

# API Documentation

API Documentation

February 27, 2015

## Contents

<b>Contents</b>	<b>1</b>
<b>1 Module CsTransform.pynufft</b>	<b>2</b>
1.1 Functions . . . . .	2
1.2 Variables . . . . .	3
1.3 Class pynufft . . . . .	3
1.3.1 Methods . . . . .	3

# 1 Module CsTransform.pynufft

package docstring author: Jyh-Miin Lin (Jimmy), Cambridge University address: jyhmiinlin at gmail.com  
Created on 2013/1/21

=====  
This file is part of pynufft.

pynufft is free software: you can redistribute it and/or modify it under the terms of the GNU General Public License as published by the Free Software Foundation, either version 3 of the License, or (at your option) any later version.

pynufft is distributed in the hope that it will be useful, but WITHOUT ANY WARRANTY; without even the implied warranty of MERCHANTABILITY or FITNESS FOR A PARTICULAR PURPOSE. See the GNU General Public License for more details.

You should have received a copy of the GNU General Public License along with pynufft. If not, see <<http://www.gnu.org/licenses/>>. =====

First, see test\_1D(), test\_2D(), test\_3D(), examples

## 1.1 Functions

<b>DFT_slow(<math>x</math>)</b>
---------------------------------

Compute the discrete Fourier Transform of the 1D array x <a href="https://jakevdp.github.io/blog/2013/08/28/understanding-the-fft/">https://jakevdp.github.io/blog/2013/08/28/understanding-the-fft/</a>
---

<b>DFT_point(<math>x, k</math>)</b>
-------------------------------------

Compute the discrete Fourier Transform of the 1D array x <a href="https://jakevdp.github.io/blog/2013/08/28/understanding-the-fft/">https://jakevdp.github.io/blog/2013/08/28/understanding-the-fft/</a>
---

<b>show_3D()</b>
------------------

<b>test_3D()</b>
------------------

<b>test_prolate()</b>
-----------------------

<b>test_2D()</b>
------------------

<b>rFOV_2D()</b>
------------------

<b>test_radial()</b>
----------------------

<b>test_1D()</b>
------------------

<b>test_2D_multiprocessing()</b>
----------------------------------

<code>test_wavelet()</code>
-----------------------------

<code>histeq(im, nbr_bins=256)</code>
---------------------------------------

Histogram equalization of a grayscale image.
--

<code>test_SR()</code>
------------------------

## 1.2 Variables

Name	Description
<code>cmap</code>	<b>Value:</b> <code>matplotlib.cm.gray</code>
<code>norm</code>	<b>Value:</b> <code>matplotlib.colors.Normalize(vmin= 0.0, vmax= 1.0)</code>
<code>__package__</code>	<b>Value:</b> <code>'CsTransform'</code>
<code>bpass</code>	<b>Value:</b> <code>array([ 6.85477291e-13, 6.85293298e-13, 6.84956443e-...</code>
<code>dirichlet</code>	<b>Value:</b> <code>&lt;scipy.interpolate.interpolate.interp1d object at 0x7f6b3...</code>
<code>t</code>	<b>Value:</b> <code>array([-49.9875, -49.975 , -49.9625, ..., 49.975 , 49.9...</code>

## 1.3 Class `pynufft`

`CsTransform.nufft.nufft` — `CsTransform.pynufft.pynufft`

### 1.3.1 Methods

<code>__init__(self, om, Nd, Kd, Jd, n_shift=None)</code>
---

constructor of <code>pyNufft</code>
-------------------------------------

Overrides: <code>CsTransform.nufft.nufft.__init__</code> extit(inherited documentation)
---

<code>initialize_gpu(self)</code>
-----------------------------------

Overrides: <code>CsTransform.nufft.nufft.initialize_gpu</code>
--

<code>gpu_k_deconv(self)</code>
---------------------------------

<code>gpu_k_modulate(self)</code>
-----------------------------------

<code>gpu_Nd2KdWKd2Nd(self, x, weight_flag)</code>
--

Now transform <code>Nd</code> grids to <code>Kd</code> grids(not be reshaped)
---

<b>gpu_forwardbackward</b> ( <i>self</i> , <i>x</i> )
---

<b>true_forward</b> ( <i>self</i> , <i>my_phantom</i> )
---

compute the exact NUFT without sparse approximation only for simulation
---

<b>forwardbackward</b> ( <i>self</i> , <i>x</i> )
---

<b>pseudoinverse2</b> ( <i>self</i> , <i>data</i> )
---

density compensation
----------------------

<b>pseudoinverse3</b> ( <i>self</i> , <i>data</i> , <i>mu</i> , <i>LMBD</i> , <i>gamma</i> , <i>nInner</i> , <i>nBreg</i> )
---

<b>pseudoinverse</b> ( <i>self</i> , <i>data</i> , <i>mu</i> , <i>LMBD</i> , <i>gamma</i> , <i>nInner</i> , <i>nBreg</i> )
--

<b>forwardbackward2</b> ( <i>self</i> , <i>x</i> )
--

Update the data-space
-----------------------

<b>maxrowsum</b> ( <i>self</i> )
----------------------------------

<b>backward2</b> ( <i>self</i> , <i>X</i> )
---

backward2(x): method of class pyNufft
---------------------------------------

from [M x Lprod] shaped input, compute its adjoint(conjugate) of Non-uniform Fourier transform
--

INPUT: X: ndarray, [M, Lprod] (Lprod=1 in case 1) where M =st['M']
--

OUTPUT: x: ndarray, [Nd[0], Nd[1], ... , Kd[dd-1], Lprod]
---

<b>adjoint</b> ( <i>self</i> , <i>f</i> )
---

adjoint operator to calcualte AT*y
------------------------------------

<b>adjoint2</b> ( <i>self</i> , <i>f</i> )
--

adjoint operator to calcualte AT*y
------------------------------------

### *Inherited from CsTransform.nufft.nufft*

Kd2Nd(), Nd2Kd(), backward(), emb\_fftn(), emb\_ifftn(), finalization(), forward(),  
gpufftn(), gpuifftn(), linear\_phase(), pipe\_density()

## Index

CsTransform (*package*)  
CsTransform.pynufft (*module*), 2–4  
CsTransform.pynufft.DFT\_point (*function*), 2  
CsTransform.pynufft.DFT\_slow (*function*),  
2  
CsTransform.pynufft.histeq (*function*),  
3  
CsTransform.pynufft.pynufft (*class*), 3–  
4  
CsTransform.pynufft.rFOV\_2D (*function*),  
2  
CsTransform.pynufft.show\_3D (*function*),  
2  
CsTransform.pynufft.test\_1D (*function*),  
2  
CsTransform.pynufft.test\_2D (*function*),  
2  
CsTransform.pynufft.test\_2D\_multiprocessing  
(*function*), 2  
CsTransform.pynufft.test\_3D (*function*),  
2  
CsTransform.pynufft.test\_prolate (*function*), 2  
CsTransform.pynufft.test\_radial (*function*),  
2  
CsTransform.pynufft.test\_SR (*function*),  
3  
CsTransform.pynufft.test\_wavelet (*function*), 2