**[ndarray modules](https://0fps.net/2013/06/01/ndarray-modules/)**

Posted on [June 1, 2013](https://0fps.net/2013/06/01/ndarray-modules/)by [mikolalysenko](https://0fps.net/author/mikolalysenko/" \o "View all posts by mikolalysenko)

In the [last](https://0fps.wordpress.com/2013/05/22/implementing-multidimensional-arrays-in-javascript/) [two](https://0fps.wordpress.com/2013/05/28/cache-oblivious-array-operations/) posts I introduced [ndarrays](https://github.com/mikolalysenko/ndarray), and explained the rationale and implementation details of the library.  In this post I am going to show a few of the libraries that I have written using ndarrays.  All of this code works in both node.js and within any browser that [supports typed arrays](http://caniuse.com/typedarrays).  You can run the demos directly in [node.js](http://nodejs.org/) or else test them out in a browser using a bundler like [browserify](https://0fps.wordpress.com/2013/05/22/implementing-multidimensional-arrays-in-javascript/).  You can click on the links for each of them to find out more details about a specific module.

Vector arithmetic

[cwise](https://github.com/mikolalysenko/cwise): Array operation meta-programming library

[ndarray-ops](https://github.com/mikolalysenko/ndarray-ops): Common array operations

[ndarray-complex](https://github.com/mikolalysenko/ndarray-complex): Array operations for complex numbers

Image processing

[get-pixels](https://github.com/mikolalysenko/get-pixels): Reads the pixels from an image as an ndarray

[save-pixels](https://github.com/mikolalysenko/save-pixels): Writes a 2D ndarray to an image file

[lena](https://github.com/mikolalysenko/lena): The Lena test image as a require()able commonjs module.

[luminance](https://github.com/mikolalysenko/luminance): Converts an rgb image into luminance.

[normalize](https://github.com/mikolalysenko/ndarray-normalize): Scales an ndarray to mean 0 and standard deviation 1

[ndarray-warp](https://github.com/mikolalysenko/ndarray-warp): Apply a non-linear warp to an ndarray

Fourier analysis

[ndarray-fft](https://github.com/mikolalysenko/ndarray-fft): Fast fourier transform for ndarrays

[ndarray-convolve](https://github.com/mikolalysenko/ndarray-convolve): Convolutions and correlations for ndarrays

[phase-align](https://github.com/mikolalysenko/phase-align): Pattern matching and alignment

[ndarray-translate-fft](https://github.com/mikolalysenko/ndarray-translate-fft): Phase shifts an ndarray

Morphology and miscellaneous stuff

[ndarray-pack](https://github.com/mikolalysenko/ndarray-pack): Convert a numeric.js array to an ndarray

[ndarray-moments](https://github.com/mikolalysenko/ndarray-moments): Calculate first few terms of moment generating function

[distance-transform](https://github.com/mikolalysenko/distance-transform): Fast L^p distance transforms

[ball-morphology](https://github.com/mikolalysenko/ball-morphology): Mathematical morphology with ball-shaped structuring elements

Conclusions

This list is by no means exhaustive, and I have been writing more modules as I need them.  One of the nice things about working with CommonJS modules is that it is pretty straight forward to create your own module on [npm](https://npmjs.org/), and reuse its functionality.  I think that this style of programming could make building large scientific computing projects like SciPy/NumPy much more manageable.  Each function in such a project could be decomposed into a separate module, and it would be easy to experiment with different implementations.