Companion software for "Volker Ziemann, *Hands-on Accelerator physics using MATLAB, CRCPress, 2019*" (https://www.crcpress.com/9781138589940)

Beam optics support functions 2D (Section 3.3)

Volker Ziemann, 211119

In this live script we define the functions for the 2D beam optics calculations, such as `calcmat()` that a frequently used in other calculations. All described functions reside in the subdirectory 2D that is contained in the archive `BeamOpticsSupportFile.zip`. Any scripts using these function need to include that subirectory with the command "`addpath ./2D`".

The function `calcmat()` to calculate all transfer matrices

The following function receives the `beamline` description as input and returns

- Racc(2,2,nmat): transfer matrices from the start to the each of each segment, such that R(:,:,end) is the transfer matrix from the start to the end of the beamline.
- spos: position along the beamline after each segment, useful when plotting.
- nmat: number of segments
- nlines: number of lines in the beamline

```
function [Racc,spos,nmat,nlines]=calcmat(beamline)
ndim=size(DD(1),1);
nlines=size(beamline,1);      % number of lines in beamline
nmat=sum(beamline(:,2))+1;    % sum over repeat-count in column 2
Racc=zeros(ndim,ndim,nmat);   % matrices from start to element-end
Racc(:,:,1)=eye(ndim);        % initialize first with unit matrix
spos=zeros(nmat,1);           % longitudinal position
ic=1;                         % element counter
for line=1:nlines             % loop over input elements
  for seg=1:beamline(line,2)  % loop over repeat-count
     ic=ic+1;                 % next element
     Rcurr=eye(2);            % matrix in next element
     switch beamline(line,1)
       case 1   % drift
         Rcurr=DD(beamline(line,3));
       case 2    % thin quadrupole
         Rcurr=Q(beamline(line,4));
       case 5   % thick quadrupole
         Rcurr=QQ(beamline(line,3),beamline(line,4));
       otherwise
         disp('unsupported code')
     end
     Racc(:,:,ic)=Rcurr*Racc(:,:,ic-1);    % concatenate
     spos(ic)=spos(ic-1)+beamline(line,3); % position of element
  end
end
end
```

Transfer matrix for a drift space `DD(L)`

The function `DD()` receives the length `L` of a drift space and resturns the 2x2 transfer matrix `out` for a drift space.

```
function out=DD(L)
  out=[1,L;0,1];
end
```

Transfer matrix for a thin-lens quadrupole `Q(F)`

The function `Q()` receives the focal length `F` as input and returns the 2x2 transfer matrix `out` for a thin-lens quadrupole.

```
function out=Q(F)
out=eye(2);
if abs(F)<1e-8, return; end   % turn off, if F=0
out=[1,0;-1/F,1];             % transfer matrix
end
```

Transfer matrix for a thick quadrupole `Q(F)`

The function `QQ()` receives the length `L` and `k1` as input and returns the 2x2 transfer matrix `out` for a thick quadrupole.

```
function out=QQ(L,k1)
ksq=sqrt(abs(k1));
if abs(k1) < 1e-6
    out=[1,L;0,1];
elseif k1>0
    out=[cos(ksq*L),sin(ksq*L)/ksq;-ksq*sin(ksq*L),cos(ksq*L)];
else
    out=[cosh(ksq*L),sinh(ksq*L)/ksq;ksq*sinh(ksq*L),cosh(ksq*L)];
end
end
```

R2beta()

The function `R2beta()` receives a transfer matrix `R` as input and returns the "tune" $Q = \mu/2\pi$ for the transfer matrix R, as well as the periodic Twiss parameters α, β, and γ following Equation 3.60.

```
function [Q,alpha,beta,gamma]=R2beta(R)
mu=acos(0.5*(R(1,1)+R(2,2)));
if (R(1,2)<0), mu=2*pi-mu; end
Q=mu/(2*pi);
beta=R(1,2)/sin(mu);
alpha=(0.5*(R(1,1)-R(2,2)))/sin(mu);
gamma=(1+alpha^2)/beta;
end
```

plot_betas()

The function `plot_betas()` receives the `beamline` description and the initial 2x2 beam matrix `sigma0` as input an produces a plot of the beta function. This function assumes that the emittance of sigma0 is 1, or $\det \sigma_0 = 1$, such that $\sigma_{11} = \beta$ is the beta function. It then uses Equation 3.43 to propagate σ.

```
function plot_betas(beamline,sigma0)
[Racc,spos]=calcmat(beamline);
betax=zeros(1,length(spos));
for k=1:length(spos)
   sigma=Racc(:,:,k)*sigma0*Racc(:,:,k)';
   betax(k)=sigma(1,1);
end
plot(spos,betax,'k');
xlabel(' s[m]'); ylabel('\beta_x [m]')
axis([0, max(spos), 0, 1.05*max(betax)])
end
```