

How to use the unit tests for the Benchop complex?

1 Local Stochastic Volatility

1. Make sure that the unit test function is in the folder with all your functions or has an access to the folder via the correct path.
2. The unit test takes two variables: the method handle `MTH` and the tolerance `tol`. The method handle should be a string and the tolerance is a real number.
Example: `MTH = 'RBFPU'`; `tol=0.1`; The `MTH` will specify your method and the `tol` will set a tolerance, and you will be informed by a warning message if the error exceeds the tolerance.

3. The unit test give out one out put variable `pr` that is a structure with the following fields

`pr.name` - name of the problem being tested

`pr.refval` - reference values for the problem

`pr.err` - error (absolute) that your method makes

All the data contained in the structure will be printed on your screen.

4. The unit test can be called in the following way

```
pr = unitTestBenchopLSV(MTH,tol);
```

5. First, the unit test will check the existence of the files and will return a message either of the type

`SABReuCallI_RBFPUM exists and will be tested` if the file exists

or

`SABReuCallI_RBFPUM does NOT exist` if the file does not exist in the folder.

6. Then the test will move onto testing the existing files. You will be informed on which of the files is being tested right now by a message of this type

`Testing SABReuCallI_RBFPUM...`

7. If your function has an error the unit test will not break. Instead it will move to the following function. But it will capture the error message and display it in the command line.

Example: If you had a dimension mismatch somewhere in the code the following message will be printed

`!!!`

`Failed to run`

`Matrix dimensions must agree.`

`!!!`

8. Hopefully your code is error free. In this case the function will be executed and the output will be compared with the reference value. Your output must be a row-vector of a certain length depending on the problem specification. The size of the output must match the size of the reference value vector. If it does not match then the following warning message will be printed

`The output has incorrect format. It must be a vector 1x3`

and you will have to adjust the size of your output.

9. If the sizes match then the absolute error can be computed. If the absolute error is larger than the selected tolerance you will see the message

`The absolute error is larger than the chosen tolerance`

10. If the error fulfils the selected tolerance you will see the message

`All correct`

11. After the testing has been completed the function will print

`TESTING COMPLETED`

12. Then it will print the output in the format that contains the function name and the error

`Function name: SABReuCallI_RBFPUM`

`ERR = 0.9717 0.9634 0.9336`

2 Basket Options

1. Make sure that the unit test function is in the folder with all your functions or has an access to the folder via the correct path.
2. The unit test takes two variables: the method handle `MTH` and the tolerance `tol`. The method handle should be a string and the tolerance is a real number.
Example: `MTH = 'RBFPU'`; `tol=0.1`; The `MTH` will specify your method and the `tol` will set a tolerance, and you will be informed by a warning message if the error exceeds the tolerance.
3. The unit test give out one out put variable `pr` that is a structure with the following fields
`pr.name` - name of the problem being tested
`pr.refval` - reference values for the problem
`pr.err` - error (absolute) that your method makes
All the data contained in the structure will be printed on your screen.
4. The unit test can be called in the following way
`pr = unitTestBenchopBASKET(MTH,tol);`
5. First, the unit test will check the existence of the files and will return a message either of the type
`BSamPut10DbasketHVCU_RBFPU exists and will be tested` if the file exists
or
`BSamPut10DbasketHVCU_RBFPU does NOT exist` if the file does not exist in the folder.
6. Then the test will move onto testing the existing files. You will be informed on which of the files is being tested right now by a message of this type
`Testing BSamPut10DbasketHVCU_RBFPU...`
7. If your function has an error the unit test will not break. Instead it will move to the following function. But it will capture the error message and display it in the command line.

Example: If you had a dimension mismatch somewhere in the code the following message will be printed

```
!!!
```

```
Failed to run
```

```
Matrix dimensions must agree.
```

```
!!!
```

8. Hopefully your code is error free. In this case the function will be executed and the output will be compared with the reference value. Your output must be a row-vector of a certain length depending on the problem specification. The size of the output must match the size of the reference value vector. If it does not match then the following warning message will be printed

```
The output has incorrect format
```

and you will have to adjust the size of your output.

9. If the sizes match then the absolute error can be computed. If the absolute error is larger than the selected tolerance you will see the message

```
The absolute error is larger than the chosen tolerance
```

10. If the error fulfils the selected tolerance you will see the message

```
All correct
```

11. After the testing has been completed the function will print

```
TESTING COMPLETED
```

12. Then it will print the output in the format that contains the function name and the error

```
Function name:  BSamPut10DbasketHVCU_RBFPUM
```

```
ERR = 0
```