

## Chapter 7

---

### Supported NumPy Functions

All NumPy and SciPy functions are supported for use in task parallel operations using `starp.ppeval`. This chapter lists the NumPy functions that are currently supported for use in data parallel operations using Star-P. Please refer to the support web page, <http://www.interactivesupercomputing.com/support>, for the most up-to-date function status.

### Overloaded Data Parallel NumPy Operators and Functions

---

Operators	Explanation
+	Matrix addition (element-wise)
-	Matrix subtraction (element-wise)
*	Multiplication (element-wise, not matrix multiplication)
/	Division (element-wise, not matrix division)
**	Exponentiation (element-wise, not matrix exponentiation)
<	Less than (element-wise)
>	Greater than (element-wise)
==	Equivalence (element-wise)
>=	Greater than or equal to (element-wise)
<=	Less than or equal to (element-wise)
!=	Nonequivalence (element-wise)

**Note:** All relational operators between NumPy and Star-P objects, do not necessarily produce the proper output due to the following bug in NumPy: <http://scipy.org/scipy/scipy/ticket/521>. The bug occurs when a distributed Star-P object is on the right hand side and a local NumPy object is on the left hand side of a relational operator.

Functions	Explanation
<code>starp.numpy.abs(A)</code>	Returns the absolute value of each element in the input array.
<code>starp.numpy.add(A,B)</code>	Matrix addition (element-wise), same as A+B
<code>starp.numpy.all(A)</code>	Returns true if all elements of A are true.
<code>starp.numpy.any(A)</code>	Returns true if any of the elements of A are true
<code>starp.numpy.arange(N, M=None, stride=None)</code>	Works like <code>numpy.arange()</code> . It returns a 1D server array of unit spaced values from N to M (but not including M). If M is not specified, then it returns unit spaced values from 0 to N (not including N). The optional stride parameter may be set to specify a spacing different from one.
<code>starp.numpy.arccos(A)</code>	Returns the inverse cosine of each element in the input array.
<code>starp.numpy.arccosh(A)</code>	Returns the inverse hyperbolic cosine of each element in the input array.
<code>starp.numpy.arcsin(A)</code>	Returns the inverse sine of each element in the input array.
<code>starp.numpy.arcsinh(A)</code>	Returns the inverse hyperbolic sine of each element in the input array.
<code>starp.numpy.arctan(A)</code>	Returns the inverse tangent of each element in the input array.
<code>starp.numpy.arctan2(A,B)</code>	Works like <code>numpy.arctan2()</code> . It returns the four quadrant arctan computed from the ratio A/B performed elementwise. The args A and B must be the same shape. The returned object is a server array.
<code>starp.numpy.arctanh(A)</code>	Returns the inverse hypbolic tangent of each element in the input array.
<code>starp.numpy.array(&lt;list&gt;, dtype=&lt;type&gt;, copy=&lt;int&gt;)</code>	Works like <code>numpy.array()</code> . It takes a list of numbers in square brackets and returns a starp array object (i.e. a server array).
<code>starp.numpy.ceil(A)</code>	Rounds the elements of the input array up (towards positive infinity) to the next integer.
<code>starp.numpy.concatenate(&lt;tuple&gt;, axis=&lt;int&gt;)</code>	Works like <code>numpy.concatenate()</code> . Pass it a list or tuple of Star-P arrays, and it will concatenate them on the axis specified (default axis=0).

<code>starp.numpy.conj(A)</code>	Returns the complex conjugate of the array (element-wise).
<code>starp.numpy.conjugate(A)</code>	Returns the complex conjugate of the array (element-wise)
<code>starp.numpy.copy(A, dist=None)</code>	Returns a copy of the matrix A, with a possibly different distribution. The argument to <code>dist</code> is the same as for the distribution property.
<code>starp.numpy.correlate(a, b, mode='&lt;string&gt;')</code>	<p>Works like <code>numpy.correlate()</code>. Given two server vectors <code>a</code> and <code>b</code>, it returns their correlation as a server vector. Only 1D objects (vectors) are supported.</p> <p>The keyword argument "mode" defines the length of the returned vector. Valid values for mode are:</p> <ul style="list-style-type: none"> <li>* full -- Returns the full correlation vector.</li> <li>* same -- Returns a vector of the same size as largest input. The vector is centered on the middle of the correlation range.</li> <li>* valid -- Return vector elements only when both input vectors fully overlap (default).</li> </ul>
<code>starp.numpy.cos(A)</code>	Returns the cosine of each element in the input array.
<code>starp.numpy.cosh(A)</code>	Returns the hyperbolic cosine of each element in the input array.
<code>starp.numpy.cumsum(A, axis=&lt;int&gt;)</code>	<p>Works like <code>numpy.cumsum()</code>. It returns the cumulative sum of the array A taken along the axis specified by "axis". The returned array lives on the server.</p> <p>Currently <code>cumsum</code> is only supported for 1D and 2D args.</p>

<code>starp.numpy.diag(v,k=0)</code>	<p>Works like <code>numpy.diag()</code>. Given an array <code>v</code> and a diagonal <code>k</code>, it returns a 2D server array with the vector <code>v</code> along the <math>k^{\text{th}}</math> diagonal, and zeros elsewhere.</p> <p>The array <code>v</code> must be a server array.</p> <p>The size of the returned array is determined by the size of <code>v</code>. The diagonal <code>k</code> is zero for the main diagonal, and is positive or negative for diagonals above or below the main diagonal.</p> <p>By default, <code>k=0</code>.</p>
<code>starp.numpy.diagonal(A, k)</code>	<p>Implements the same method as <code>numpy.diagonal()</code>.</p> <p>Given a 2D server matrix, it returns the elements of the matrix along the diagonal specified by offset. The elements are returned as a 1D server array. If offset is not specified, then the elements along the main diagonal are returned.</p>
<code>starp.numpy.divide(A,B)</code>	Same as <code>a/b</code> when <code>__future__.division</code> is not in effect.
<code>starp.numpy.dot(A, B)</code>	Works like <code>numpy.dot()</code> . It performs matrix multiplication on server arrays <code>A</code> and <code>B</code> , and returns the result as a server array. The arrays <code>A</code> and <code>B</code> must be conformable. Only 1D and 2D args are currently supported.
<code>starp.numpy.exp(A)</code>	Returns the exponential of each element in the input array.
<code>starp.numpy.expm1(A)</code>	Returns $(\exp(x)-1)$ for each element in the input array
<code>starp.numpy.eye(N, M=None, k=None)</code>	Works like <code>numpy.eye()</code> . It returns a server 2D array of size <code>NxM</code> (or <code>NxN</code> if <code>M</code> is not specified). The returned array has ones along the $k^{\text{th}}$ diagonal (or main diagonal if <code>k</code> is not specified), and is zero everywhere else.
<code>starp.numpy.fft.fft(A)</code>	Returns the discrete Fourier transform of a one dimensional array <code>A</code> . Currently the arbitrary <code>n</code> -point DFT (optional input argument <code>n</code> ) from NumPy is not supported.

<code>starp.numpy.fft.fft2(A)</code>	Returns the 2D FFT of matrix A. The shape parameter (optional input argument <code>s</code> ) and axis parameter (optional input argument <code>axis</code> ) from NumPy are currently not supported.
<code>starp.numpy.fft.fftshift(A, axes=None)</code>	Rearranges the vector or matrix <code>arg</code> to place the zero frequency elements to the center of the array. Use this function after performing an FFT.
<code>starp.numpy.fft.ifft(A)</code>	Returns the inverse discrete Fourier transform of a one dimensional array A. Currently the arbitrary n-point DFT (optional input argument <code>n</code> ) from NumPy is not supported.
<code>starp.numpy.fft.ifft2(A)</code>	Returns the inverse 2D FFT of matrix A. The shape parameter (optional input argument <code>s</code> ) and axis parameter (optional input argument <code>axis</code> ) from NumPy are currently not supported.
<code>starp.numpy.fix(A)</code>	Rounds the elements of the input array to the next integer towards zero.
<code>starp.numpy.floor(A)</code>	Implements the same method as <code>numpy.floor()</code> . It returns a server array whose elements are the elements of A rounded down (towards negative infinity) to the next integer.
<code>starp.numpy.histogram(A, bins=&lt;int&gt;, range=&lt;tuple&gt;, normed=&lt;bool&gt;)</code>	<p>Works like <code>numpy.histogram()</code>. It places the values of A into bins, and returns the count per bin as a server vector. If A is not 1D, it is flattened into a 1D vector before binning.</p> <p>The keyword args set the following behaviors:</p> <p><code>bins</code> -- This sets the number of bins to use. (Default=10.)</p> <p><code>range</code> -- Range must be a (float, float) tuple. It sets the min/max range of the binning. If <code>range=None</code>, then the min/max range is determined by the min/max of the input A. (Default=None.)</p> <p><code>normed</code> -- If <code>normed=True</code>, the histogram is normalized to give an integral of one. (Note that the integral is taken by assuming that the distribution is a density. That means that the sum of the values is not unity, but rather the sum of the elements times the step size.) (Default=False)</p>

<code>starp.numpy.hstack(&lt;tuple&gt;)</code>	Works like <code>numpy.hstack()</code> . It takes the arrays listed in the input tuple and concatenates them horizontally (along the first axis). The input arrays must have the same size along all axes except the first.
<code>starp.numpy.imag(A)</code>	Works like <code>numpy.imag(A)</code> . It returns a server array containing the imaginary part of the input arg A.
<code>starp.numpy.isfinite(A)</code>	Returns True for every array element which is finite.
<code>starp.numpy.isinf(A)</code>	Returns True for every array element which is infinite.
<code>starp.numpy.isnan(A)</code>	Returns True for every array element which is NaN.
<code>starp.numpy.kron(A, B)</code>	Works like <code>numpy.kron()</code> . Given two server arrays A and B, it returns a server array containing their Kronecker product. Only 2D arrays are currently supported..
<code>starp.numpy.linalg.cholesky(A)</code>	Works the same as <code>numpy.linalg.cholesky()</code> . It takes the 2D array A, and returns its Cholesky decomposition as a distributed 2D array. Only one matrix is returned, since the other matrix in the Cholesky decomposition is the transpose of the first one.  The input matrix must be positive definite for a result to be returned.
<code>starp.numpy.linalg.det(A)</code>	Implements the same method as <code>numpy.linalg.det()</code> . It takes a 2D array A, and returns its determinant as a scalar value.
<code>u,v = starp.numpy.linalg.eig(A)</code>	Works like <code>numpy.linalg.eig(A)</code> . It returns a tuple whose first element is a server vector holding the eigenvalues of A, and the second element is a server array holding the corresponding eigenvectors of A.  Currently, <code>starp.numpy.linalg.eig(A)</code> accepts only two dimensional arguments for A. A must either be a real, symmetric matrix, or a complex Hermitian matrix.

<code>starp.numpy.linalg.eigvals(A)</code>	<p>Works like <code>numpy.linalg.eigvals(A)</code>. It returns a server vector holding the eigenvalues of the matrix A.</p> <p>Accepts only two dimensional square matrices as arguments for A.</p>
<code>starp.numpy.linalg.inv(A)</code>	<p>Implements the same method as <code>numpy.linalg.inv()</code>. It takes the 2D array A, and returns its inverse as a 2D distributed array.</p>
<code>starp.numpy.linalg.norm(A, ord=None)</code>	<p>Implements the same method as <code>numpy.linalg.norm()</code>. This method returns the norm for the server matrix A as a scalar. The keyword <code>ord</code> may be one of the following:</p> <p><code>ord='fro'</code> -- Returns the Frobenius norm of A (default)</p> <p><code>ord=1</code> -- Takes the abs of each element, sums along axis 0, and returns the max of the sums.</p> <p><code>ord=-1</code> -- Takes the abs of each element, sums along axis 0, and returns the min of the sums.</p> <p><code>ord=2</code> -- Returns the max value from the SVD decomposition of A.</p> <p><code>ord=-2</code> -- Returns the min value from the SVD decomposition of A.</p> <p><code>ord=numpy.Inf</code> -- Takes the abs of each element, sums along axis 1, and returns the max of the sums.</p> <p><code>ord=-numpy.Inf</code> -- Takes the abs of each element, sums along axis 1, and returns the min of the sums.</p>
<code>starp.numpy.linalg.solve(A,b)</code>	<p>Implements the same method as <code>numpy.linalg.solve()</code>. It takes the 2D server array A and the 1D server array b, and returns the 1D array which solves the equation:</p> $Ax = b$

<code>starp.numpy.linalg.svd(A, compute_uv=&lt;int&gt;)</code>	Implements the same method as <code>numpy.linalg.svd()</code> . This method returns the singular value decomposition (SVD) of the server array A as a tuple of server arrays. The optional args have the following meanings:  <code>compute_uv</code> -- If = 1, compute and return the matrix decompositions along with the singular values (default). If = 0, only compute and return the singular values.
<code>starp.numpy.log(A)</code>	Returns the logarithm (base e) of each element in the array.
<code>starp.numpy.log10(A)</code>	Returns the logarithm (base 10) of each element in the array.
<code>starp.numpy.log1p(A)</code>	Returns $(\log(x+1))$ for each element x in the input array.
<code>starp.numpy.log2(A)</code>	Returns the logarithm (base 2) of each element in the input array.
<code>starp.numpy.max(A)</code>	Returns the maximum element in the array A.
<code>starp.numpy.mean(A, &lt;dim&gt;)</code>	Returns the average of the elements in array A along the specified dimension
<code>starp.numpy.min(A)</code>	Returns the minimum element in the array A.
<code>starp.numpy.multiply(A,B)</code>	Same as $A*B$ . Multiplies the elements of A and B element-wise.
<code>starp.numpy.nonzero(A)</code>	Implements the same method <code>numpy.nonzero()</code> . It returns the indices of the non-zero elements of A. The indices are returned as an N-tuple, where N is the dimension of A. The $m^{\text{th}}$ element of each tuple corresponds to the index of the $m^{\text{th}}$ non-zero entry.
<code>starp.numpy.ones([a, b])</code>	Works like <code>numpy.ones()</code> . It returns a server array whose elements are all one. The size of the array is specified by the input tuple.
<code>starp.numpy.power(A,B)</code>	Works like <code>numpy.power()</code> . It raises the elements of A to the power B, element-wise. A and B must have the same shape. The return result is a server object.
<code>starp.numpy.prod(A, &lt;dim&gt;)</code>	Returns the product of the matrix elements along a given dimension.

<code>starp.numpy.random.rand(&lt;int&gt;, &lt;int&gt;, ....)</code>	Works like <code>numpy.random.rand()</code> . It returns a server array of random numbers uniformly distributed between 0 and 1. The size of the array is specified by the input integers.
<code>starp.numpy.random.randn(&lt;int&gt;, &lt;int&gt;, ....)</code>	Works like <code>numpy.random.randn()</code> . It returns a server array of random numbers with Gaussian distribution having zero mean. The size of the array is specified by the input integers.
<code>starp.numpy.ravel(App, order=&lt;char&gt;)</code>	Implements the same method as <code>numpy.ravel()</code> . It takes the ND array A, and flattens it into a 1D array. The keyword argument, order, is a single character which specifies whether the array is handled assuming C ordering ('C' -- default) or Fortran ordering ('F').
<code>starp.numpy.real(A)</code>	Works like <code>numpy.real(A)</code> . It returns a server array containing the real part of the input arg A.
<code>starp.numpy.reshape(A, &lt;shape&gt;)</code>	Takes an ND array A, and reshapes it into the shape specified by the tuple <shape>.
<code>starp.numpy.round(A)</code>	Rounds elements of A down (towards negative infinity).
<code>starp.numpy.shape(A)</code>	Works like <code>numpy.shape(A)</code> . It returns the shape of the calling arg A as a (client side) tuple.
<code>starp.numpy.sign(A)</code>	Returns the sign of the elements in A. Positive values return 1. Negative values return -1, and Zero values return 0.
<code>starp.numpy.sin(A)</code>	Returns sine of A (element-wise).
<code>starp.numpy.sinh(A)</code>	Returns hyperbolic sine of A (element-wise).
<code>starp.numpy.size(A)</code>	Works like <code>numpy.size(A)</code> . It returns the number of elements in the calling arg A as an integer.

<code>starp.numpy.sort(A, axis=&lt;int&gt;)</code>	<p>Works like <code>numpy.sort()</code>. It returns the elements of the calling argument sorted into ascending order. For 2D objects, the sort occurs along the axis specified by the “axis” keyword arg. If “axis” is not specified, the sort takes place along the array’s last axis.</p> <p>Currently, only 1D and 2D objects may be sorted.</p> <p>Currently, only quicksort is supported.</p> <p>Currently, only sorts in ascending order are supported.</p>
<code>starp.numpy.sqrt(A)</code>	Returns the square root of A (element-wise).
<code>starp.numpy.subtract(A,B)</code>	Same as <code>A - B</code> . Subtracts the arguments A and B (element-wise)
<code>starp.numpy.sum(A)</code>	Returns a scalar equal to the sum of the elements of A.
<code>starp.numpy.tan(A)</code>	Returns the tangent of A (element-wise).
<code>starp.numpy.tanh(A)</code>	Returns the hyperbolic tangent of A (element-wise).
<code>starp.numpy.transpose(A, axes=&lt;tuple&gt;)</code>	Works like <code>numpy.transpose()</code> . It takes the array A and reorders it by permuting on the axes specified in the second arg axes. If <code>axes=None</code> (default), then <code>starp.numpy.transpose</code> returns the ordinary transpose of the array.
<code>starp.numpy.vstack(&lt;tuple&gt;)</code>	Works like <code>numpy.vstack()</code> . It takes the arrays listed in the input tuple and concatenates them vertically (along the 0 <sup>th</sup> axis). The input arrays must have the same size along all axes except the 0 <sup>th</sup> .
<code>starp.numpy.zeros([a, b])</code>	Works like <code>numpy.zeros()</code> . It returns a server array whose elements are all zero. The size of the array is specified by the input tuple.