Tutorial 1: First MATLAB warmup

Let's try out a few things:

Contents

Generate a random vector of length 4

type 'help rand' to see the help function of rand initialize random number generator

```
rand('seed',1);
v = rand(4,1);
% output it: leave away the semicolon at the end of the line
v
% or use
disp('output v:');
disp(v);
```

```
v =

    0.5129
    0.4605
    0.3504
    0.0950

output v:
    0.5129
    0.4605
    0.3504
    0.0950
```

Generate matrices

```
Mid = eye(2)    % 2x2 Identity matrix
Mdiag = diag([1 2 3]) % diagonal matrix with 1 2 3 on the diagonal
```

```
M = rand(3,3)  % a random 3x3 matrix
```

```
Mid =

     1     0
     0     1


Mdiag =

     1     0     0
     0     2     0
     0     0     3


M =

    0.4337    0.0781    0.1922
    0.7092    0.3693    0.4714
    0.1160    0.0336    0.1449
```

Accessing parts of a matrix

```
M(:,1)  % print first column
M(2,:)  % print second row
M(1:2,1:2) % plot the 2x2 submatrix
```

```
ans =

    0.4337
    0.7092
    0.1160


ans =

    0.7092    0.3693    0.4714


ans =

    0.4337    0.0781
    0.7092    0.3693
```

Useful matrix operations

```
Mconj = M';  % matrix conjugation
a = M + 2;   % add 2 to all elements
b = M * M;   % matrix-matrix multiplication
c = M .* M;  % element-wise multiplication
d = expm(M); % matrix exponential
```

Useful matrix routines

use 'help *functionname*' to see the syntax of these functions

```
M = rand(10,10);    % random 10x10 matrix
M = M + M';         % initialize hermitian matrix
[V,D] = eig(M);     % Eigenvalue decomposition
% Iterative eigensolver: compute the 2 eigenvector with smallest
% eigenvalues
[V,D] = eigs(M,2,'SA');
% Singular value decomposition (economy size)
[U,s,V] = svd(M,'econ');
% qr decomposition
[Q,R] = qr(M,0);
```

Generating tensors

```
T1 = rand(2,3,3);    % 2x3x3 random tensor
T2 = zeros(2,4,1);   % 2x4x1 tensor with zeros
```

Get size of a tensor

```
size(T1)
disp(['Total number of elements:',num2str(numel(T1))]);
```

```
ans =

     2     3     3

Total number of elements:18
```

reshaping

```
v = 1:9    % this is a vector with elements from 1...9
M = reshape(v,3,3) % reshape it into a 3x3 matrix
v2 = 1:8   % this is a vector with elements from 1...8
T = reshape(v2,2,2,2) % reshape it into a 2x2x2 tensor
```

```
v =

     1     2     3     4     5     6     7     8     9


M =

     1     4     7
     2     5     8
     3     6     9


v2 =
```

```
     1     2     3     4     5     6     7     8


T(:,:,1) =

     1     3
     2     4


T(:,:,2) =

     5     7
     6     8
```

permutation

```
Mp = permute(M,[2 1]) % permute indices of matrix (=transpose)
Tp = permute(T,[1 3 2]) % permute indices of tensor
```

```
Mp =

     1     2     3
     4     5     6
     7     8     9


Tp(:,:,1) =

     1     5
     2     6


Tp(:,:,2) =

     3     7
     4     8
```

Tensor-tensor contraction using matrix-matrix multiplication

NOTE: we will use special functions tcontract and ncon which will do this automatically (see tutorial 5)!

```
A = rand(2,3,4);
B = rand(3,4,2);
% contract legs 2 and 3 of A with legs 1 and 2 of B
% determine sizes of A and B
sA = size(A);
lAdimleft = sA(1);
lAdimright = sA(2:3);
sB = size(B);
lBdimleft = sB(1:2);
lBdimright = sB(3);
% now reshape A and B into matrices
```

```
MA = reshape(A,lAdimleft,prod(lAdimright));
MB = reshape(B,prod(lBdimleft),lBdimright);
% now multiply matrices
MC = MA*MB;
% and now reshape back intro tensor
disp('Resulting 2x2 tensor:');
C = reshape(MC,[lAdimleft,lBdimright])
```

```
Resulting 2x2 tensor:

C =

    2.8774    2.0582
    2.9092    3.2396
```

Tensor-tensor contraction requiring a permutation first

Same example as before but now the legs are not in correct order

```
A = rand(3,2,4);
B = rand(2,4,3);
% contract legs 1 and 3 of A with legs 3 and 2 of B
% first permute the legs:
% the legs of A which are going to be contracted need to be on the right
% side:
A = permute(A,[2,1,3]); % now is a 2x3x4 tensor
B = permute(B,[3,2,1]); % now is a 3x4x2 tensor
% and now use same code as above:
% determine sizes of A and B
sA = size(A);
lAdimleft = sA(1);
lAdimright = sA(2:3);
sB = size(B);
lBdimleft = sB(1:2);
lBdimright = sB(3);
% now reshape A and B into matrices
MA = reshape(A,lAdimleft,prod(lAdimright));
MB = reshape(B,prod(lBdimleft),lBdimright);
% now multiply matrices
MC = MA*MB;
% and now reshape back intro tensor
disp('Resulting 2x2 tensor:');
C = reshape(MC,[lAdimleft,lBdimright])
```

```
Resulting 2x2 tensor:

C =

    3.6716    2.7358
    3.5599    2.5533
```

Matlab cell arrays

Cell arrays are useful containers in Matlab, which can be filled with arbitrary types (also mixing types in the same cell), see e.g.:

```
c1 = cell(1,2); % initialize a 1x2 cell with empty elements
c1{1} = rand(2,2);  % add a random 2x2 matrix to c1
c1{2} = rand(3,3);  % add a random 3x3 matrix to c2
c1{end+1} = 2;      % add a number to the cell array
c1      % output content of cell array
c1{1}   % output the first element
c1{2}   % output the second element
```

```
c1 =

  1×3 cell array

    [2×2 double]    [3×3 double]    [2]


ans =

    0.5913    0.7222
    0.8394    0.2539


ans =

    0.8545    0.9986    0.6232
    0.1628    0.1064    0.6792
    0.6090    0.1090    0.1376
```

Published with MATLAB® R2017a

Tutorial 2: Note on functions in Matlab

Functions are usually definied in separate .m files It is not possible to combine function definitions within scripts. It is possible to add several functions within the same file as below, but then only the top function is accessible from external scripts

Contents

Define a function

The name of the m-file needs to be the same as the function, and then you can call it from other scripts or from the command line. E.g. type 'tutorial(1,3)' on the command line to call this function You can also define functions which do not return anything

```
function res = tutorial2(a, b)
    % check number of input arguments with keyword nargin
    if nargin == 0
        % no input: specify default values:
        a = 1;
        b = 1;
    end
    % Call another function defined below (not accessible from outside)
    res = myfunc(a) + myfunc(b);
end

function y = myfunc(x)
    % Example function which can be called from tutorial2 function
    y = x^3 + x;
end
```

```
ans =

     4
```

Published with MATLAB® R2017a

Tutorial 3: Exact diagonalization

Let's do a simple exact diagonalization of a Heisenberg model with open boundary conditions:

Contents

$$H = J\sum_{\langle i,j\rangle} S_i S_j = H = J\sum_{\langle i,j\rangle} S_i^z S_j^z + \frac{1}{2}(S_i^+ S_j^- + S_i^- S_j^+)$$

Constructing the Hamiltonian in basis {up,down}

```
Sz = diag([1/2,-1/2]);
Sp = [0 1;0 0];
Sm = Sp';
Id = eye(2);

% Two site Hamiltonian:
H2 = kron(Sz,Sz) + 0.5*(kron(Sm,Sp) + kron(Sp,Sm))
```

```
H2 =

    0.2500         0         0         0
         0   -0.2500    0.5000         0
         0    0.5000   -0.2500         0
         0         0         0    0.2500
```

Diagonalize and get lowest energy state

```
[V,D] = eigs(H2,1,'SA');
disp(['2-site ground state energy:',num2str(D)]);
disp('Eigenvector (singlet):')
disp(V);
```

```
2-site ground state energy:-0.75
Eigenvector (singlet):
   -0.0000
   -0.7071
    0.7071
   -0.0000
```

And now construct L-site Hamiltonian recursively

```
L = 10; %example
% Use sparse representation for Hamiltonian:
```

```
H2 = sparse(H2);
HL = H2;
Id2 = speye(2); %sparse identity
for k=3:L
    % construct identity operator on k-2 sites
    NId = speye(2^(k-2));
    HL = kron(HL,Id2)+kron(NId,H2);
end

% Diagonalize and get lowest energy state
[V,D] = eigs(HL,1,'SA');
disp(['10-site ground state energy:',num2str(D)]);
```

```
10-site ground state energy:-4.258
```

Published with MATLAB® R2017a

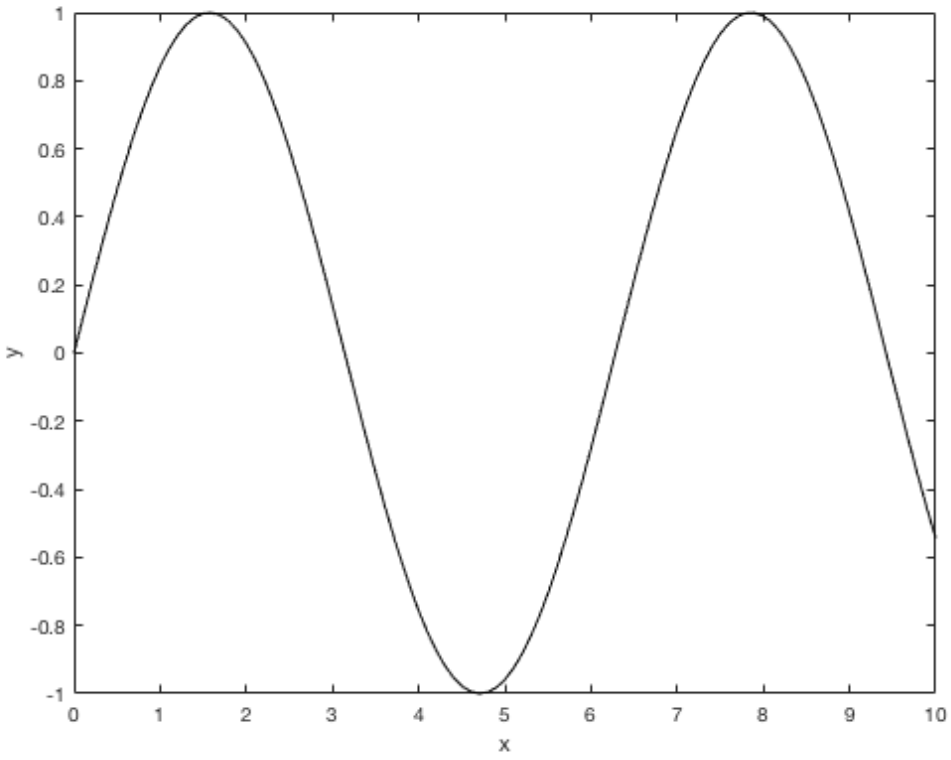
Tutorial 4: Basic plotting with Matlab

Here is an example of some basic plotting tools. You can checkout more in the documentation, e.g. type 'help plot' or online

Contents

Basic plot

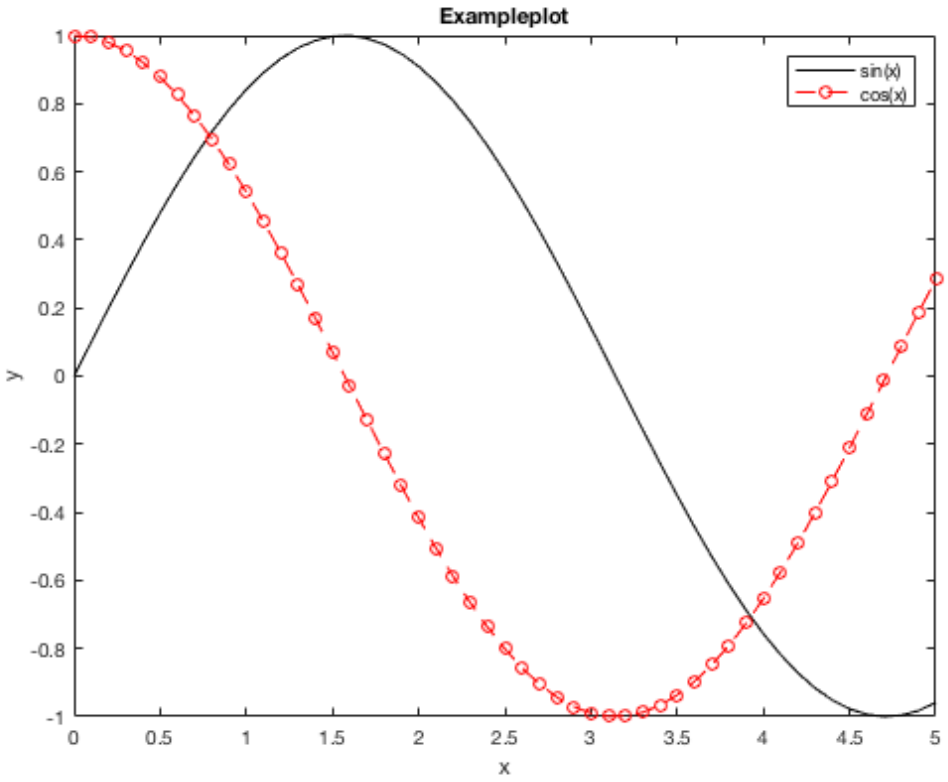
```
x=0:0.1:10; % vector of x values between 0 and 10 with spacing 0.1
figure;
set(gca,'FontSize',15); % make the default font a bit bigger
plot(x,sin(x),'k-'); % plot sin(x)
% put labels on axis:
xlabel('x');
ylabel('y');
```



add another curve to above plot

```
hold on; % use this otherwise previous curve gets overwritten
plot(x,cos(x),'ro--');
% to specify x and y limits use:
xlim([0 5]);  % x-limits
ylim([-1 1]); % y-limits
```

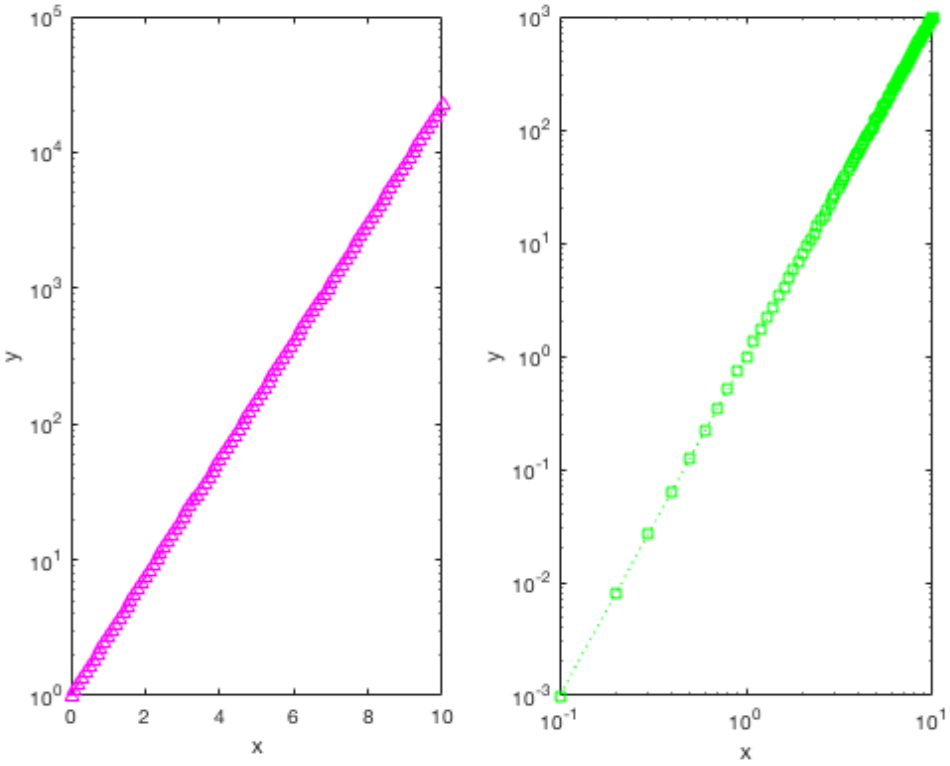
```
% add title and legend
title('Exampleplot');
legend('sin(x)','cos(x)');
```



Using subplots

```
figure;
subplot(1,2,1);  % first subplot
set(gca,'FontSize',15);
semilogy(x,exp(x),'m^-.'); % semilogarithmic plot
xlabel('x');
ylabel('y');

subplot(1,2,2);  % first subplot
set(gca,'FontSize',15);
loglog(x,x.^3,'gs:'); % loglog plot
xlabel('x');
ylabel('y');
```



Published with MATLAB® R2017a

Tutorial 5: Additional tensor network and other tools

Check out these useful routines, especially ncon and tensorsvd simplify tensor network codes a lot!

Contents

addpath

First you need to add the path to the additional functions tested here:

```
addpath('../minclude');
```

disp0

this function just simplifies output of text/numbers: instead of writing

```
disp(strcat('The result is:',num2str(4)));
% simply write
disp0('The result is:',4);
```

```
The result is:4
The result is:4
```

lreshape

with lreshape you can reshape and permute some legs and create a matrix directly. Checkout the documentation and/or code by 'help lreshape' Let's take the following example:

```
rand('seed',1);
A = rand(3,2,5,4);
B = rand(2,5,3);
% contract leg 3 of A with leg 2 of B
% This becomes:
[MA,lAdimleft,lAdimright] = lreshape(A,[1 2 4],[3]);
[MB,lBdimleft,lBdimright] = lreshape(B,[2],[1 3]);
MC = MA*MB;
C = reshape(MC,[lAdimleft,lBdimright]);
```

```
disp0('Resulting size of C: ',size(C));
```

```
Resulting size of C: 3  2  4  2  3
```

tcontract

tcontract allows you to contract two tensors (same as above) in one command

```
C = tcontract(A,B,3,2);
disp0('Resulting size of C: ',size(C));
% another example:
% contract A and B on indices [1,2] of A and [3,1] of B
D = tcontract(A,B,[1 2],[3 1]);
disp0('Resulting size of D: ',size(D));
```

```
Resulting size of C: 3  2  4  2  3
Resulting size of D: 5  4  5
```

ncon

ncon is a great tool to contract entire tensor networks, written by Robert N. C. Pfeifer, Glen Evenbly, Sukhwinder Singh, Guifre Vidal see documentation https://arxiv.org/abs/1402.0939 With this function above example is done in one line:

```
A = rand(3,2,5,4);
B = rand(2,5,3);
C = ncon({A,B},{[-1 -2 1 -3],[-4 1 -5]});
disp0('Resulting size of C: ',size(C));
```

```
Resulting size of C: 3  2  4  2  3
```

Example: Doing an SVD of a tensor with lreshape

```
T = rand(1,2,3,2);
% lets to an svd of this tensor with legs 1 and 3 on the left
% and legs 2 and 4 on the right:
[MT,dl,dr] = lreshape(T,[1 3],[2 4]);
[MU,s,MV] = svd(MT,'econ');
disp('Singular values:');
disp(diag(s)')
% now reshape U and V back to tensors:
U = reshape(MU,[dl,size(s,1)]);
V = reshape(MV,[dr,size(s,2)]);
disp0('Size of U: ',size(U));
disp0('Size of S: ',size(s));
disp0('Size of V: ',size(V));
```

```
Singular values:
    2.0736    0.5550    0.1331
```

```
Size of U: 1  3  3
Size of S: 3  3
Size of V: 2  2  3
```

tensorsvd

this function does above in one line:

```
[U,s,V] = tensorsvd(T,[1 3],[2 4]);
disp0('Size of U: ',size(U));
disp0('Size of S: ',size(s));
disp0('Size of V: ',size(V));
% WATCH OUT: in the usual svd routine the V contains the complex conjugates
% whereas tensorsvd does NOT return the complex conjugates
% In other words: By contracting U.s.V one recovers the matrix T
```

```
Size of U: 1  3  3
Size of S: 3  3
Size of V: 2  2  3
```

tensorsvd with compression

One can specify the number of singular values to keep:

```
nsingularvalues=3;
[U,s,V] = tensorsvd(T,[1 3],[2 4],nsingularvalues);
disp0('Size of U: ',size(U));
disp0('Size of S: ',size(s));
disp0('Size of V: ',size(V));
```

```
Size of U: 1  3  3
Size of S: 3  3
Size of V: 2  2  3
```

Doing an SVD of a tensor and multipy resulting tensors back together

Check out the following example and write down the diagrams by hand

```
T = rand(1,2,3,2);
[U,s,V] = tensorsvd(T,[1 3],[2 4]);
T2 = ncon({U,s,V},{[-1 -3 1],[1 2],[-2 -4 2]});
% compute the sum of the abs value of all elements:
sumabsdiff = sum(abs(T(1:end)-T2(1:end)));
disp0('Difference between T and T2:',sumabsdiff);
```

```
Difference between T and T2:5.3221e-15
```

Absorbing singular values in tensorsvd

With tensorsvd you can optionally absorb the singular values in U (with 'l') or in V ('r') or sqrt(s) on both sides ('m'):

```
T = rand(1,2,3,2);
[Us,s,V] = tensorsvd(T,[1 3],[2 4],inf,'l'); %absorb left
%singular values are contained in Us
T2 = ncon({Us,V},{[-1 -3 1],[-2 -4 1]});
% compute the sum of the abs value of all elements:
sumabsdiff = sum(abs(T(1:end)-T2(1:end)));
disp0('Difference between T and T2:',sumabsdiff);
```

```
Difference between T and T2:3.5527e-15
```

Published with MATLAB® R2017a