac{\sqrt{t}\,s}{2}+\frac{(-q+r)\,t+\text{Log}\left[\frac{S}{K}\right]}{\sqrt{t}\,s}\right)^2}\sqrt{t}$$

While this example is not hard with "pen & paper", corresponding formulae for, e.g. Barrier options with rebat errors. It is also a simple matter to explore such functions using elegant 3D graphics tools. Here we plot the de nightmare at the strike, near maturity, is easily exposed:

```
Plot3D@BDelta@1, S, 100, 0.2, 0.1, 0, tD, 8S, 90, 110<,
 8t, 0.0001, 1<, PlotPoints -> 21, PlotRange -> 80, 0.5<D;
```



Using such an approach, the whole range of analytical exotics can be coded up, from the fundamental research

Barrier Option Quickie

Let's take a look at something more serious, to see the real power of symbolic calculus on a computer. The foll
Reiner (see e.g. "Breaking Down the Barriers", RISK, September 1991). We use continuously compounded va

```
l@r_, q_, sd_D := 1/2 + (r - q)/sd^2
```

```
x@p_, k_, sd_, r_, q_, t_D := Log[p/k]/(sd Sqrt[t]) + l@r, q, sdD sd Sqrt[t];
```

```
xone@p_, h_, sd_, r_, q_, t_D := x@p, h, sd, r, q, tD;
```

```
y@p_, k_, h_, sd_, r_, q_, t_D := Log[h^2/(p k)]/(sd Sqrt[t]) + l@r, q, sdD sd Sqrt[t];
```

```
yone@p_, h_, sd_, r_, q_, t_D := x@h, p, sd, r, q, tD;
```

```
a@r_, q_, sd_D := (r - q)/sd^2 - 1/2;

b@r_, q_, sd_D := (-q - sd^2/2 M^2 + 2 r sd^2)/sd^2;

z@p_, h_, sd_, r_, q_, t_D := Log[h/p]/(sd Sqrt[t]) + b@r, q, sdD sd Sqrt[t];
```

```
intone@p_, k_, sd_, r_, q_, t_, f_D :=
 f p Exp@-q tD Norm@f x@p, k, sd, r, q, tDD - f k Exp@-r tD NormAf x@p, k, s
```

```
inttwo@p_, k_, h_, sd_, r_, q_, t_, f_D :=
 f p Exp@-q tD Norm@f xone@p, h, sd, r, q, tDD - f k Exp@-r tD NormAf xone@p
```

```
intthree@p_, k_, h_, sd_, r_, q_, t_, f_, h_D := f p Exp@-q tD (h/p)^2 l@r,q,s

  f k Exp@-r tD (h/p)^2 l@r,q,sdD-2 NormAh y@p, k, h, sd, r, q, tD - h/(sd Sqrt[t])E
```

```
intfour@p_, k_, h_, sd_, r_, q_, t_, f_, h_D := f p Exp@-q tD (h/p)^2 l@r,q,sd

  f k Exp@-r tD (h/p)^2 l@r,q,sdD-2 NormAh yone@p, h, sd, r, q, tD - h/(sd Sqrt[t])E
```

```
intfive@reb_, p_, h_, sd_, r_, q_, t_, h_D :=
 reb Exp@-r tD (NormAh xone@p, h, sd, r, q, tD - h/(sd Sqrt[t])E - (h/p)^2 l@r,q,sdD-2
```

```
intsix@reb_, p_, h_, sd_, r_, q_, t_, h_D := reb (h/p)^a@r,q,sdD+b@r,q,sdD Nor

    (h/p)^a@r,q,sdD-b@r,q,sdD NormAh z@p, h, sd, r, q, tD - 2 h b@r, q, sdD sd Sqrt[t]
```

Up/Down And Out Puts

We can do numerical and graphical checks against the original Rubenstein and Reiner paper. First we introduc
computer does all the symbolic differentiations necessary to compute delta, gamma, vega etc.

```
UpAndOutPut[reb_, p_, k_, h_, sd_, r_, q_, t_] :=
If[k > h,
inttwo[p, k, h, sd, r, q, t, -1] -
intfour[p, k, h, sd, r, q, t, -1, -1] +
intsix[reb, p, h, sd, r, q, t, -1],
intone[p, k, sd, r, q, t, -1] -
intthree[p, k, h, sd, r, q, t, -1, -1] +
intsix[reb, p, h, sd, r, q, t, -1]]
```

```
UpAndOutPutDelta[reb_, p_, k_, h_, sd_, r_, q_, t_] =
If[k > h,
Evaluate[D[inttwo[p, k, h, sd, r, q, t, -1] -
intfour[p, k, h, sd, r, q, t, -1, -1] +
intsix[reb, p, h, sd, r, q, t, -1], p]],
Evaluate[D[intone[p, k, sd, r, q, t, -1] -
intthree[p, k, h, sd, r, q, t, -1, -1] +
intsix[reb, p, h, sd, r, q, t, -1],p]]];
```

```
UpAndOutPutTheta[reb_, p_, k_, h_, sd_, r_, q_, t_] =
If[k > h,
Evaluate[-D[inttwo[p, k, h, sd, r, q, t, -1] -
intfour[p, k, h, sd, r, q, t, -1, -1] +
intsix[reb, p, h, sd, r, q, t, -1], t]],
Evaluate[-D[intone[p, k, sd, r, q, t, -1] -
intthree[p, k, h, sd, r, q, t, -1, -1] +
intsix[reb, p, h, sd, r, q, t, -1],t]]];
```

```
UpAndOutPutGamma[reb_, p_, k_, h_, sd_, r_, q_, t_] =
If[k > h,
Evaluate[D[inttwo[p, k, h, sd, r, q, t, -1] -
intfour[p, k, h, sd, r, q, t, -1, -1] +
intsix[reb, p, h, sd, r, q, t, -1], {p,2}]],
Evaluate[D[intone[p, k, sd, r, q, t, -1] -
intthree[p, k, h, sd, r, q, t, -1, -1] +
intsix[reb, p, h, sd, r, q, t, -1],{p,2}]]];
```

```
UpAndOutPutVega[reb_, p_, k_, h_, sd_, r_, q_, t_] =
If[k > h,
Evaluate[D[inttwo[p, k, h, sd, r, q, t, -1] -
intfour[p, k, h, sd, r, q, t, -1, -1] +
intsix[reb, p, h, sd, r, q, t, -1], sd]],
Evaluate[D[intone[p, k, sd, r, q, t, -1] -
intthree[p, k, h, sd, r, q, t, -1, -1] +
intsix[reb, p, h, sd, r, q, t, -1],sd]]];
```

```
UpAndOutPutRho[reb_, p_, k_, h_, sd_, r_, q_, t_] =
If[k > h,
Evaluate[D[inttwo[p, k, h, sd, r, q, t, -1] -
intfour[p, k, h, sd, r, q, t, -1, -1] +
intsix[reb, p, h, sd, r, q, t, -1], r]],
Evaluate[D[intone[p, k, sd, r, q, t, -1] -
intthree[p, k, h, sd, r, q, t, -1, -1] +
intsix[reb, p, h, sd, r, q, t, -1],r]]];
```

Getting the Computer to do the Work!

For example - a less trivial delta:

```
UpAndOutPutDelta@reb, p, k, h, sd, r, q, tD
```

Numerical Checks against original research

First we checked that our exact model agreed with the results tabulated in the original research. The following table in Rubenstein and Reiner. It is in precise agreement, bearing in mind the limited precision of the results c

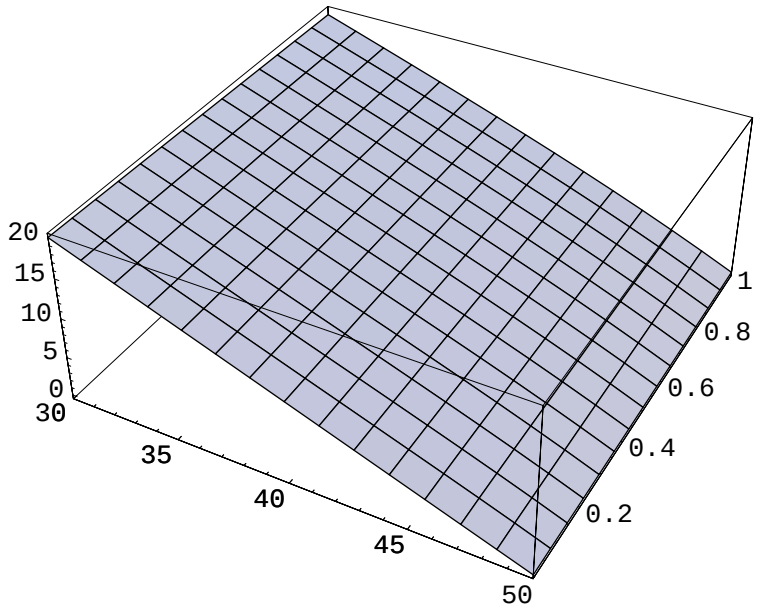
```
MatrixForm[
Table[N[UpAndOutPut[2,100,k,103,0.20,Log[1.1],Log[1.05],t]],
{k, 90, 110, 10},
{t, 0.5, 1.5, 0.25}]]
```

90	2.36986	2.55061	2.63858	2.68125	2.69901
100	3.54582	3.52027	3.46541	3.40241	3.33814
110	4.96897	4.62758	4.38154	4.18678	4.02461

Case Investigated: A
K = H (barrier = strike) and q > r (div. yield greater than risk free rate.)

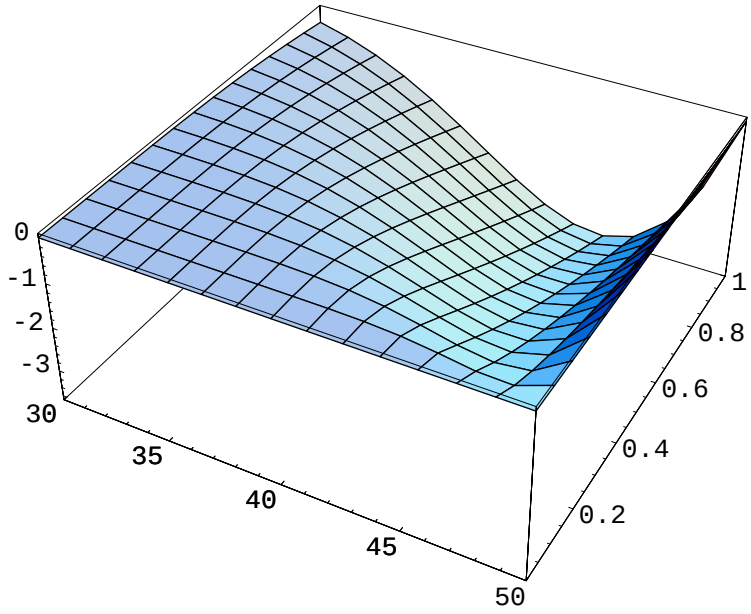
In the following plots we investigate the structure of the valuation function and associated delta, gamma, and v the value, here viewed as a surface V(S,t). The value is zero at the barrier, here set to 50. The strike is also 50, compounded interest rate r =10%. The dividend yield is 15%. The time is measured in years and varies from z (barrier).

```
Plot3D[
UpAndOutPut[0, S, 50, 50, 0.25,0.1, 0.15, t],
{S, 30, 50}, {t, 0.001, 1}, PlotRange -> All];
```



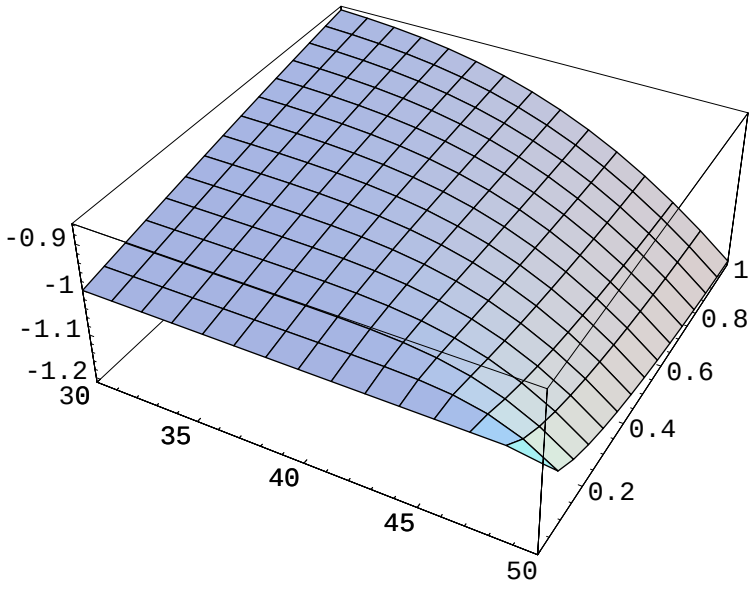
In the following we show vega, computed symbolically by the computer. It is negative everywhere.

```
Plot3D[
UpAndOutPutVega[0, S, 50, 50, 0.25,0.1, 0.15, t],
{S, 30, 50}, {t, 0.001, 1}, PlotRange -> All];
```



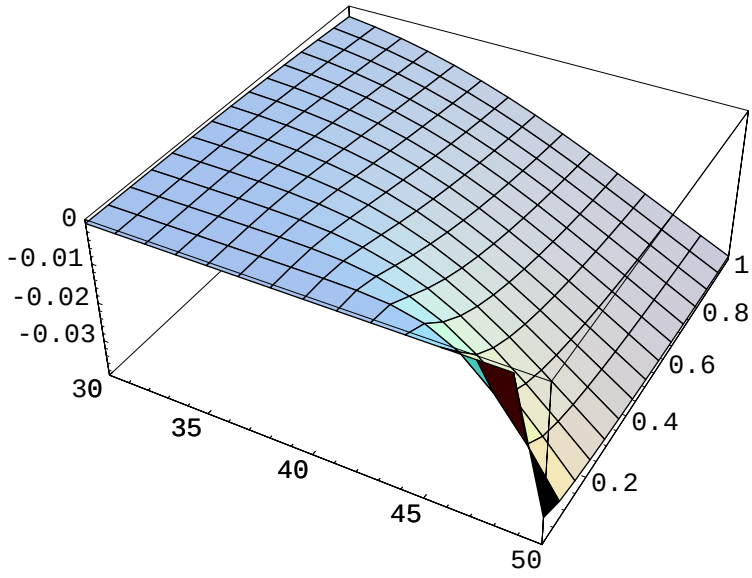
Because of the flatness of the value surface, delta is close to -1 everywhere, as revealed in the next plot.

```
Plot3D[
UpAndOutPutDelta[0, S, 50, 50, 0.25,0.1, 0.15, t],
{S, 30, 50}, {t, 0.001, 1}, PlotRange -> All];
```



Gamma is very small and NEGATIVE, as is shown in the final plot (note the vertical axes scale.)

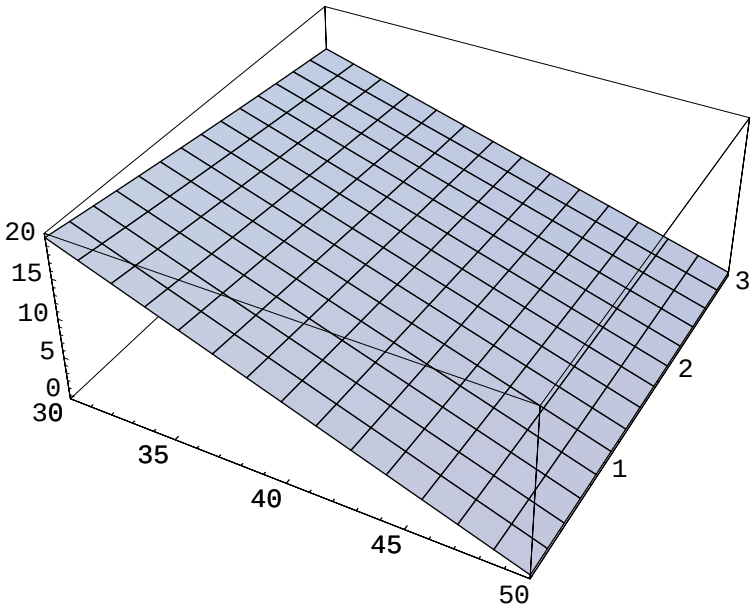
```
Plot3D[
UpAndOutPutGamma[0, S, 50, 50, 0.25,0.1, 0.15, t],
{S, 30, 50}, {t, 0.001, 1}, PlotRange -> All];
```



Case Investigated: B
K = H (barrier = strike) and q = r (div. yield equal to risk free rate.)

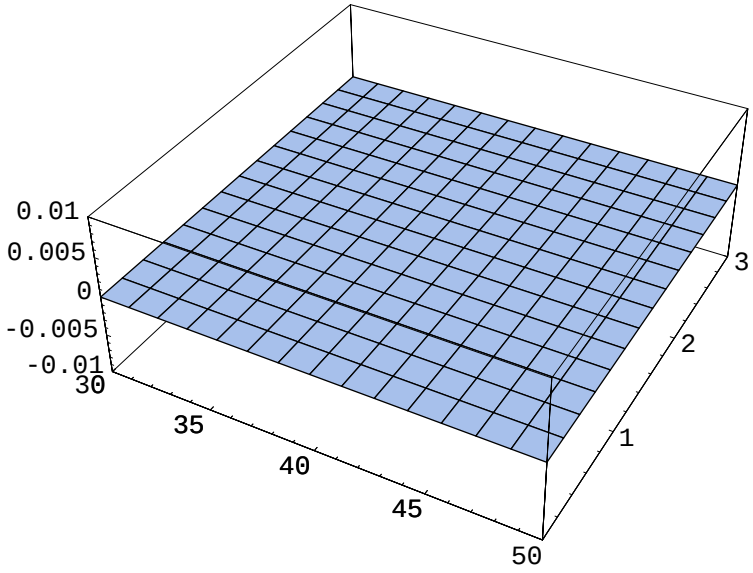
The strike is also 50, the volatility is 0.25 (25%), and the continuously compounded interest rate r =10%. The and varies from zero to 3 year. The stock price varies from 30 to 50 (barrier).

```
Plot3D[
UpAndOutPut[0, S, 50, 50, 0.25,0.1, 0.1, t],
{S, 30, 50}, {t, 0.001, 3}, PlotRange -> All];
```



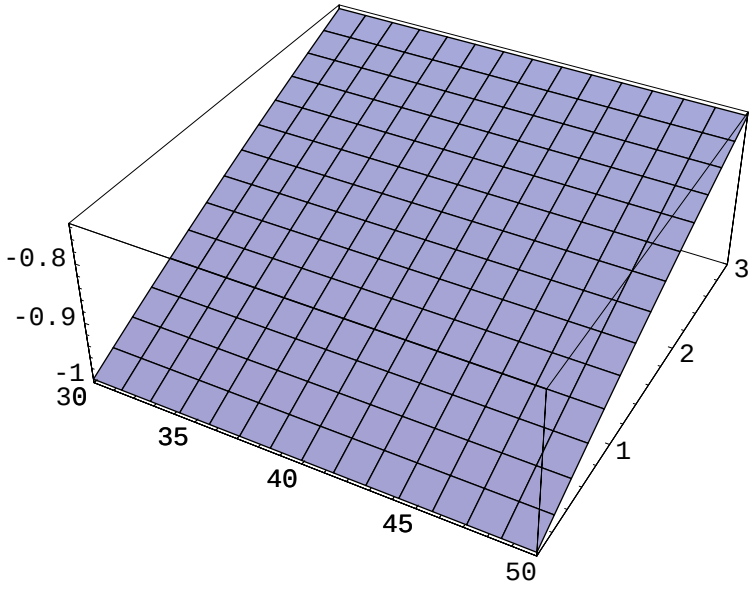
In the following we show vega, computed symbolically by the computer. It is zero everywhere (to machine pr
consequences of this and other problems for the notion of implied volatility.

```
Plot3D[
UpAndOutPutVega[0, S, 50, 50, 0.25,0.1, 0.10, t],
{S, 30, 50}, {t, 0.001, 3}, PlotRange -> {-0.01, 0.01}];
```



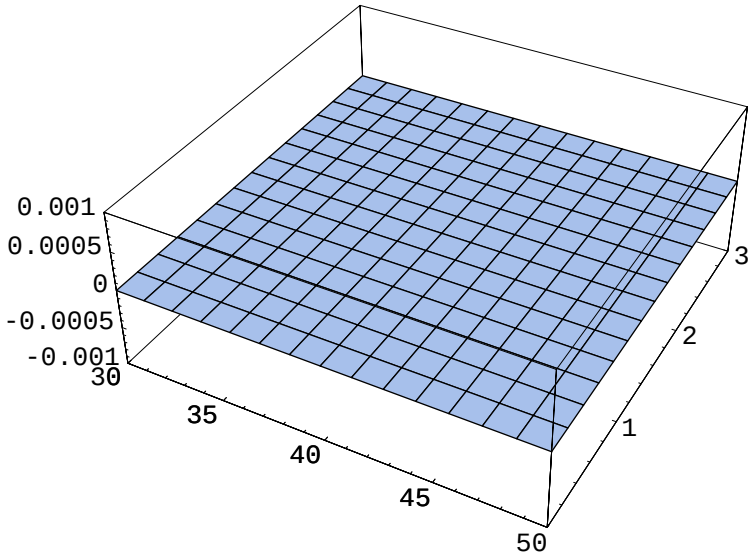
Because of the flatness of the value surface, delta is close to -1 everywhere, as revealed in the next plot - it is c

```
Plot3D[
UpAndOutPutDelta[0, S, 50, 50, 0.25,0.1, 0.1, t],
{S, 30, 50}, {t, 0.001, 3}, PlotRange -> All];
```



Gamma is zero, as is shown in the next plot:

```
Plot3D[
UpAndOutPutGamma[0, S, 50, 50, 0.25,0.1, 0.1, t],
{S, 30, 50}, {t, 0.001, 3}, PlotRange -> {-0.001, 0.001}];
```



2. Exact Solutions and Special Functions in Verification

Informal Definitions of Model and Algorithm Risk

Model Risk can arise in a variety of ways. Given a financial instrument, we generally formulate a 3-stage valu

a) Conceptual description;

b) Mathematical formulation of the conceptual description;

c) Solution of the mathematical model through an analytical and numerical process.

It is possible to make mistakes, or perhaps unreasonable simplifying assumptions, at any one of these three sta risk that the process from reading a contract to responding with a set of answers (fair value, delta, gamma, ..., i supplied.

In this talk I will focus mainly on the third point, which we might call *algorithm risk*. One might think that thi that derivative securities are capable of exhibiting some diverse forms of mathematical pathology that confour of-the-art algorithms. Later in this talk I will present some familiar and hopefully less familiar examples of tro

Verification and resolving disagreement amongst experts

A good rule of thumb is that in general you can give what you think is the same problem to 6 different groups derivatives modelling you can usually get 4 different answers, especially if you include:

a) One Analytics fanatic;

b) One Tree-Model fanatic;

c) One Monte-Carlo fanatic;

d) One Finite-Difference fanatic

among the group of people you choose.

The only way to sort out material differences in answers is to perform some form of *verification study*.This ca important ones are:

a) Comparison of modelling systems with some form of absolutely accurate benchmark (typically only availab algorithms do work correctly when specialized to simple cases);

b) Intercomparison of models and codes when applied to complex instruments.

The basic idea is to define a comprehensive set of test problems, and run off a benchmark model and operatio

The Power of Symbolic Computer Calculus

In recent years tools such as *Mathematica* have emerged, that extend the utility of computers from "Number C calculus. These systems combine such symbolic capability with advanced numerical algorithms, 2D, 3D and a tools are particularly relevant to the derivatives modelling and testing area due to

a) the fact that they have built-in routines for most basic common operations;

b) the ability to do calculus with simple and complex functions can eliminate many of the headaches of Greek'

c) the ability to compare analytical symbolic models directly with the results from a numerical algorithm;

d) the ability to visualize the results, and hopefully literally see the errors if there are any.

The Use of *Mathematica*

The power of *Mathematica* as an investigate tool becomes clear in this context. We can define a series of test *Mathematica* to characterize the solution exactly, using *Mathematica*'s infinite-precision arithmetic to get the *Mathematica*'s symbolic calculus capabilities to define the Greek's by differentiation. Then, for more complex within *Mathematica*. Then we make a comparison, within *Mathematica*, between these analytical forms and t company systems. The results are compared, and the reasons for any differences or other problems are explore

We shall see detailed examples in Part 2.

3. Reporting Tools

Can produce reports showing:

the mathematics;

comparisons with basic research;

sample values;

2D and 3D Plots;

Presentation in V 3.0

Can take textual input and tidy up for presentation. Let's look at one of our definitions as it was input:

Input Form (version 2.x)

Standard Form

Traditional Form

Writing it Up (Numbered Equations)

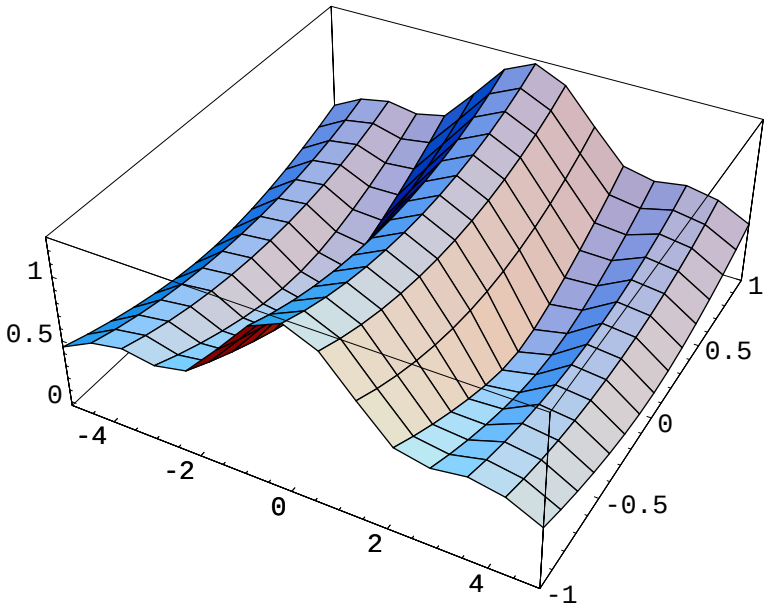
As you have already seen, this can be combined with prose, plots, and in-line material a = $\bar{s}$ - you have a fu
with the property that the equations can be evaluated.

4. Compilation - the new Numerical Features

Shortly after *Mathematica* was first released a few financial people realised the importance of its symbolic, gr
unfair to say that there was some disappointment on the speed of its execution of purely numerical operations.

Some of this criticism was not justified. For example, it has always been possible to deal with a myriad of spec
efficiently, using fast compiled built-in routines:

```
Plot3D@Abs@BesselJ@0, x + I yDD, 8x, -5, 5<, 8y, -1, 1<D;
```



'C' fanatics should consider carefully the effort involved in doing this, especially with arbitrary precision arithı

```
N@BesselJ@0, 2 + 3 I DD
```

```
- 0.469517 - 4.31379
```

```
N@BesselJ@0, 2 + 3 I D, 40D
```

```
- 0.469517192044070187374681089001214396155 - 4.31378840946892241033616505803603313
```

This example might seem contrived, but it is directly relevant to derivatives:

a) CIR interest rate model and CEV equity model both involve Bessel functions;

b) Laplace transform models involve complex analytical structures and contour integration.

Secondly, one should never forget that programming time is expensive compared to faster computers. The rap
been a significant factor in keeping down development costs.

However, until version 3, the compilation of a user-defined numerical algorithm was limited to routines with a
3, you can compile routines involving scalars, vectors, matrices (indeed tensors of arbitrary rank) filled with ir
possibility of efficient compiled models using:

* binomial trees;

* implicit finite-difference methods.

We shall look at some results obtained from the second case in part 2.

Here's a quick canter throught the simplest type of American option done with a compiled binomial model. Fi
a CRR tree:

```
CRRupBinApp[n_, r_, q_, s_, T_] := Exp[s Sqrt[T/n]];
CRRdownBinApp[n_, r_, q_, s_, T_] := Exp[-s Sqrt[T/n]];
Discount[n_, r_, T_] := N[Exp[-r T/n]];
afunc[n_, r_, q_, T_] := N[Exp[(r-q) T/n]];
Pfunc[up_,down_, a_, disc_] := N[disc*(a-down)/(up-down)];
Qfunc[up_, down_, a_, disc_] := N[disc-Pfunc[up,down,a,disc]];
```

Next we compile the operations to do one and many time steps (note details):

```
OneTimeStepAmerFun = Compile@88initial, _Real, 1<, 8payoff, _Real,
  MapThread@Max, 8payoff, p * Drop@initial, -1D + q * Drop@RotateLeft@
```

```
ManyTimeStepAmerPutFun =
Compile@88initial, _Real, 1<, 8p, _Real, 0<, 8q, _Real, 0<,
   8S, _Real, 0<, 8u, _Real, 0<, 8d, _Real, 0<, 8n, _Integer, 0<, 8
Module@8new = initial, k, r, old, vold, payoff, finpay = 8Max@K - S,
For@k = 1, k <= n - 1, k++,
Hpayoff = Table@Max@K - S d^r u^Hn - k - rL, 0D, 8r, 0, n - k<D;
old = new; new = OneTimeStepAmerFun@old, payoff, p, qDLD;
vold = old; old = new;
new = OneTimeStepAmerFun@old, finpay, p, qD;
8Join@new, oldD, vold<DD;
```

Finally we wrap it up and extract the value, delta, gamma and theta:

```
CRRFastAmerPutFun[S_, K_, n_, r_, q_, s_, T_] :=
Module[{u = CRRupBinApp[n,r,q,s,T], d = CRRdownBinApp[n,r,q,s,T],
a = afunc[n,r,q,T],
disc = Discount[n,r,T],
P,Q,TreeOp,result,payoffin},
P = Pfunc[u,d,a,disc];
Q = Qfunc[u,d,a,disc];
payoffin = Table[Max[K-S d^node u^(n-node),0], {node, 0, n, 1}];
result = ManyTimeStepAmerPutFun[payoffin,P,Q,S,u,d,n,K];
value = result[[1,1]];
delta = (result[[1,2]] - result[[1,3]])/((u-d)*S);
gamma = ((result[[2,1]] - result[[2,2]])/(u^2-1) - (result[[2,2]] -
2));
theta = (result[[2,2]] - result[[1,1]])/(2*T/n);
{value, delta, gamma, theta}]
```

For example - here is an American put with a binomial tree with 100 slices. On my 300Mhz machine (Power M
comparing performance in this talk with state of the art or what you have at work):

```
Timing@CRRFastAmerPutFun@10, 10, 100, 0.05, 0, 0.2, 1DD
```

```
80.4 Second, 80.608235, -0.411636, 0.231395, -0.22626<<
```

```
Timing@CRRFastAmerPutFun@10, 10, 100, 0.05, 0, 0.2, 1DD
```

5. Mathematica vs C/C++ and Spreadsheets

It is my firm belief that in the long term such hybrid symbolic-numeric-graphical systems will replace the spre
modelling. Their primary advantage lies in the provision of advanced symbolic capabilities. In a numerically-b
Greek such as delta, are a) to work out the derivative on paper and code it up, b) work out the value of an optic
and take differences. Both of these have drawbacks, and the potential for human error at the pen and paper sta
differentiation problems. Instead we can now ask the computer to do the differentiation for us. Such symbolic
can dramatically reduce development time below that for spreadsheet or C/C++ systems. Furthermore, in the t
integrated environment where a numerical model can be compared directly with its exact form. And I mean ex
with ease. The latest versions of these systems can also produce mathematical typeset material, so you can wri
plot the results, and produce a report for Risk Management/Trading/Regulatory bodies all within one environn

Symbolic Calculus

Integrated numerics, symbolics, graphics, programming

Reporting (Typesetting) Tools Built in.

Arbitrary Precision Arithmetic

Clear Organization

Definition of a Spreadsheet (not one all my colleagues support!) - the best way ever invented to muddle up inp
"3 Men and a Baby" when the issue of changing a baby's nappy (diaper to US folks) arises - one of the guys of
feel when asked to find out what is wrong with a spreadsheet model. There is data, equations, inputs and outpu
program should have the structure:

input and other data;

model;

output.

Mathematica encourages this, indeed, it goes several stages further - a notebook can have:

Prose (your explanations);

Maths (typeset equations);

Model Definition (executable form of maths - it looks almost the same);

Inputs;

Model Execution;

Outputs, (tables, 2D, 3D graphics)

Prose comments.

This encourges total transparency of function once you get used to the language syntax. It is a great way of m
done the right thing (or, indeed, allowing others to see where you may have messed up). Such reports can go t
internal and formal external model approval.