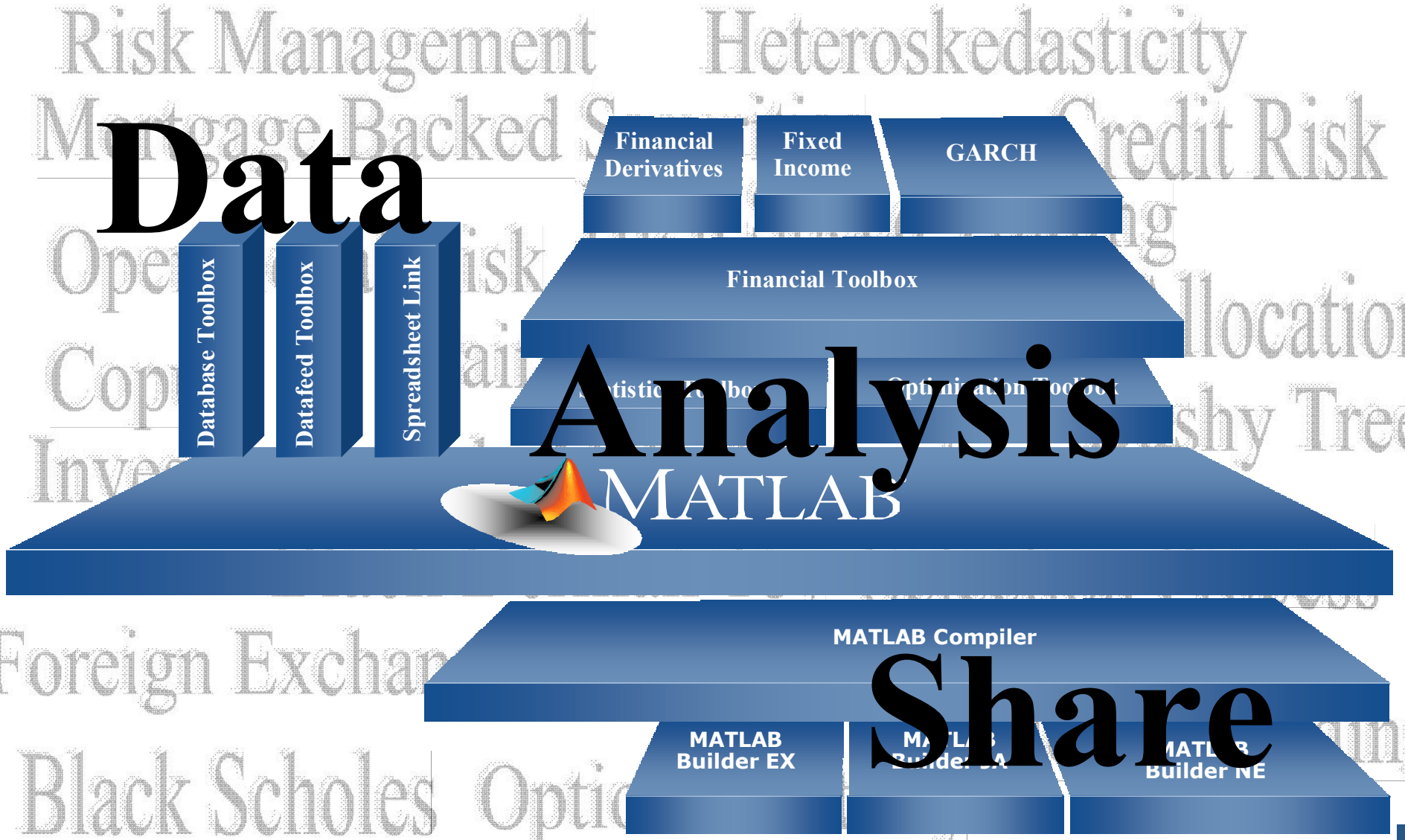
MATLAB Integration in Financial Applications

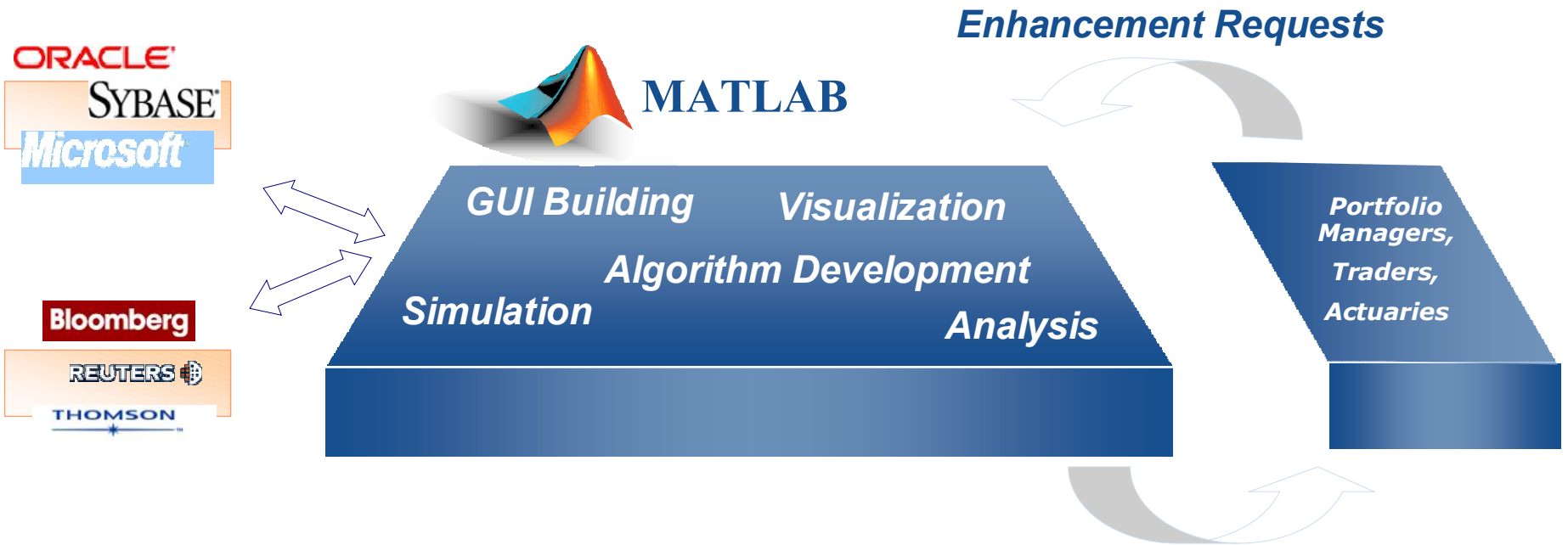
Eugene McGoldrick

The MathWorks Inc.

Typical Process Flow … MATLAB Tools



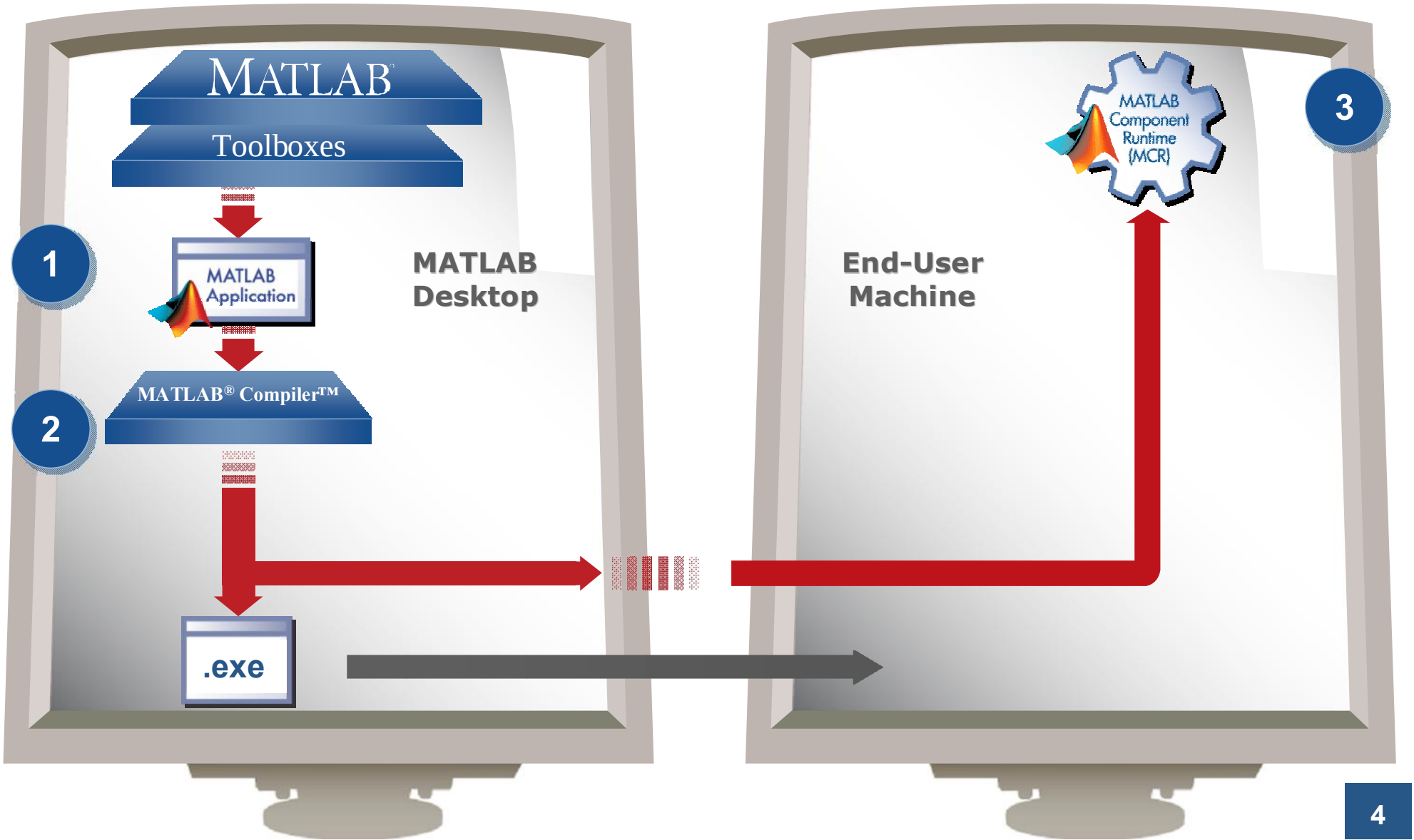
Deploying Applications with MATLAB



Traditional prototype to production system port
… development timeline **2 weeks ~ 6+ months**

MATLAB prototype to production system
… Push button in the new project manager

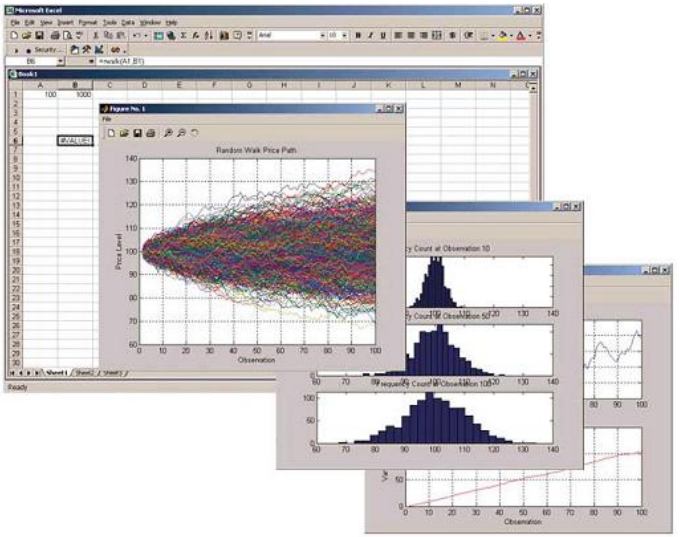
Deploying Applications with MATLAB



MATLAB Builder™ EX

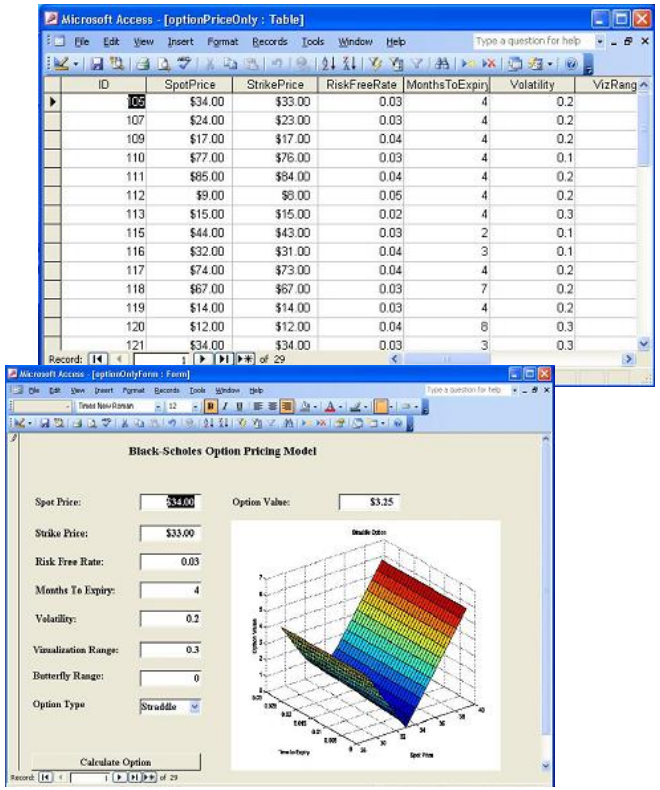
Excel Add-in

- Automatically converts MATLAB algorithms into independent Excel add-ins
- Free deployment model
- Advantages of MATLAB Excel add-ins:
  - Run faster than Visual Basic add-ins
  - Handle larger data sets and run larger simulations
  - Improved productivity



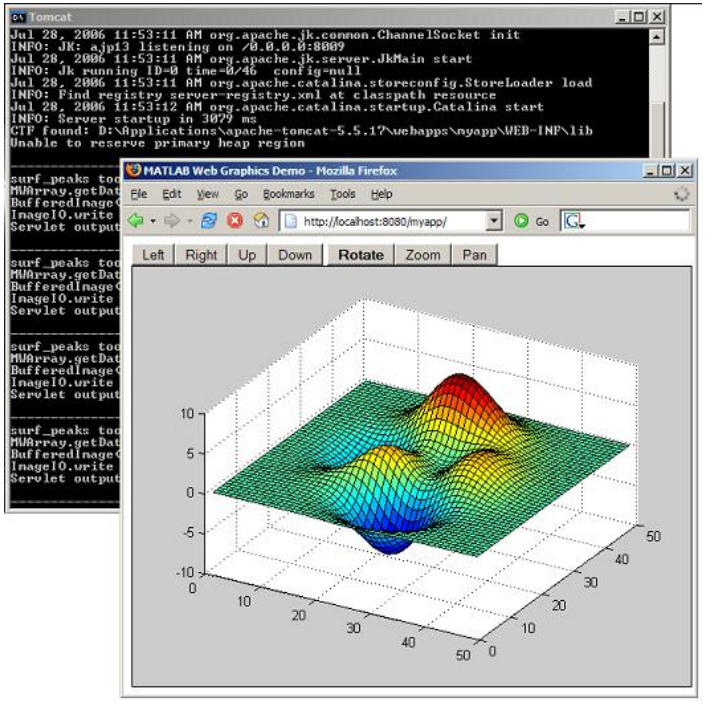
MATLAB Builder™ NE

- Automatically generates independent .NET or COM components
- .NET components can be integrated with .NET framework languages, including:
  - VB.NET
  - C#
  - Web applications in ASP.NET
- COM components can be integrated with any COM-compliant technology, including:
  - Visual Basic
  - Microsoft Excel
  - Web applications in ASP
- Free deployment model



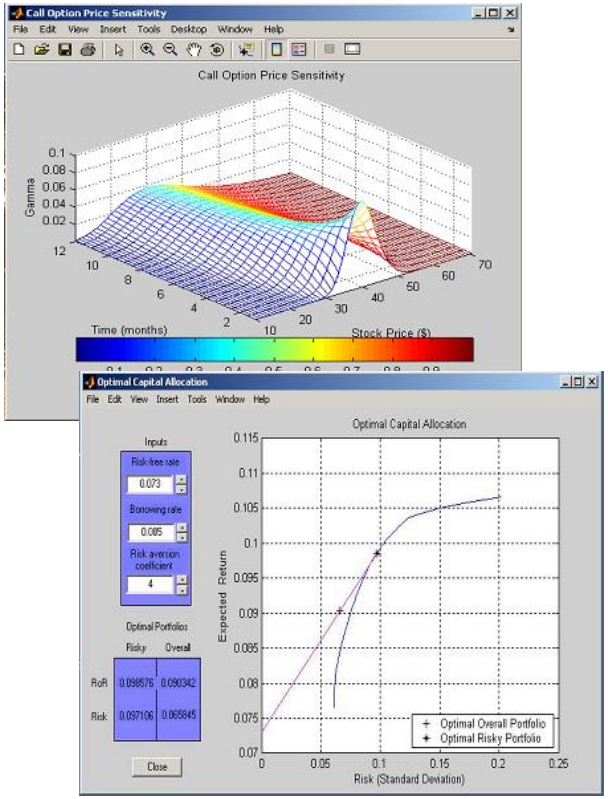
MATLAB Builder™ JA

- Automatically generates Java classes from MATLAB functions
- Java components can be integrated with:
  - Java applications
  - Java application servers
  - Web applications: Servlets, JSP
- Free deployment model



Desktop Applications

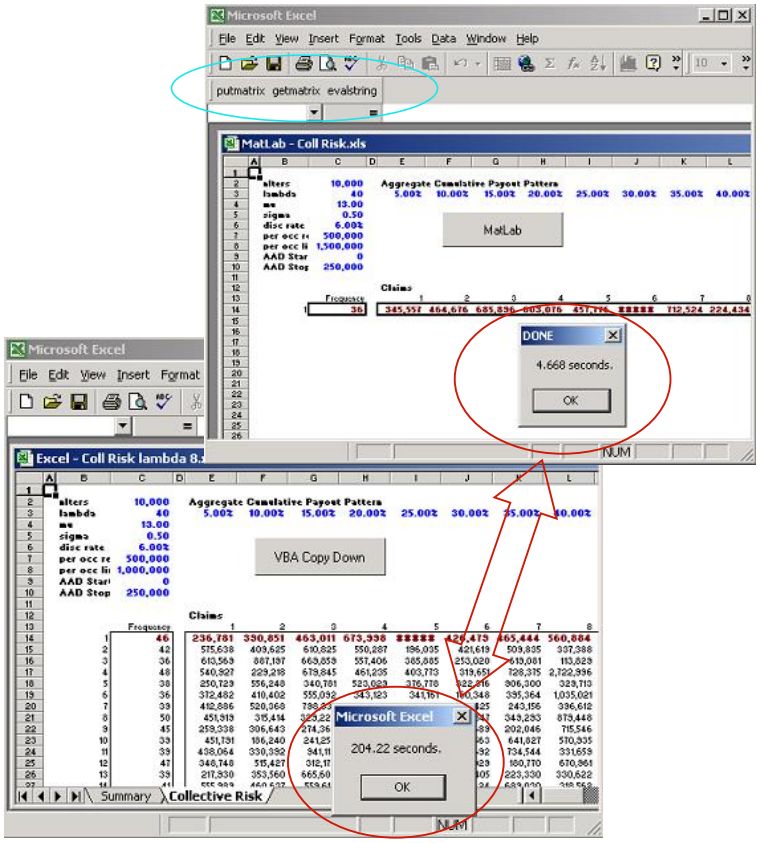
- Standalone Executables
- Excel Spreadsheets
- Access Database
- Integration with .Net applications
- Integration with Java Applications

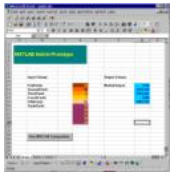


Accelerate Spreadsheet Computations

Spreadsheet Link

- Dynamically link into MATLAB from Excel
- MATLAB as the computational engine
- Faster scalable solution



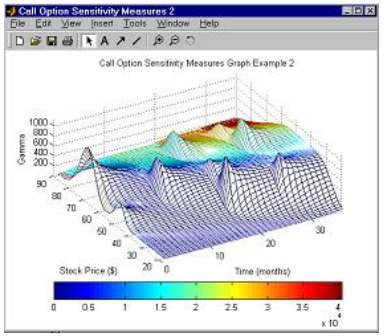


Collective Risk Model
5 Seconds vs. 204 Seconds

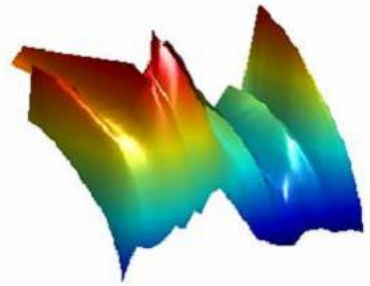
Databases Applications

- Microsoft SQL Server … COM component generated with MATLAB Builder™ NE

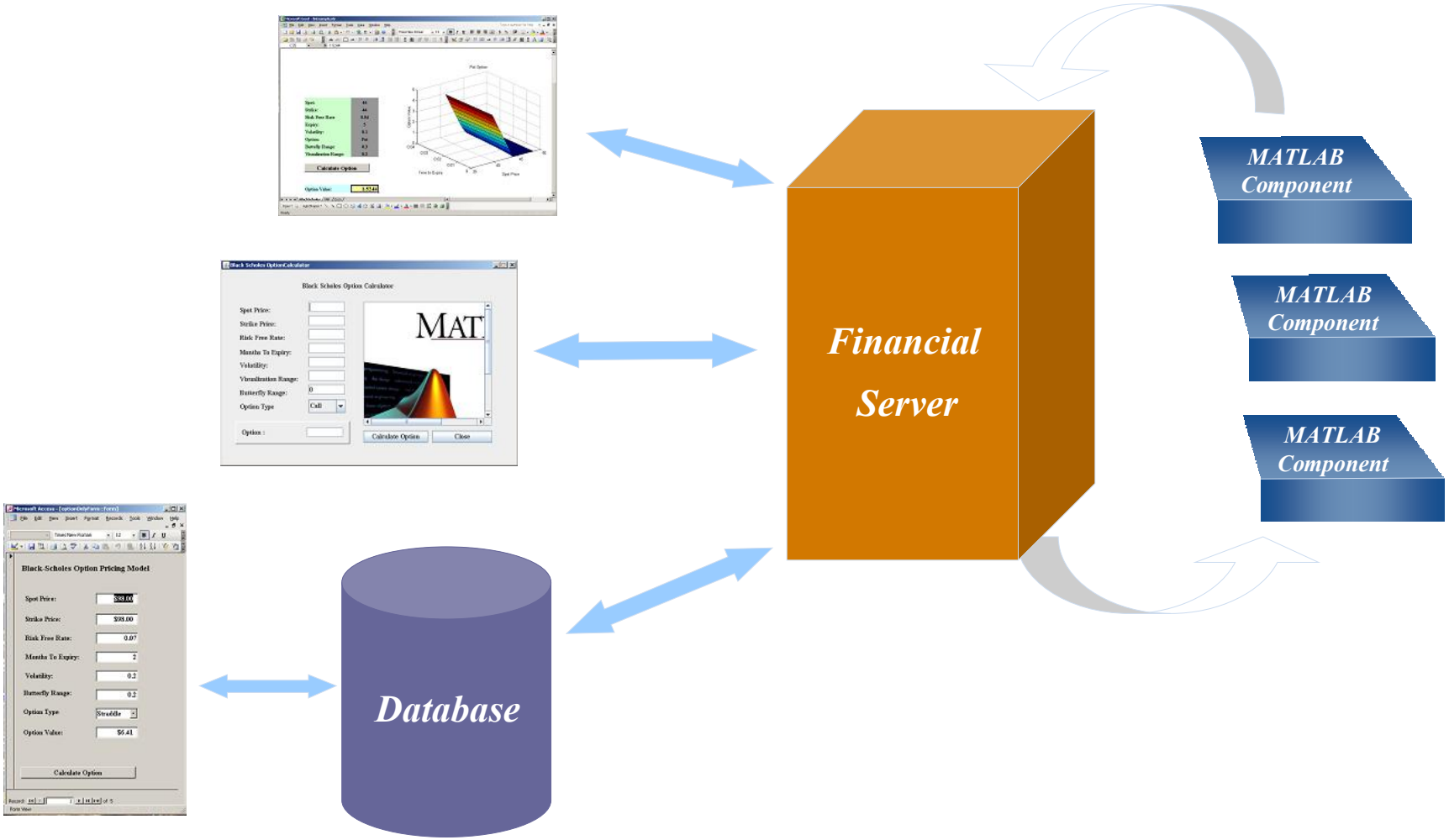
- Oracle Database … COM component generated with MATLAB Builder™ NE



- Oracle Database … Java class generated with MATLAB Builder™ JA



Example 2 Financial Server/Desktop Applications



MATLAB Component and PERL

```
use Win32::OLE;
use Win32::OLE::Variant;

$ex = Win32::OLE->new('perlBls.perlBlsclass') or die "Oops, cannot load Object";

# Option Model loaded

# define input values

$spot = Variant(VT_R8, 44);
$strike = Variant(VT_R8,44);
$riskFRate = Variant(VT_R8, 0.03);
$monthsToExpiry = Variant(VT_R8, 3);
$volatility = Variant(VT_R8, 0.2);
$vizRange = Variant(VT_R8, 0.1);
$bflyRange = Variant(VT_R8, 0);
$optionType = Variant(VT_BSTR, "Straddle");

# output value

$optionValueReturned = Win32::OLE::Variant->new(VT_VARIANT|VT_BYREF,0);

# Invoke the method

$ex->OptionValue(1,$optionValueReturned,$spot,$strike,$riskFRate,$monthsToExpiry,$volatility,"Call",$bflyRange);

#print  $optionValueReturned->Get, "\n\n";

#$ex->perlvizroutine(1,$theFilename,$spot,$strike,$riskFRate,$monthsToExpiry,$volatility,$vizRange,"put",$bflyRange);

print "\nThe Option Value returned is : $optionValueReturned \n";
```

Componentization .. Embed MATLAB algorithms Across the Enterprise