

# btf

August 4, 2024

This file is part of CasADi.

CasADi -- A symbolic framework for dynamic optimization.  
Copyright (C) 2010–2023 Joel Andersson, Joris Gillis, Moritz Diehl,  
KU Leuven. All rights reserved.  
Copyright (C) 2011–2014 Greg Horn

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

CasADi 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  
Lesser General Public License for more details.

You should have received a copy of the GNU Lesser General Public  
License along with CasADi; if not, write to the Free Software  
Foundation, Inc., 51 Franklin Street, Fifth Floor, Boston, MA 02110-1301 USA

```
[1]: from casadi import *  
import numpy
```

Let's construct a block diagonal structure

```
[2]: b1 = DM([[2,3],[4,5]])  
b2 = DM([[6,7,8],[9,10,11],[12,13,14]])  
A = diagcat(1,b1,b2,15)
```

```
[3]: print("original: ")  
print(A)
```

original:

```
[[1, 00, 00, 00, 00, 00, 00],  
 [00, 2, 3, 00, 00, 00, 00],  
 [00, 4, 5, 00, 00, 00, 00],  
 [00, 00, 00, 6, 7, 8, 00],
```

```
[00, 00, 00, 9, 10, 11, 00],
[00, 00, 00, 12, 13, 14, 00],
[00, 00, 00, 00, 00, 00, 15]]
```

Ruin the nice structure

```
[4]: numpy.random.seed(0)
p1 = numpy.random.permutation(A.size1())
p2 = numpy.random.permutation(A.size2())
```

```
[5]: S = A[p1,:]
      #S = A[p1,p2]
```

```
[6]: print("randomly permuted: ")
      print(S)
      nb, rowperm, colperm, rowblock, colblock, coarse_rowblock, coarse_colblock = S.
      ↪sparsity().btf()
```

randomly permuted:

```
[[00, 00, 00, 00, 00, 00, 15],
 [00, 4, 5, 00, 00, 00, 00],
 [00, 2, 3, 00, 00, 00, 00],
 [00, 00, 00, 6, 7, 8, 00],
 [1, 00, 00, 00, 00, 00, 00],
 [00, 00, 00, 12, 13, 14, 00],
 [00, 00, 00, 9, 10, 11, 00]]
```

```
[7]: print("number of blocks: ", nb)
      print("rowperm: ", rowperm)
      print("colperm: ", colperm)
      print("restored:")
      print(S[rowperm,colperm])
      print("rowblock: ", rowblock)
      print("colblock: ", colblock)
      print("coarse_rowblock: ", coarse_rowblock)
      print("coarse_colblock: ", coarse_colblock)
```

```
number of blocks: 4
rowperm:  [0, 1, 2, 3, 5, 6, 4]
colperm:  [6, 1, 2, 3, 4, 5, 0]
restored:
```

```
[[15, 00, 00, 00, 00, 00, 00],
 [00, 4, 5, 00, 00, 00, 00],
 [00, 2, 3, 00, 00, 00, 00],
 [00, 00, 00, 6, 7, 8, 00],
 [00, 00, 00, 12, 13, 14, 00],
 [00, 00, 00, 9, 10, 11, 00],
```

```
[00, 00, 00, 00, 00, 00, 1]]  
rowblock:  [0, 1, 3, 6, 7]  
colblock:  [0, 1, 3, 6, 7]  
coarse_rowblock:  [0, 0, 0, 7, 7]  
coarse_colblock:  [0, 0, 7, 7, 7]
```