Subject: Re: Speed does matter
Posted by on Thu, 04 Oct 2012 13:24:25 GMT
View Forum Message <> Reply to Message

> Similar benchmarks were computed on another computer between Python/Numpy MKLPython/Numpy MKL and Matlab, demonstrating other artifacts but mostly with "comparable" performances (in particular with Python MKL 64 bits). These results highlight the incredible performances impact of the Intel Math Kernel Library, in particular here in linear algebra routines. Since this Library is a Royalty-free, per developer licensing, I'd dream to see a future IDL compilation against such Library.
>
> Any chances ?

By the way, Matlab is indeed using Intel's MKL
(source :http://software.intel.com/en-us/articles/using-intel-mkl-wi th-matlab/).

Subject: Re: Speed does matter
Posted by on Thu, 13 Aug 2015 07:31:35 GMT
View Forum Message <> Reply to Message

I just run the same test on the official "unofficial" IDL 8.5 which promised some speed improvement. Strangely enough no such claims appeared in the "What's new" and indeed results of the above tests are strictly similar in IDL 8.5 ...

The good news is that we can now switch to Python MKL inside IDL when dealing with poor performances.

Best

Subject: Re: Speed does matter
Posted by on Thu, 13 Aug 2015 07:33:20 GMT
View Forum Message <> Reply to Message

I just ran the same tests on the official "unofficial" IDL 8.5 which promised some speed improvement. Strangely enough no such claims appeared in the "What's new" and indeed results of the above tests are strictly similar in IDL 8.5 ...

The good news is that we can now switch to Python MKL inside IDL when dealing with poor performances.

Best

Subject: Re: Speed does matter
Posted by chris_torrence@NOSPAM on Thu, 13 Aug 2015 21:19:56 GMT
View Forum Message <> Reply to Message

On Thursday, August 13, 2015 at 1:33:24 AM UTC-6, Kallisthène wrote:
> I just ran the same tests on the official "unofficial" IDL 8.5 which promised some speed improvement. Strangely enough no such claims appeared in the "What's new" and indeed results of the above tests are strictly similar in IDL 8.5 ...
>
> The good news is that we can now switch to Python MKL inside IDL when dealing with poor performances.
>
>
> Best

Hi Kallisthène,

Well, you posts contain a lot of information. If I could sum them up, I would say "IDL is faster at some computations, Python or Matlab is faster at others."

It really does depend upon your code and your algorithm. To make a sweeping generalization, IDL's interpreter will be faster than Python's for "normal" problems - things with lots of "for" loops, small-to-medium size arrays, and image processing. Python and Matlab will be faster for hard-core linear algebra problems with large matrices.

The chances of compiling IDL against the Intel Math Kernel library is low - the IDL team isn't huge, and we have a lot of pending features on our plate.

So I think your solution is a good one. Use IDL as your general purpose scripting engine, input/output of data, use it for medium-size arrays. Then use the Python bridge to process your large arrays.

I'd love to see real-world examples of using the Python bridge, so please post again!

Cheers,
Chris
VIS/Exelis/Harris

Subject: Re: Speed does matter
Posted by chris_torrence@NOSPAM on Thu, 13 Aug 2015 21:21:10 GMT
View Forum Message <> Reply to Message

On Thursday, August 13, 2015 at 1:33:24 AM UTC-6, Kallisthène wrote:
> I just ran the same tests on the official "unofficial" IDL 8.5 which promised some speed improvement. Strangely enough no such claims appeared in the "What's new" and indeed results of the above tests are strictly similar in IDL 8.5 ...
>

> The good news is that we can now switch to Python MKL inside IDL when dealing with poor performances.
>
>
> Best

Just out of curiosity, who promised the speed improvements? Hope it wasn't me. :-)
-C

Subject: Re: Speed does matter
Posted by on Fri, 14 Aug 2015 11:59:48 GMT
View Forum Message <> Reply to Message

Le jeudi 13 août 2015 23:21:12 UTC+2, Chris Torrence a écrit :
> On Thursday, August 13, 2015 at 1:33:24 AM UTC-6, Kallisthène wrote:
>> I just ran the same tests on the official "unofficial" IDL 8.5 which promised some speed improvement. Strangely enough no such claims appeared in the "What's new" and indeed results of the above tests are strictly similar in IDL 8.5 ...
>>
>> The good news is that we can now switch to Python MKL inside IDL when dealing with poor performances.
>>
>>
>> Best
>
> Just out of curiosity, who promised the speed improvements? Hope it wasn't me. :-)
> -C

Well, it originated in one of Mark Piper post (
https://groups.google.com/forum/#!searchin/comp.lang.idl-pvw
ave/Mark$20Piper$208.4/comp.lang.idl-pvwave/bEr8Bh5iLrI/ASa9 4eksIbcJ).

Best

Subject: Re: Speed does matter
Posted by on Fri, 14 Aug 2015 12:16:44 GMT
View Forum Message <> Reply to Message

Hi Chris,

thanks for your answer.

> Hi Kallisthène,

>
> Well, you posts contain a lot of information. If I could sum them up, I would say "IDL is faster at some computations, Python or Matlab is faster at others."
>
> It really does depend upon your code and your algorithm. To make a sweeping generalization, IDL's interpreter will be faster than Python's for "normal" problems - things with lots of "for" loops, small-to-medium size arrays, and image processing. Python and Matlab will be faster for hard-core linear algebra problems with large matrices.

Well, that's indeed vastly sweeping. When I watch my computer cores while executing IDL codes containing a lot of matrix operations among those I tested, I see only one core in action. The Thread pool seems to me a "theoretical" solution with not that much impact in the real world.

Meanwhile Python (and Matlab I think) "MKLed" enjoy a full 100% CPU occupation. It means that IDL still hasn't fully embraced the multi-processor revolution (like a lot of software ecosystems, I must admit)

>
> The chances of compiling IDL against the Intel Math Kernel library is low - the IDL team isn't huge, and we have a lot of pending features on our plate.

That I can understand, to change your compilation environment can be a real mess. But hasn't a single guy recompiled Python and numerous libraries with MKL library ?
(see http://www.lfd.uci.edu/~gohlke/pythonlibs/)

>
> So I think your solution is a good one. Use IDL as your general purpose scripting engine, input/output of data, use it for medium-size arrays. Then use the Python bridge to process your large arrays.

By the way I just ran into a problem in this respect, I can't seem to be able to load matplotlib.pyplot as in your example. I've got "% PYTHON_IMPORT: Exception: No module named Tkinter." error message.
Since enough modules do work well, I wonder if it might come from the fact that this module has another name in Python 3 : tkinter ?
What do you think ?

Best regards

>
> I'd love to see real-world examples of using the Python bridge, so please post again!
>
> Cheers,
> Chris

> VIS/Exelis/Harris

Subject: Re: Speed does matter
Posted by chris_torrence@NOSPAM on Thu, 20 Aug 2015 17:47:02 GMT
View Forum Message <> Reply to Message

On Friday, August 14, 2015 at 6:16:46 AM UTC-6, Kallisthène wrote:
>
> By the way I just ran into a problem in this respect, I can't seem to be able to load matplotlib.pyplot as in your example. I've got "% PYTHON_IMPORT: Exception: No module named Tkinter." error message.
> Since enough modules do work well, I wonder if it might come from the fact that this module has another name in Python 3 : tkinter ?
> What do you think ?
>

Hi Kallisthène,

That's a strange error. We're not using the "Tkinter" module, at least not explicitely. Maybe it's being used by matplotlib under the covers. But if that were the case, you would think that matplotlib would have spelled the name correctly. Is it possible that you have some sort of mixed Python environment, where you have a Python 2.7 version of one module that has been installed into Python 3 (or vice versa)?

Just an aside, regarding the "one guy" recompiling Python, I just looked at that page and it says... "The files are provided "as is" without warranty or support of any kind. The entire risk as to the quality and performance is with you."

So, your mileage may vary. At least with IDL, you can call or email Tech Support and talk to someone. :-)

Cheers,
Chris

Subject: Re: Speed does matter
Posted by on Fri, 21 Aug 2015 08:09:10 GMT
View Forum Message <> Reply to Message

Le jeudi 20 août 2015 19:47:05 UTC+2, Chris Torrence a écrit :
> On Friday, August 14, 2015 at 6:16:46 AM UTC-6, Kallisthène wrote:
>>
>> By the way I just ran into a problem in this respect, I can't seem to be able to load matplotlib.pyplot as in your example. I've got "% PYTHON_IMPORT: Exception: No module named Tkinter." error message.
>> Since enough modules do work well, I wonder if it might come from the fact that this module

has another name in Python 3 : tkinter ?
>> What do you think ?
>>
>
> Hi Kallisthène,
>
> That's a strange error. We're not using the "Tkinter" module, at least not explicitely. Maybe it's being used by matplotlib under the covers. But if that were the case, you would think that matplotlib would have spelled the name correctly. Is it possible that you have some sort of mixed Python environment, where you have a Python 2.7 version of one module that has been installed into Python 3 (or vice versa)?

Well, these days there are many software using python and thus silently installing it. But I use Winpython (natively MKL) 2.7 which I registered explicitly. And indeed matplolib calls tkinter, we tried to use Qt instead but without success yet. The bug is still a mystery since tkinter works well from winpython.

>
> Just an aside, regarding the "one guy" recompiling Python, I just looked at that page and it says... "The files are provided "as is" without warranty or support of any kind. The entire risk as to the quality and performance is with you."
>
> So, your mileage may vary. At least with IDL, you can call or email Tech Support and talk to someone. :-)

The argument is difficult to understand, I don't believe that any flavor of Python does indeed promise any warranty or support of any kind. Then why did you provide a bridge to an unreliable software ?

On the other side our experiences with corporate software is much worse. One reason we switched from Pv-Wave to IDL was that they were sitting on a 10 years old known FFT error without taking any steps to correct it !

For the reliability to become a valuable service, you need to communicate heavily on the methods you use to guarantee it.

Best regards

>
> Cheers,
> Chris

Subject: Re: Speed does matter

> On Thursday, August 13, 2015 at 1:33:24 AM UTC-6, Kallisthène wrote:
>> I just ran the same tests on the official "unofficial" IDL 8.5 which promised some speed improvement. Strangely enough no such claims appeared in the "What's new" and indeed results of the above tests are strictly similar in IDL 8.5 ...
>>
>> The good news is that we can now switch to Python MKL inside IDL when dealing with poor performances.
>>
>>
>> Best
>
> Hi Kallisthène,
>
> Well, you posts contain a lot of information. If I could sum them up, I would say "IDL is faster at some computations, Python or Matlab is faster at others."
>
> It really does depend upon your code and your algorithm. To make a sweeping generalization, IDL's interpreter will be faster than Python's for "normal" problems - things with lots of "for" loops, small-to-medium size arrays, and image processing. Python and Matlab will be faster for hard-core linear algebra problems with large matrices.

IDL's for loops very slow comparesion with matlab.
for example

IDL(8.5) code

```
nx = 500
ny = 500
nz = 500

arr = dblarr(nx, ny, nz)


tic
for z = 0, nz - 1 do begin
   for y = 0, ny - 1 do begin
      for x = 0, nx - 1 do begin
         arr[x, y, z] = 1
      endfor
   endfor
endfor
toc
```

And MATLAB(2014a) code

```
nx = 500;
ny = 500;
nz = 500;

arr = zeros(nx, ny, nz);

tic
for z=1:nz
   for y = 1:ny
      for x = 1:nx
         arr(x,y,z) =  1;
      end
   end
end
toc
```

The result, on my desktop
IDL -- 5.16 second
MATLAB --0.45 second

Roughly, MATLAB's for loop x10 faster.

>
> The chances of compiling IDL against the Intel Math Kernel library is low - the IDL team isn't huge, and we have a lot of pending features on our plate.
>
> So I think your solution is a good one. Use IDL as your general purpose scripting engine, input/output of data, use it for medium-size arrays. Then use the Python bridge to process your large arrays.
>
> I'd love to see real-world examples of using the Python bridge, so please post again!
>
> Cheers,
> Chris
> VIS/Exelis/Harris

Subject: Re: Speed does matter
Posted by markb77 on Tue, 25 Aug 2015 08:32:20 GMT
View Forum Message <> Reply to Message

Nice comparison. On my desktop, running IDL 8.5 and Matlab 2010b, I get the following results:

IDL: % Time elapsed: 6.4400001 seconds.

MATLAB 2010b: Elapsed time is 0.594112 seconds.

Again, MATLAB for loops run faster by a factor of ~10.

Subject: Re: Speed does matter
Posted by on Tue, 25 Aug 2015 13:03:12 GMT
View Forum Message <> Reply to Message

Le mardi 25 août 2015 10:32:22 UTC+2, superchromix a écrit :
> Nice comparison. On my desktop, running IDL 8.5 and Matlab 2010b, I get the following results:
>
> IDL: % Time elapsed: 6.4400001 seconds.
>
> MATLAB 2010b: Elapsed time is 0.594112 seconds.
>
> Again, MATLAB for loops run faster by a factor of ~10.

Great idea,

I ran also these codes, with bigger numbers, 1000 instead of 500 for each dimension and with Matlab 2015a I got 6.3 secondes to compare with 102 secondes on my IDL 8.5.
My desktop runs 4 cores and has 12 Go RAM and it runs faster with Matlab by a factor of 16 ! Damn.

Subject: Re: Speed does matter
Posted by Michael Galloy on Tue, 25 Aug 2015 21:35:07 GMT
View Forum Message <> Reply to Message

On 8/25/15 7:03 AM, Kallisthène wrote:
> Le mardi 25 août 2015 10:32:22 UTC+2, superchromix a écrit :
>> Nice comparison. On my desktop, running IDL 8.5 and Matlab 2010b,
>> I get the following results:
>>
>> IDL: % Time elapsed: 6.4400001 seconds.
>>
>> MATLAB 2010b: Elapsed time is 0.594112 seconds.
>>
>> Again, MATLAB for loops run faster by a factor of ~10.
>
> Great idea,
>
>
> I ran also these codes, with bigger numbers, 1000 instead of 500 for

> each dimension and with Matlab 2015a I got 6.3 secondes to compare
> with 102 secondes on my IDL 8.5. My desktop runs 4 cores and has 12
> Go RAM and it runs faster with Matlab by a factor of 16 ! Damn.
>

For what it's worth, I got about a 5x speedup of IDL over Python. I used
1000 elements for each dimension:

```
$ python mg_loopspeed.py
222.15 secs
```

Code is:

```
  import numpy as np
  import time

  nx = 1000
  ny = 1000
  nz = 1000

  arr = np.zeros((nx, ny, nz))

  tic = time.clock()

  for z in xrange(nz):
      for y in xrange(ny):
          for x in xrange(nx):
              arr[x, y, z] = 1.0

  toc = time.clock()
  print '%0.2f secs' % (toc - tic)
```

For comparison, my IDL time:

```
$ idl -e ".run mg_loopspeed"
IDL Version 8.5, Mac OS X (darwin x86_64 m64).
(c) 2015, Exelis Visual Information Solutions, Inc., a subsidiary of
Harris Corporation.

% Compiled module: $MAIN$.
% Compiled module: TIC.
  Elapsed Time:  47.486635
```

Mike
--
Michael Galloy
www.michaelgalloy.com
Modern IDL: A Guide to IDL Programming (http://modernidl.idldev.com)

Subject: Re: Speed does matter
Posted by on Wed, 26 Aug 2015 08:41:31 GMT
View Forum Message <> Reply to Message

Le mardi 25 août 2015 23:35:11 UTC+2, Michael Galloy a écrit :
> On 8/25/15 7:03 AM, Kallisthène wrote:
>> Le mardi 25 août 2015 10:32:22 UTC+2, superchromix a écrit :
>>> Nice comparison. On my desktop, running IDL 8.5 and Matlab 2010b,
>>> I get the following results:
>>>
>>> IDL: % Time elapsed: 6.4400001 seconds.
>>>
>>> MATLAB 2010b: Elapsed time is 0.594112 seconds.
>>>
>>> Again, MATLAB for loops run faster by a factor of ~10.
>>
>> Great idea,
>>
>>
>> I ran also these codes, with bigger numbers, 1000 instead of 500 for
>> each dimension and with Matlab 2015a I got 6.3 secondes to compare
>> with 102 secondes on my IDL 8.5. My desktop runs 4 cores and has 12
>> Go RAM and it runs faster with Matlab by a factor of 16 ! Damn.
>>
>
> For what it's worth, I got about a 5x speedup of IDL over Python. I used
> 1000 elements for each dimension:
>
> $ python mg_loopspeed.py
> 222.15 secs
>
> Code is:
>
> import numpy as np
> import time
>
> nx = 1000
> ny = 1000
> nz = 1000
>
> arr = np.zeros((nx, ny, nz))
>
> tic = time.clock()
>
> for z in xrange(nz):
> for y in xrange(ny):
> for x in xrange(nx):
> arr[x, y, z] = 1.0
>

> toc = time.clock()
> print '%0.2f secs' % (toc - tic)
>
> For comparison, my IDL time:
>
> $ idl -e ".run mg_loopspeed"
> IDL Version 8.5, Mac OS X (darwin x86_64 m64).
> (c) 2015, Exelis Visual Information Solutions, Inc., a subsidiary of
> Harris Corporation.
>
> % Compiled module: $MAIN$.
> % Compiled module: TIC.
> Elapsed Time: 47.486635
>
> Mike
> --
> Michael Galloy
> www.michaelgalloy.com
> Modern IDL: A Guide to IDL Programming (http://modernidl.idldev.com)

That's right, I got similar results on my Winpython 3.4. The reason seems to lie in the Just In Time compilation heavily used by Matlab (and Julia as well). To check it I used the "feature JIT off" undocumented Matlab command to see how it would fare.
And I got 48 secondes instead of the previous 6.3 secondes, an impressive change which puts Matlab un-JITed in the same league as IDL (102 secondes).
It is likely that's the reason.
To go further on this point with Python we'd need to check the efficiency of the Pypy Python interpreter and JIT compiler.

Seems to me that the work necessary to develop a JIT compiler for IDL is much heavier than compiling IDL against MKL.

Best

Best

Subject: Re: Speed does matter
Posted by on Thu, 03 Sep 2015 07:47:38 GMT
View Forum Message <> Reply to Message

Hi,

on the MKL subject, this Library just got free :

"No Cost Options for Intel Math Kernel Library (MKL), Support yourself, Royalty-Free

The Intel® Math Kernel Library (Intel® MKL), the high performance math library for x86 and x86-64, is available for free for everyone (click here now to register and download). Purchasing is only necessary if you want access to Intel® Premier Support (direct 1:1 private support from Intel), older versions of the library or access to other tools in Intel® Parallel Studio XE. Intel continues to actively develop and support this very powerful library - and everyone can benefit from that!"

Here is the link : https://software.intel.com/en-us/articles/free_mkl

No direct financial reason to skip it now !

best

Subject: Re: Speed does matter
Posted by on Mon, 17 Oct 2016 07:17:20 GMT
View Forum Message <> Reply to Message

Well, just saw a very ill-informed blog article from harrisgeospatial, where the anonymous author pits IDL against non-MKL Python.
While the recommended solution is to use Anaconda which is now MKL-compiled by default !
The article pretends that IDL is quicker than Python ...
Let us state the obvious : Among its category IDL is right now one the slowest "serious" software, I'd say roughly at least one order of magnitude.

Some have been pestering for years Exelis and Harris to simply recompile their software with Intel-MKL.
Right now I am enjoying a 40 speedup on SVD over IDL by using the IDL-Python bridge.

here is the culprit :

"The Amazing Race!

Wednesday, September 28, 2016
It wasn't so long ago that the IDL-Python bridge was introduced to IDL 8.5. It was with this new version, that I got my first experience programming with Python and testing the IDL-Python bridge. Through the past year it has been exciting to see the new changes and improvements that have become a part of the bridge.

Some of these new features include:

-Better error catching with the IDL-Python bridge

-Enhanced Jupyter notebook that allows for the development of full IDL programs and Python code in the same environment

-Improved support for variables passing back and forth

With all the time I have spent with Python, I have always wondered what some of the advantages are between Python and IDL. One thing that I have commonly heard several engineers say was that IDL was much faster than Python. For this blog, I decided to put that to the test and see how Python and IDL really compared to one another.

Before talking about the test, I do just want to explain things a bit about how it was set up and some potential caveats about the processing times that will be shown. With the tests I created, I did my best to choose tests that were comparable between IDL and Python. Since I'm no expert at Python, there very well may have been other methods that could be faster than what I will show. Most of the pieces I included in the test are things I found easily by doing a web search - meaning that most of the approaches I used were the most common programming methods that people are likely using. This shows how much faster IDL might be than a general program than something that someone might write in Python.

The test:

Here is what was actually tested between IDL and Python with an array of [10000,10000] or 10000*10000 elements

-Array creation time

-Type conversion times

-Index array creation times (i.e. [0,1,2,3,4...,n-1])

-Incrementing array values of all elements by 1

-Complex math expression with array (exact equation: sqrt(sin(arr*arr)))

-Single threaded for IDL and multithreaded

-Array element access times (i.e. setting y = arr[i])

-Simple image processing filter times (filters: sobel, roberts, prewitt)

The results:

Average array creation time (seconds):

Python : 0.213000 +/- 0.00953933

IDL : 0.0936666 +/- 0.0155028

Total time (seconds):

Python : 0.639000

IDL : 0.281000

Python/IDL time ratio: 2.27402

Average array data type conversion time (seconds):

Python : 0.171333 +/- 0.0155028

IDL : 0.0730000 +/- 0.00866031

Total time (seconds):

Python : 0.514000

IDL : 0.219000

Python/IDL time ratio: 2.34703

Average index array creation time (seconds):

Python : 0.229000 +/- 0.00866031

IDL : 0.124667 +/- 0.0160104

Total time (seconds):

Python : 0.687000

IDL : 0.374000

Python/IDL time ratio: 1.83690

Average increasing array value time (seconds):

Python : 0.0933333 +/- 0.000577446

IDL : 0.0313334 +/- 0.000577377

Total time (seconds):

Python : 0.280000

IDL : 0.0940001

Python/IDL time ratio: 2.97872

Average complex math statements (1 thread) time (seconds):

Python : 6.36967 +/- 0.0645319

IDL : 8.34667 +/- 0.0155028

Total time (seconds):

Python : 19.1090

IDL : 25.0400

Python/IDL time ratio: 0.763139

Average complex math statements (8 thread) time (seconds):

Python : 6.34400 +/- 0.0321871

IDL : 1.93933 +/- 0.00923762

Total time (seconds):

Python : 19.0320

IDL : 5.81800

Python/IDL time ratio: 3.27123

Average loop through array element time (seconds):

Python : 11.5290 +/- NaN

IDL : 3.29100 +/- NaN

Total time (seconds):

Python : 11.5290

IDL : 3.29100

Python/IDL time ratio: 3.50319

Average image processing routines time (seconds):

Python : 15.3660 +/- 0.0829635

IDL : 1.39900 +/- 0.0238955

Total time (seconds):

Python : 46.0980

IDL : 4.19700

Python/IDL time ratio: 10.9836

Conclusion:

In short, IDL significantly outperformed Python will all the speed tests apart from the complex math statement. However, IDL has access to built in multithreading for large arrays and, with multithreading enabled, IDL outperforms Python significantly when using all available cores.

Below is the IDL code used to compare the processing speed of IDL and Python. To use it you will need a few Python modules which can be found at the beginning of the procedure "python_test". "

Subject: Re: Speed does matter
Posted by wlandsman on Mon, 17 Oct 2016 18:34:21 GMT
View Forum Message <> Reply to Message

On Monday, October 17, 2016 at 3:17:23 AM UTC-4, Kallisthène wrote:

>
> Some have been pestering for years Exelis and Harris to simply recompile their software with Intel-MKL.
> Right now I am enjoying a 40 speedup on SVD over IDL by using the IDL-Python bridge.

Thanks for pushing on this. It does seem that Harris Geospatial needs to update IDL to keep it speed competitive with other interpreted programs.
As you note, MATLAB now has available both just-in-time compilation, and Intel MKL compilation.

https://www.mathworks.com/products/matlab/matlab-execution-e ngine/
https://software.intel.com/en-us/articles/using-intel-math-k
ernel-library-with-mathworks-matlab-on-intel-xeon-phi-coproc essor-system

And Python has "broadcasting" syntax
https://docs.scipy.org/doc/numpy/user/basics.broadcasting.ht ml

I should note though, that I was recently able to speed up a matrix-heavy IDL computation by a factor of 3 by judicious of the BLAS_AXPY (and REPLICATE_INPLACE) routine. --Wayne

---

Subject: Re: Speed does matter
Posted by Markus Schmassmann on Tue, 18 Oct 2016 10:35:34 GMT
View Forum Message <> Reply to Message

On 10/17/2016 08:34 PM, wlandsman wrote:
> And Python has "broadcasting" syntax
> https://docs.scipy.org/doc/numpy/user/basics.broadcasting.ht ml
so does MATLAB, since 2007a as bsxfun, as of 2016b also implicitly using normal operators, see

https://de.mathworks.com/help/matlab/ref/bsxfun.html