

# How to lift VBA User- Defined Functions over pairwise arrays



Suppose we have these two very simple input arrays

	A	B	C	D
1	x		y	
2		1		-1
3		2		-2
4		3		-3
5		4		-4
6		5		-5
7		6		-6
8		7		-7
9		8		-8
10		9		-9
11		10		-10
12				

## And suppose we have some very simple User-Defined Functions with two parameters

Option Explicit

```
Public Function multiply(x As Double, y As Double)
```

```
multiply = x * y
```

```
End Function
```

```
Public Function add(x As Double, y As Double)
```

```
add = x + y
```

```
End Function
```

**We can call the functions row by row  
with no problem**

	A	B	C	D	E	F	G	H	I
1	x		y						
2	1		-1		-1	=multiply(A2,C2)		0	=add(A2,C2)
3	2		-2		-4	=multiply(A3,C3)		0	=add(A3,C3)
4	3		-3		-9	=multiply(A4,C4)		0	=add(A4,C4)
5	4		-4		-16	=multiply(A5,C5)		0	=add(A5,C5)
6	5		-5		-25	=multiply(A6,C6)		0	=add(A6,C6)
7	6		-6		-36	=multiply(A7,C7)		0	=add(A7,C7)
8	7		-7		-49	=multiply(A8,C8)		0	=add(A8,C8)
9	8		-8		-64	=multiply(A9,C9)		0	=add(A9,C9)
10	9		-9		-81	=multiply(A10,C10)		0	=add(A10,C10)
11	10		-10		-100	=multiply(A11,C11)		0	=add(A11,C11)

But what if we want to “lift” the function over the two ranges and spill the result?

This doesn't work.

	A	B	C	D	E	F	G	H	I
1	x		y						
2	1		-1		#VALUE!	=multiply(A2:A11,C2:C11)		#VALUE!	=add(A2:A11,C2:C11)
3	2		-2						
4	3		-3						
5	4		-4						
6	5		-5						
7	6		-6						
8	7		-7						
9	8		-8						
10	9		-9						
11	10		-10						

## For simple functions with a known count of parameters, we can construct a “lift” function to do the work

```
Public Function lift(func As String, ParamArray arrs() As Variant)
    Dim i As Long
    Dim leadingArray As Variant
    Dim result As Variant
    Dim resultLength As Long

    leadingArray = arrs(0)
    resultLength = UBound(leadingArray)

    ReDim result(1 To resultLength, 1 To 1)

    For i = 1 To resultLength
        result(i, 1) = Application.Run(func, arrs(0)(i, 1), arrs(1)(i, 1))
    Next i

    lift = result
End Function
```

**This allows us to pass the name of the UDF we want to lift as an argument to the lift UDF, and the function spills**

	A	B	C	D	E	F	G	H	I
1	x		y						
2	1		-1		-1	=lift("multiply",A2:A11,C2:C11)		0	=lift("add",A2:A11,C2:C11)
3	2		-2		-4			0	
4	3		-3		-9			0	
5	4		-4		-16			0	
6	5		-5		-25			0	
7	6		-6		-36			0	
8	7		-7		-49			0	
9	8		-8		-64			0	
10	9		-9		-81			0	
11	10		-10		-100			0	

## But this is fraught with problems...

Relies on correctly spelled  
function name 🦴

```
Public Function lift(func As String, ParamArray arrs() As Variant)
    Dim i As Long
    Dim leadingArray As Variant
    Dim result As Variant
    Dim resultLength As Long

    leadingArray = arrs(0)
    resultLength = UBound(leadingArray)

    ReDim result(1 To resultLength, 1 To 1)

    For i = 1 To resultLength
        result(i, 1) = Application.Run(func, arrs(0)(i, 1), arrs(1)(i, 1))
    Next i

    lift = result
End Function
```

Need to add code to  
check for same-sized  
arrays in arrs() 🤔

Can't pass ParamArray  
to Application.Run 🤔,  
each argument must  
be passed individually



## And anyway, we have MAP for lifting functions...

	A	B	C	D	E	F	G	H	I
1	x		y						
2	1		-1		-1	=MAP(A2:A11,C2:C11,LAMBDA(x,y,multiply(x,y)))		0	=MAP(A2:A11,C2:C11,LAMBDA(x,y,add(x,y)))
3	2		-2		-4			0	
4	3		-3		-9			0	
5	4		-4		-16			0	
6	5		-5		-25			0	
7	6		-6		-36			0	
8	7		-7		-49			0	
9	8		-8		-64			0	
10	9		-9		-81			0	
11	10		-10		-100			0	

Or if you think you'll be doing a lot of this kind of thing, curry the function so you can pass it to MAP more simply




	A	B	C	D	E	F	G	H	I	J
1	x	y								
2	1		-1		-1	=MAP(A2:A11,C2:C11,two_arg(multiply))		0	=MAP(A2:A11,C2:C11,two_arg(add))	
3	2		-2		-4			0		
4	3		-3		-9			0		
5	4		-4		-16			0		
6	5		-5		-25			0		
7	6		-6		-36			0		
8	7		-7		-49			0		
9	8		-8		-64			0		
10	9		-9		-81			0		
11	10		-10		-100			0		
12										
13										
14										

Excel Labs

Grid

Names

Modules

Workbook 

+

 New

1

two\_arg = LAMBDA(function,

2

LAMBDA(x, y,

3

function(x, y)

4

)

5

);

**These are trivial examples. Nobody would create a “multiply” UDF.**

**But you might create something more complex as a UDF which is either very difficult or not possible with a formula.**

**So, what’s the point?**

**You're not limited to just one paradigm. Just because you wrote the scalar function in VBA, doesn't mean you have to lift it with VBA.**

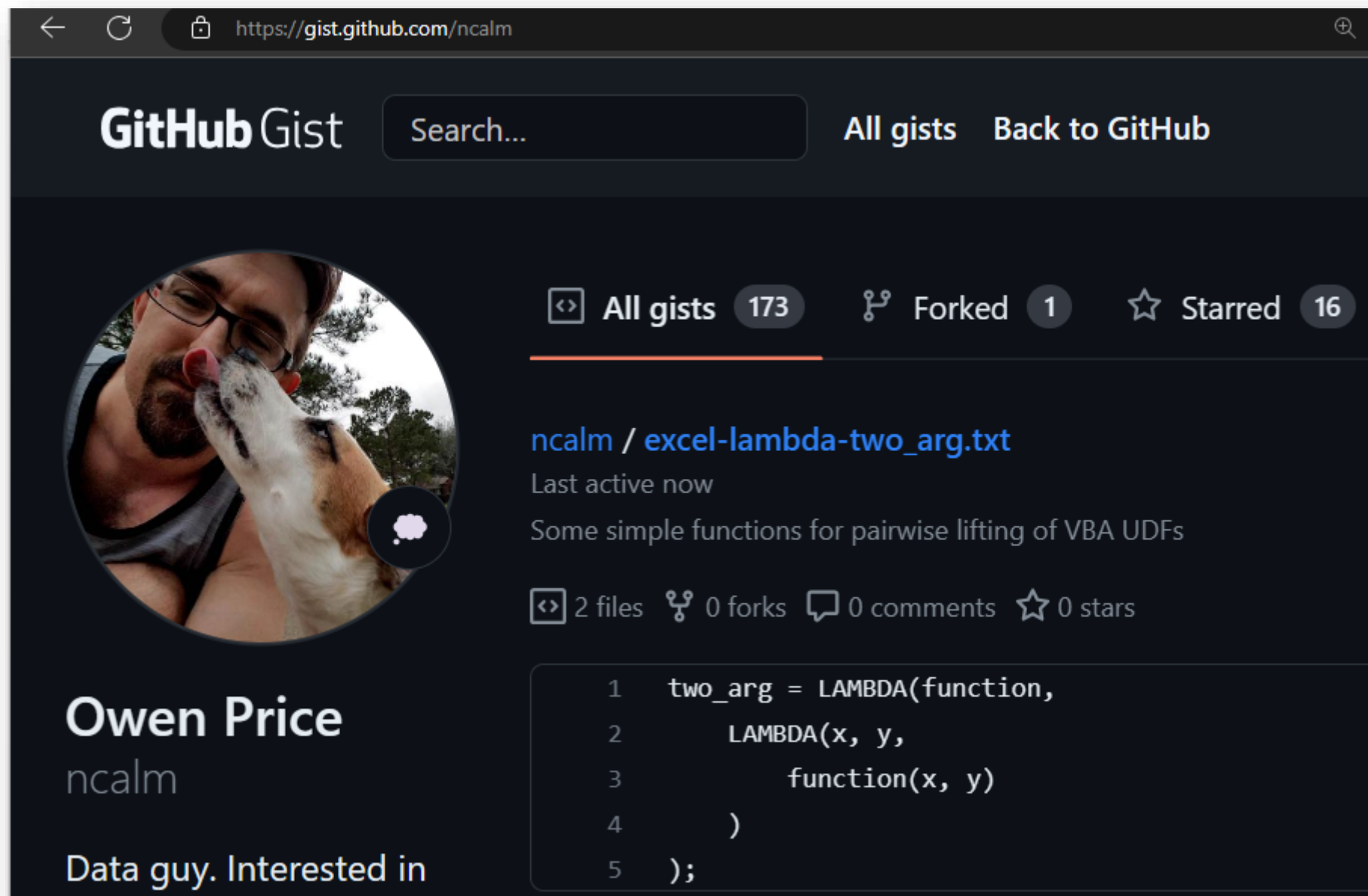
**Combine the tools available to get the job done!**



Interested? Grab the code!


<http://gist.github.com/ncalm>

Other links: <https://linktr.ee/flexyourdata>



The screenshot shows a web browser window displaying a GitHub Gist page. The address bar shows the URL `https://gist.github.com/ncalm`. The page header includes the "GitHub Gist" logo, a search bar, and links for "All gists" and "Back to GitHub". On the left, there is a circular profile picture of a man with glasses and a beard, identified as Owen Price (ncalm), with a bio that says "Data guy. Interested in". To the right of the profile picture, there are statistics for the user's gists: "All gists 173", "Forked 1", and "Starred 16". Below these statistics, the specific gist is titled "ncalm / excel-lambda-two\_arg.txt" and is noted as "Last active now". The description of the gist is "Some simple functions for pairwise lifting of VBA UDFs". At the bottom of the gist header, it shows "2 files", "0 forks", "0 comments", and "0 stars". The main content area displays a VBA code snippet for a lambda function named "two\_arg".

GitHub Gist Search... All gists Back to GitHub

 All gists 173 Forked 1 Starred 16

ncalm / excel-lambda-two\_arg.txt  
Last active now  
Some simple functions for pairwise lifting of VBA UDFs

2 files 0 forks 0 comments 0 stars

```
1 two_arg = LAMBDA(function,  
2     LAMBDA(x, y,  
3         function(x, y)  
4     )  
5 );
```