

## Using FINCAD® Developer with Maple Software

Visit [www.fincad.com](http://www.fincad.com) to download a free 7-day trial version of FINCAD Developer, a financial analytics library containing over 1,400 financial functions used for valuing and measuring the risk of financial securities and derivatives.

### Maple Overview

Maple is a mathematical software package that was originally developed as an advanced research project at the University of Waterloo in 1980. The purpose was to develop a language that allowed students in Mathematics, Engineering, and Science courses to develop solutions to algebraic problems. In 1988 Waterloo Maple Software was founded to sell and enhance the Maple modeling environment. The product is currently sold by MapleSoft which is a division of the Waterloo Maple Software company.

Maple is available on a variety of platforms including Windows, UNIX and Linux. It is a great tool for developing mathematical programs because of the large number of built-in functions and mathematical operations that are available.

The company has formed research partnerships with various academic institutions including the University of Waterloo, University of Western Ontario, Simon Fraser University, Ontario Research Center for Computer Algebra, Florida State University, Southern Methodist University, ETH, Moscow State University and INRIA. The company also has partnerships with various software companies including National Instruments, Mathworks, SolidWorks, NAG and Thomson Learning. You can find out more about Maple by going to the company's website <http://www.maplesoft.com>

### Purpose of Article

The focus of this article is on calling FINCAD Developer functions from Maple using the Java interface. Making external functions available in Maple is done by using the `define_external` routine in the package. The example below will create a package with an interface to the `aaAccrual_factor` function using FINCAD Developer Version 9. To explore how you could use FINCAD Developer's functions, please download a free 7-day trial of FINCAD Developer.

The module below can be treated as one executable statement in Maple. The `'>'` character will be used to indicate statements which need to be entered on a separate line. This will be familiar for users of the classic interface for Maple.

```
FINCAD := module()
  option package;

  export aaAccrual_factor;

  local aaAccrual_factor_ext,FC_CLASSPATH;

  #create a link to the math library through the java jar file
  FC_CLASSPATH:="c:/Program Files/FINCAD/fcm19/lib/fincad.jar";

  #setup the external function call
  aaAccrual_factor_ext := define_external(
    'aaAccrual_factor_java',
    CLASS="com.fincad.ShowAll",
    CLASSPATH=FC_CLASSPATH,
    JAVA,
    'd_e':float[8],
    'd_t':float[8],
    'acc':integer[4],
    RETURN::ARRAY(datatype=float[8])
  );

  #call the function
  aaAccrual_factor := proc(d_e::float,d_t::float,acc::integer)::float;
```

```

local ret::Array(datatype=float);
ret := aaAccrual_factor_ext(d_e,d_t,acc);
if ret[1] = 16 then error "fincad error" fi;
ret[2];
end proc;

```

end module:

A good way to use this new package is to save the module as a library so that it can be referenced from any maple code that you may be writing. Use commands like the ones below to save the library:

```

>savelibname := "C:/fincad.lib":
>savelib('FINCAD'):

```

The two lines above save the code as an add-in library which allows the code to be loaded on demand. Use the with command below to load the library and then the following two commands to run the function and display the results:

```

>with(FINCAD):
>return_value:=aaAccrual_factor(34236.0, 35000.0, 2):
>printf("%f\n",return_value):

```

The function will display the accrual factor between "24-Sep-93" and "28-Oct-95". The input numbers in the formula are the serialdate values for these dates. You can use the function aaDateSerial to calculate a serialdate given a value for the year, month and day.

Following the steps above allows you to call a simple FINCAD function from Maple using the Java interface. Below is an example of calling a more complex function.

The restart: command will clear the previous example from memory.

```

>restart:
> FINCAD := module()
  option package;

export aaBSG;
local FC_CLASSPATH;

#create a link to the math library through the java jar file
FC_CLASSPATH:="c:/Program Files/FINCAD/fcm19/lib/fincad.jar";

aaBSG := define_external(
  'aaBSG_java',
  CLASS="com.fincad.ShowAll",
  CLASSPATH=FC_CLASSPATH,
  JAVA,
  'price_u'::float[8],
  'price_e'::float[8],
  'd_e'::float[8],
  'd_v'::float[8],
  'vlt'::float[8],
  'rate'::float[8],
  'hc'::float[8],
  'option_type'::integer[4],
  'stats'::ARRAY(1..1,1..m,datatype=float[8]),
  'accrual_riskless'::integer[4],
  'acc_hc'::integer[4],
  RETURN::ARRAY(datatype=float[8])
);

```

end module:

```
> with(FINCAD);
> stat:=Array(1..1,1..1,(i,j)->1,datatype=float[8],order=C_order);
> return_value:=aaBSG(60,60,36526,36161,0.14,0.05,0.0,1,stat,1,1);
> printf("%f\n",return_value);
```

The above commands will call the function aaBSG and return the results into the variable return\_value. This function call is slightly more complex due to the variable return\_value being an array and due to the need to setup one of the inputs (stat) as an array. The printf statement will display the entire contents of the array.

The array will hold the values 64.000000 1.000000 1.000000 4.893370

The first array input indicates that the return\_value variable is an array of data. The following two values provide the number of rows and number of columns of data.

The result of the function call is actually in the fourth array field. Below is an example of calling the function with a larger input stat list:

```
> stat:=Array(1..1,1..8,(i,j)->j,datatype=float[8],order=C_order);
> return_value:=aaBSG(60,60,36526,36161,0.14,0.05,0.0,1,stat,1,1);
> printf("%f\n",return_value);
```

The return\_value array is now a larger array of data and holds the values below

64.000000 1.000000 8.000000 4.893370 0.662210 0.043511 -0.008863 0.219295 0.331802 -0.397326 0.609686

If the first array value is 16 this indicates that the function call has returned an error code. The second array field will contain the error code.

There are two different end-of-line characters that you can use in Maple. When you use the “.” character the instruction will be executed in memory with no output displayed in the command window. When you use the “;” character the result of the instruction will be written to the command window. The additional functionality provided by using the “;” character can be useful when developing new code.

For example, you could replace the three instructions above with the statements below

```
> stat:=Array(1..1,1..8,(i,j)->j,datatype=float[8],order=C_order);
> return_value:=aaBSG(60,60,36526,36161,0.14,0.05,0.0,1,stat,1,1);
> printf("%f\n",return_value);
```

The result will be that only the third instruction will display the results in the command window. You can also output a single value by using a command like the one below:

```
>return_value[1];
```

This command will display the first element in the array to the command window.

You can get the number of rows in the array by using the formula below. Subtract three to ignore the type flag and the number of rows and columns:

```
>ArrayNumElems(return_value)-3;
```

You can get the number of dimensions of the table using the command:

```
>ArrayNumDims(return_value);
```

Once you know the size of the return array, you know which elements can be indexed and you can assign particular array elements to other variables. Calling other functions involves going through a similar process to what has been done in the two examples above.

**Disclaimer**

Your use of the information in this article is at your own risk. The information in this article is provided on an "as is" basis and without any representation, obligation, or warranty from FINCAD of any kind, whether express or implied. We hope that such information will assist you, but it should not be used or relied upon as a substitute for your own independent research.

Copyright © 2006 FinancialCAD Corporation. All rights reserved. FinancialCAD® and FINCAD® are registered trademarks of FinancialCAD Corporation. Other trademarks are the property of their respective holders. This email is for informational purposes only. FinancialCAD Corporation MAKES NO WARRANTIES, EXPRESSED OR IMPLIED, IN THIS SUMMARY.