

Tensor Node Notation

Single Nodes

Creating a tensor node

With a tensor node N we denote a d -dimensional tensor which modes have been named for example $\alpha = \{\alpha_1, \dots, \alpha_d\}$. In short, we write $N = N(\alpha)$. If every mode name has domain $\{1, 2\}$, then all mode sizes of the tensor N correspondingly equal 2.

```
d = 3
```

```
d = 3
```

```
alpha = mna('alpha',1:d)
```

```
alpha = 1×3 cell array
    {'alpha_1'}    {'alpha_2'}    {'alpha_3'}
```

```
N = init_node(alpha,2)
```

```
N = struct with fields:
    mode_names: {'alpha_1' 'alpha_2' 'alpha_3'}
    pos: [1×1 struct]
    data: [2×2×2 double]
```

```
N.pos
```

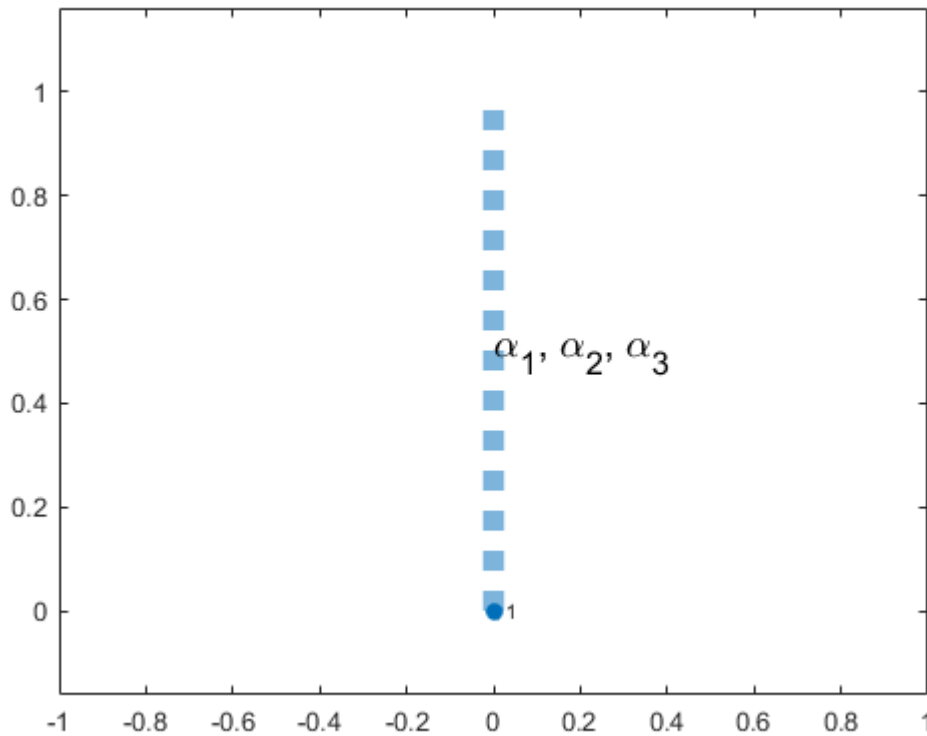
```
ans = struct with fields:
    alpha_1: 1
    alpha_2: 2
    alpha_3: 3
```

```
N.data
```

```
ans =
ans(:,:,1) =
    0    0
    0    0
```

```
ans(:,:,2) =
    0    0
    0    0
```

```
net_view(N) % there is not much to see so far
```



For now, we manually set the entries in `N.data`. However later, `N.data` may have been permuted.

```
nN = numel(N.data)
```

```
nN = 8
```

```
N.data(:) = 1:nN
```

```
N = struct with fields:
    mode_names: {'alpha_1' 'alpha_2' 'alpha_3'}
    pos: [1x1 struct]
    data: [2x2x2 double]
```

Entries and subtensors

In order to select a single entry $N(\alpha_1 = i_1, \dots, \alpha_d = i_d)$, we use the following command `node_part`. The mode names do not have to be given in a specific order. Also, `node_part` always returns a tensor node, not a number.

```
i = {1,2,1};
Ni = node_part(N, 'alpha_1', i{1}, ...
    'alpha_3', i{3}, ...
    'alpha_2', i{2})
```

```
Ni = struct with fields:
```

```

mode_names: {'alpha_1' 'alpha_2' 'alpha_3'}
pos: [1×1 struct]
data: 3

```

```
Ni.data
```

```
ans = 3
```

```
N.data(i{:})
```

```
ans = 3
```

Or faster:

```
node_part(N,alpha,i)
```

```

ans = struct with fields:
    mode_names: {'alpha_1' 'alpha_2' 'alpha_3'}
    pos: [1×1 struct]
    data: 3

```

To select a subtensor $N(\alpha_1 \in s_1, \dots, \alpha_d \in s_d)$, the same command is used.

```

s = {1:2,2,1};
Ns = node_part(N,'alpha_1',s{1},...
    'alpha_3',s{3},...
    'alpha_2',s{2})

```

```

Ns = struct with fields:
    mode_names: {'alpha_1' 'alpha_2' 'alpha_3'}
    pos: [1×1 struct]
    data: [2×1 double]

```

```
Ns.data
```

```

ans = 2×1
     3
     4

```

```
N.data(s{:})
```

```

ans = 2×1
     3
     4

```

If we do not want to restrict the domain of a mode, we can leave it unspecified.

```

s = {1:2,2,1};
Ns = node_part(N,'alpha_3',s{3},...
    'alpha_2',s{2})

```

```

Ns = struct with fields:
    mode_names: {'alpha_1' 'alpha_2' 'alpha_3'}

```

```
pos: [1×1 struct]
data: [2×1 double]
```

```
Ns.data
```

```
ans = 2×1
      3
      4
```

Or, more convenient, we may use the short notation $N(\alpha \in s)$:

```
Ns = node_part(N,alpha,s)
```

```
Ns = struct with fields:
    mode_names: {'alpha_1' 'alpha_2' 'alpha_3'}
        pos: [1×1 struct]
        data: [2×1 double]
```

```
Ns.data
```

```
ans = 2×1
      3
      4
```

```
Ns = node_part(N,alpha(2:3),s(2:3))
```

```
Ns = struct with fields:
    mode_names: {'alpha_1' 'alpha_2' 'alpha_3'}
        pos: [1×1 struct]
        data: [2×1 double]
```

```
Ns.data
```

```
ans = 2×1
      3
      4
```

Unfolding

In order to turn a tensor node in a conventional tensor (or matrix or vector), we can use the command `unfold`. For example, to obtain $N^{\{\alpha_1, \alpha_2\}, (\alpha_3)}$, call

```
A = unfold(N,alpha(1:2),'alpha_3')
```

```
A = 4×2
     1     5
     2     6
     3     7
     4     8
```

```
unfold(N,alpha(1:2))
```

```
ans = 4×2
```

1	5
2	6
3	7
4	8

Its transpose, $N^{(\alpha_3),(\{\alpha_1,\alpha_2\})}$, is the output of

```
unfold(N, 'alpha_3', alpha(1:2))
```

```
ans = 2x4
     1     2     3     4
     5     6     7     8
```

A'

```
ans = 2x4
     1     2     3     4
     5     6     7     8
```

but we can also obtain a three dimensional tensor $N^{(\alpha_1),(\alpha_2),(\alpha_3)}$

```
unfold(N, alpha{:})
```

```
ans =
ans(:, :, 1) =
     1     3
     2     4
```

```
ans(:, :, 2) =
     5     7
     6     8
```

while $N^{(\alpha)}$ is just a vector

```
unfold(N, alpha)
```

```
ans = 8x1
     1
     2
     3
     4
     5
     6
     7
     8
```

Size of a node

The command `node_size(N)` returns a `struct` which contains the sizes of the domains of all mode names of N . A selection of mode sizes can also be entered. See `help node_size` for more information.

```
n = node_size(N)
```

```
n = struct with fields:
    alpha_1: 2
    alpha_2: 2
    alpha_3: 2
```

```
[~,nvec,pos] = node_size(N,alpha([3,1]))
```

```
nvec = 1x2
      2     2
pos = 1x2
     3     1
```

Folding

The folding operation is to some extent the inverse to the unfolding operation. It is a simultaneous initialization of a tensor node and assignment of data.

```
N
```

```
N = struct with fields:
    mode_names: {'alpha_1' 'alpha_2' 'alpha_3'}
    pos: [1x1 struct]
    data: [2x2x2 double]
```

```
A = unfold(N,'alpha_3','alpha_2','alpha_1')
```

```
A =
A(:,:,1) =
     1     3
     5     7
```

```
A(:,:,2) =
     2     4
     6     8
```

```
fold(A,{'alpha_3','alpha_2','alpha_1'},[2,2,2])
```

```
ans = struct with fields:
    mode_names: {'alpha_3' 'alpha_2' 'alpha_1'}
    pos: [1x1 struct]
    data: [2x2x2 double]
```

```
A = unfold(N,'alpha_3','alpha_2','alpha_1')
```

```
A =
A(:,:,1) =
     1     3
     5     7
```

```
A(:, :, 2) =
```

```
    2    4  
    6    8
```

Mode sizes

We can also use the output of `node_size` to initialize nodes.

```
n
```

```
n = struct with fields:  
    alpha_1: 2  
    alpha_2: 2  
    alpha_3: 2
```

```
fold(1:8,alpha,n)
```

```
ans = struct with fields:  
    mode_names: {'alpha_1' 'alpha_2' 'alpha_3'}  
        pos: [1×1 struct]  
        data: [2×2×2 double]
```

```
n2 = assign_mode_size(alpha,[2,3,4])
```

```
n2 = struct with fields:  
    alpha_1: 2  
    alpha_2: 3  
    alpha_3: 4
```

```
fold(1:24,alpha,n2)
```

```
ans = struct with fields:  
    mode_names: {'alpha_1' 'alpha_2' 'alpha_3'}  
        pos: [1×1 struct]  
        data: [2×3×4 double]
```