Solutions for beam optics exercises using Octave or MATLAB

V. Ziemann, FREIA, Uppsala University

Abstract
The solutions for the exercises from the *Beam Optics Primer using Octave or MATLAB*.

A file `optics-primer-solutions.zip` with all software is available from the CAS web site[1].

Exercise 1

Show that multiplying two such matrices, one with L_1 and the other with L_2 in the upper right corner, produces a matrix with the sum of the distances in the upper right corner.

$$\begin{pmatrix} 1 & L_1 \\ 0 & 1 \end{pmatrix} \begin{pmatrix} 1 & L_2 \\ 0 & 1 \end{pmatrix} = \begin{pmatrix} 1 & L_1 + L_2 \\ 0 & 1 \end{pmatrix} \tag{1}$$

Exercise 2 and 3

How do you describe a ray that is parallel to the optical axis?
How do you describe a ray that is on the optical axis?

Parallel beam has the form $\begin{pmatrix} x \\ 0 \end{pmatrix}$ and a beam on the optical axis has the form $\begin{pmatrix} 0 \\ 0 \end{pmatrix}$.

Exercise 4

Show by multiplying the respective matrices that a parallel ray, which first passes through a lens with focal length f and then moves on a straight line, actually crosses the optical axis at a distance $L = f$ downstream of the lens. Hint: think a little extra about ordering of the matrices!

We start with a parallel ray $\begin{pmatrix} x \\ 0 \end{pmatrix}$ and first pass it through a focusing lens $\begin{pmatrix} 1 & 0 \\ -1/f & 1 \end{pmatrix}$ an then through a drift space $\begin{pmatrix} 1 & L \\ 0 & 1 \end{pmatrix}$. Using the rules that govern matrix multiplication, the progression goes from right to left. We get

$$\begin{pmatrix} 1 & L \\ 0 & 1 \end{pmatrix} \begin{pmatrix} 1 & 0 \\ -1/f & 1 \end{pmatrix} \begin{pmatrix} x \\ 0 \end{pmatrix} = \begin{pmatrix} 1 - L/f & L \\ -1/f & 1 \end{pmatrix} \begin{pmatrix} x \\ 0 \end{pmatrix} = \begin{pmatrix} (1 - L/f)x \\ -x/f \end{pmatrix} . \tag{2}$$

For this ray to pass through the optical axis, we must have $(1 - L/f)x = 0$ or $L = f$.

[1] https://cas.web.cern.ch/previous-schools

Exercise 5

Set `F=3` *and numerically verify what you found in Exercise 4, namely that parallel rays cross the axis after a distance* $L = f$.

We first define anonymous functions `D()` and `Q()`, which is a handy way to avoid writing similar matrices over and over. Then we specify the initial ray `x0` as well as other parameters and multiply the matrices—note the ordering—with in ray and place the result in the variable `xout`, which is a two-component array that has the position as its first entry.

```
D=@(L)[1, L; 0, 1];    % define function for drift
Q=@(F)[1, 0; -1/F, 1]; % define function for lens
x0=[1;0];              % initial ray, parallel
F=3;                   % focal length
L=F;                   % drift space
xout=D(L)*Q(F)*x0      % xout(1) should be zero
```

Exercise 6

Recall that the imaging equation for a lens is $1/b + 1/g = 1/f$, *which corresponds to a system of one focusing lens with focal length* f, *sandwiched between drift spaces with length* g *and* b, *respectively. Write a beam-line description that corresponds to this system. We will later return to it and analyze it.*

The variables `b`,`f`, and `g` must be defined before the following description of the beam line, where a ray will first pass through the element defined in the first line, which is a drift space of length `g`. The next element is a lens with focal length `f` followed by another drift space of length `b`.

```
optics=[ 1, 1, g, 0;
         2, 1, 0, f;
         1, 1, b, 0];
```

Exercise 7 and 8

Prepare initial coordinates that describe a particle that is on the optical axis, but has an initial angle $x' = 1$ *mrad and plot the position* x *along the beam line. Plot the angle* x' *along the beam line.*

After defining the focal length `F` and the beam line, we use `calcmat()` to calculate the matrices `Racc()`. The ray of the particle that is on the optical axis, but has an angle of 1 mrad is defined as `x0` and subsequently propagated in the loop over `k` to all locations in the beam line, where its position—the first entry of `x`—is stored in the array `data()`.

```
F=2.5;   % focal length of the quadrupoles
fodo=[ 1,  5,  0.2,  0;    % 5* D(L/10)
       2,  1,  0.0, -F;    % QD
       1, 10,  0.2,  0;    % 10* D(L/10)
       2,  1,  0.0,  F;    % QF
       1,  5,  0.2,  0];   % 5* D(L/10)
beamline=fodo;             % name must be 'beamline'
[Racc,spos,nmat,nlines]=calcmat(beamline);
x0=[0; 0.001];        % exercise 7: 1 mrad angle at start
data=zeros(1,nmat);   % allocate memory
for k=1:nmat
  x=Racc(:,:,k)*x0;
```

```
  data(k)=x(1);       % store the position
end
plot(spos,1e3*data,'k','LineWidth',2);
xlabel('s [m]'); ylabel(' x [mm]'); xlim([spos(1),spos(end)])
```

After the loop we plot the positions, conveniently scaled to mm, and annotate the axes accordingly.

Exercise 9

Plot both the position x and the angle x' through five cells.

Just replace the line where the `beamline` is defined by the following

```
beamline=repmat(fodo,5,1);
```

The function `repmat()` simply copies the preciously defined beam line `fodo` five times after one another. The rest of the code from exercise 8 remains unchanged.

Exercise 10

Plot the position x through 100 cells, play with different values of the focal length `F` *and explore whether you can make the oscillations grow.*

Just make the beam line longer by calling `repmat` with a larger argument

```
beamline=repmat(fodo,100,1);
```

The beam line becomes unstable is `F`, defined at the top of the file becomes 1 or smaller. Try out `F=1.001`, `F=1.0001`, then `F=0.9999` and observe.

Exercise 11

Use the beam line for the imaging system you prepared in Exercise 6 and launch a particle with $x_0 = 0$ and an angle of $x'_0 = 1$ mrad at one end. Verify that this particle crosses the center of the beam pipe at the exit of the beam line, provided that $b, g,$ and f satisfy the imaging equation that is shown in Exercise 6.

After defining the drift space between the object and the lens, as well as the focal length of the lens, we apply the imaging equation to determine where the image should appear after the lens. These parameters are then used in the beam-line description from Exercise 6 before we define the initial ray `x0`, which sits on the optical axis, but has an initial angle of 1 mrad, and propagate it through our short beam line.

```
g=2; f=0.5;           % define drift space and focal length
b=1/(1/f-1/g);        % imaging equation, location of the image
optics=[ 1, 1, g, 0;  % from Exercise 6
         2, 1, 0, f;
         1, 1, b, 0];
beamline=optics;
[Racc,spos,nmat,nlines]=calcmat(beamline);
x0=[0; 0.001];        % <----------initial angle
data=zeros(1,nmat);   % allocate memory
for k=1:nmat
  x=Racc(:,:,k)*x0;
  data(k)=x(1);       % <------------ position
end
plot(spos,1e3*data,'k','LineWidth',2);
xlabel('s [m]'); ylabel(' x [mm]'); xlim([spos(1),spos(end)])
```

Finally, we plot the position of the ray, again converted to mm, and annotate the axes. We observe that the ray actually crosses the optical axis at the location where the image is located.

Exercise 12

Calculate the angular divergence of the `beam`.

Whereas the `beam_size` is given by the standard deviation calculated from the positions of all 10000 particles, is the `angular_divergence` the standard deviation of the angles, stored as the second entry in `beam()`.

```
angular_divergence=std(beam(2,:))
```

Note that the built-in function `std()` conveniently calculates the standard deviation for us.

Exercise 13

Try this out yourself: Scale the input vector by 17 times the month of your birthday (85 if you are born in May) and verify that the output vector from the matrix multiplication has changed by the same factor.

$$\begin{pmatrix} a & b \\ c & d \end{pmatrix} \begin{pmatrix} 85x \\ 85x' \end{pmatrix} = \begin{pmatrix} a85x + b85x' \\ c85x + d85x' \end{pmatrix} = 85 \begin{pmatrix} ax + bx' \\ cx + dx' \end{pmatrix} \tag{3}$$

Exercise 14

Display (a) the average position of the particles along the beam line. Likewise, display (b) the angular divergence.

In the following code snippet, we assume that the matrices `Racc` are previously calculated and we start by defining the initial beam `beam0`. Then we propagate `beam0` through the beam line and after each location, we store the average position and the angular divergence.

```
      :
sigx=1; x0=0;
sigxp=0.5; xp0=1;
beam(1,:)=sigx*beam(1,:)+x0;
beam(2,:)=sigxp*beam(2,:)+xp0;
beam0=beam;   % store initial beam
data=zeros(nmat,2);
for k=1:nmat
  beam=Racc(:,:,k)*beam0;
  data(k,1)=mean(beam(1,:));  % (a) average position
  data(k,2)=std(beam(2,:));   % (b) angular divergence
end
subplot(2,1,1); plot(spos,data(:,1))
xlabel('s [m]'); ylabel('x [mm]')
subplot(2,1,2); plot(spos,data(:,2))
xlabel('s [m]'); ylabel('\sigma''_x [mm]')
```

After the loop we display the position and angular divergence on two panels in the same figure.

Exercise 15

What happens if you (a) reduce or (b) increase the initial angular divergence `sigxp` *by a factor of two?*

You should observe that the beam position remains unaffected but the angular divergence grows by the applied factor. If you had plotted the beam size as well, you had seen that it is also scaled by the same factor.

Exercise 16

Using Equation 7, display (a) the average position of the particles along the beam line. Likewise, (b) display the angular divergence. Compare with the result you found in Exercise 14.

We assume that the beam line an the initial beam `beam0` are already prepared such that we can immediately propagate all particles in `beam0` through the beam line and store the beam size at each location. Then we define the input ray `X0` that defines the average position of all particles and the matrix `sigma0` from the initial beam size and angular divergence. The next loop over `k` propagates these quantities using Equation 7 and stores the beam size, which is the square root of σ_{11}

```
      :
data=zeros(nmat,2);
for k=1:nmat    % propagate all the particles
  beam=Racc(:,:,k)*beam0;
  data(k,1)=std(beam(1,:));
end
X0=[x0;xp0];
sigma0=[sigx^2,0;0,sigxp^2];
for k=1:nmat    % propagate the sigma matrix
  X=Racc(:,:,k)*X0;
  sigma=Racc(:,:,k)*sigma0*Racc(:,:,k)';
  data(k,2)=sqrt(sigma(1,1));
end
plot(spos,data(:,1),'k',spos,data(:,2),'*')  % plot both
xlabel('s [m]'); ylabel('\sigma_x [mm]')
```

Finally, we plot both beam sizes and annotate the axes.

Exercise 17

Can you find an initial beam matrix `sigma0` *that reproduces itself at the end of the beam line?*

in the following code you can change the entries in `sigma0` and try obtain the same as `sigma`.

```
beamoptics;   % with beamline=fodo
sigma0=[20,0;0,1];   %  <-----change this
sigma=Racc(:,:,end)*sigma0*Racc(:,:,end)'
```

The point is that it is rather tricky to find the periodic beam matrix by trial and error. The solution, however, is the same that you get from Equation 10 with α and β from Equation 9.

Exercise 18

Explore different initial coordinate and compare the phase-space plots you obtain.

The following code prepares the matrices `Racc` in the function `beamoptics` and, after defining the initial coordinate `x`, it tracks the particle for `Nturn` iterations. After each iteration the position `x(1)` and the angle `x(2)` of the particle are stored in the array `data`.

```
beamoptics;       % with beamline=fodo
Nturn=100; data=zeros(Nturn,2);
x=[1;0];    % <-----play with these numbers
for k=1:Nturn
  x=Racc(:,:,end)*x;
  data(k,1)=x(1);
  data(k,2)=x(2);
end
plot(data(:,1),data(:,2),'.')
xlabel( 'x [mm]'); ylabel('x'' [mrad]')
```

After the loop completes, we prepare a phase-space plot, where we display the angle versus the position for the `Nturn` iterations, which shows an ellipse. Changing the initial coordinates changes the size of the ellipse, but not the shape or the orientation.

Exercise 19

Execute `plot(data(:,1))` *to observe the turn-by-turn positions. What do you observe?*

Replace the last two lines in the previous exercise by these

```
plot(data(:,1))    % <----- only plot position vs turn number
xlabel( 'Turns'); ylabel('x [mrad]')
```

and you should observe oscillations.

Exercise 20

In the definition of `fodo` *at the top of* `beamoptics.m` *reverse the polarity of both quadrupoles and prepare a phase-space plot. How does it differ from the one in Exercise 18?*

The ellipse is now tilted the other way, but size and shape are unchanged, apart from the direction of the tilt.

Exercise 21

Prepare an array describing a FODO cell that starts immediately following the quadrupole with the negative focal length and prepare the phase-space plot.

Compared to Exercises 7 and 8 the definition of the FODO cell is changed to

```
fodo=[ 1, 10,  0.2,  0;    % 10* D(L/10)
       2,  1,  0.0,  F;    % QF
       1,  5,  0.2,  0;    % 5* D(L/10)
       1,  5,  0.2,  0;    % 5* D(L/10)
       2,  1,  0.0, -F];   % QD
```

where we simply moved the first two lines to the end. As a consequence, the shape of the ellipse has changed somewhat and the tilt angle is reversed, because the beam line from Exercise 7 and 8 starts after `QF`, whereas this one starts after `QD`.

Exercise 22

Find the range of focal lengths `F` *for which the FODO cells permit stable oscillations.*

The phase advance μ from Equation 9 must be real, which implies that $-1 < \arccos\left(\frac{R_{11}+R_{22}}{2}\right) < 1$ where R is the product of the transfer matrices for the FODO cell $R = D(L/2)Q(f)D(L)Q(-f)D(L/2)$, which is

$$R = \begin{pmatrix} 1 - \frac{L}{f} - \frac{L^2}{2f^2} & 2L - \frac{L^3}{4f^2} \\ -\frac{L}{f^2} & 1 + \frac{L}{f} - \frac{L^2}{2f^2} \end{pmatrix} . \tag{4}$$

The above condition then becomes $-1 < 1 - L^2/2f^2 < 1$ or $0 < |f| < L/2$.

We can also calculate calculate the tune for the beam line in the following code snippet for many values of the focal length `F`.

```
F=-3:0.01:3;  % range of focal lengths to test
data=zeros(length(F),1);
for k=1:length(F)
  fodo=[ 1,  5,  0.2,  0;       % 5* D(L/10)
         2,  1,  0.0, -F(k);    % QD
         1, 10,  0.2,  0;       % 10* D(L/10)
         2,  1,  0.0,  F(k);    % QF/2
         1,  5,  0.2,  0];      % 5* D(L/10)
  beamline=fodo;
  [Racc,spos,nmat,nlines]=calcmat(beamline);
  [Q,alpha,beta,gamma]=R2beta(Racc(:,:,end));
  if (~isreal(Q)) data(k)=1; end
end
plot(F,data,'*');
xlabel('F [m]'); ylabel('unstable = 1'); ylim([0,1.1])
```

On the plot, the focal lengths that allow stable oscillations are indicated by a value of 1 and 0 otherwise. Since in this example $L = 2$ m, we find that the values $-1 < f < 1$ allow stable oscillations.

Exercise 23

Run `periodic_beam` *and convince yourself that the sigma matrix at the end of the cell is indeed equal to* `sigma0`.

We assume that the beam line is defined and the matrices `Racc` are already calculated, before determining the Twiss parameters with `R2beta()` and using them to define the periodic beam matrix `sigma0` from Equation 10. In the loop over `k` we then propagate `sigma0` through the beam line and, after the loop finishes, plot the final beam matrix `sigma`, which should be the same as `sigma0`.

```
        :
[Q,alpha,beta,gamma]=R2beta(Racc(:,:,end));
data=zeros(nmat,1);  % allocate memory for display
eps=1; % set emittance to unity
sigma0=eps*[beta, -alpha;-alpha,gamma]
for k=1:nmat
  sigma=Racc(:,:,k)*sigma0*Racc(:,:,k)';
  data(k,1)=sqrt(sigma(1,1));
end
sigma=sigma % should be same as sigma0
```

Exercise 24

Write down the numerical values of initial beam matrix `sigma0`*, then build a beam line made of 15 consecutive cells by changing the definition of* `beamline` *in* `beamoptics.m` *and then, using* `sigma0` *with the noted-down numbers, prepare a plot of the beam sizes along the 15 cells. Is it also periodic?*

We first define the beam line for one FODO cell and calculate `Racc` such that the function `R2beta` returns the phase advance for the cell and Twiss parameters at the start of the cell. The latter we use to define the initial beam matrix `sigma0`. We continue by defining the beam line for 15 consecutive cells and use `calcmat()` to calculate the `Racc` for this beam line of 15 cells. Now it is easy to propagate the initial beam `sigma0` through the 15 cells and record the beam size $\sqrt{\sigma_{11}}$ at each location.

```
F=2.5;   % focal length of the quadrupoles
fodo=[ 1,  5,  0.2,  0;    % 5* D(L/10)
       2,  1,  0.0, -F;    % QD
       1, 10,  0.2,  0;    % 10* D(L/10)
       2,  1,  0.0,  F;    % QF
       1,  5,  0.2,  0];   % 5* D(L/10)
beamline=fodo;  % beam line with one cell to determine sigma0
[Racc,spos,nmat,nlines]=calcmat(beamline);
[Q,alpha,beta,gamma]=R2beta(Racc(:,:,end));
eps=1;
sigma0=eps*[beta, -alpha;-alpha,gamma]; % initial sigma matrix
beamline=repmat(fodo,15,1);  % beam line with 15 cells
[Racc,spos,nmat,nlines]=calcmat(beamline);
data=zeros(nmat,1);
for k=1:nmat
  sigma=Racc(:,:,k)*sigma0*Racc(:,:,k)';
  data(k,1)=sqrt(sigma(1,1));
end
plot(spos,data(:,1),'k')
xlabel('s [m]'); ylabel('\beta [m]')
```

Finally we plot the beam size along the beam line.

Exercise 25

Verify that all matrices `Racc(:,:,k)` *have unit determinant.*

After the beam line is prepared and the matrices `Racc` are available, the following code snippet calculates the determinant of each matrix and plots its value by an asterisk.

```
  :
data=zeros(nmat,1);
for k=1:nmat
  data(k,1)=det(Racc(:,:,k));
end
plot(spos,data,'*');
xlabel('s [m]'); ylabel('det(R)'); ylim([0,2]);
```

All displayed asterisks must be 1.

Exercise 26

Multiply `sigma0` *from Exercise 24 by 17 and calculate the emittance. Then propagate the sigma matrix through the beam line from Exercise 24 and verify that the emittance of the sigma matrix after every element is indeed constant and equal to its initial value.*

After defining the FODO cell, we use `R2beta()` to calculate the periodic Twiss parameters and the beam matrix `sigma0` with the entries increased by a factor of 17, we re-calculate the matrices `Racc`, this time for a beam line with 15 cells, and propagate `sigma0` through the beam line. At each location, we record the beam size $\sqrt{\sigma_{11}}$ and the emittance, which is the square root of the determinant of the beam matrix `sigma`.

```
            :
beamline=fodo;  % beam line with one cell to determine sigma0
[Racc,spos,nmat,nlines]=calcmat(beamline);
[Q,alpha,beta,gamma]=R2beta(Racc(:,:,end));
eps=1*17;
sigma0=eps*[beta, -alpha;-alpha,gamma]; % initial sigma matrix
beamline=repmat(fodo,15,1);    % beam line with 15 cells
[Racc,spos,nmat,nlines]=calcmat(beamline);
data=zeros(nmat,2);
for k=1:nmat
  sigma=Racc(:,:,k)*sigma0*Racc(:,:,k)';
  data(k,1)=sqrt(sigma(1,1));
  data(k,2)=sqrt(det(sigma));   % <-----calculate the emittance
end
subplot(2,1,1); plot(spos,data(:,1),'k')
xlabel('s [m]'); ylabel('\beta [m]')
subplot(2,1,2); plot(spos,data(:,2),'*')
xlabel('s [m]'); ylabel('det(\sigma)')
```

After the loop the beam size and the emittance are plotted in two panels. The emittance on the lower panel is constant.

Exercise 27

Vary `F` *by hand and try to (a) find a value that returns* $Q = 1/6$. *(b) Then try to find a value of* `F` *that produces a* 90^0 *phase-advance. What is the corresponding value of* Q*?*

Changing the value of `F` in the code below will return the phase advance $Q = \mu/2\pi$ of the cell in "tune units".

```
%F=2;      % for 60 degrees, Q=1/6
F=sqrt(2); % for 90 degrees, Q=0.25
fodo=...   % same definition as in Exercise 24
beamline=fodo;
[Racc,spos,nmat,nlines]=calcmat(beamline);
[Q,alpha,beta,gamma]=R2beta(Racc(:,:,end));
Q=Q        % display tune
```

`F=2` produces a tune of $1/6$, which is commonly referred to as a 60-degree phase advance per cell. Setting it to $\sqrt{2}$ produces a tune of 0.25, or at 90-degree phase advance per cell.

Exercise 28

Implement the procedure described in the previous paragraph.

In the following code we define the beam lines for the 60-degree and 90-degree FODO cells and find their periodic beam matrices `sigma60` and `sigma90` respectively. Then we define a long beam line and vary the focal lenths of the quadrupoles in the second cell by hand until the beam size, calculated in the loop over `k` and then displayed, in the ten subsequent 90-degree cells oscillates regularly and without beating.

```
F=2;       % for 60 degrees
fodo60= ... % same as in Exercise 24
beamline=fodo60;
[Racc,spos,nmat,nlines]=calcmat(beamline);
[Q,alpha,beta,gamma]=R2beta(Racc(:,:,end));
eps=1; sigma60=eps*[beta, -alpha;-alpha,gamma];  % <-- sigma60
F=sqrt(2);  % for 90 degree
fodo90= ... % same as in Exercise 24
beamline=fodo90;
[Racc,spos,nmat,nlines]=calcmat(beamline);
[Q,alpha,beta,gamma]=R2beta(Racc(:,:,end));
eps=1; sigma90=eps*[beta, -alpha;-alpha,gamma];  % <-- sigma90
beamline=[fodo60;fodo60;repmat(fodo90,10,1)];    % long beam line
beamline(7,4)=-2.03;   % play with QD in second cell
beamline(9,4)=1.66;    % play with QF in second cell
[Racc,spos,nmat,nlines]=calcmat(beamline);
data=zeros(nmat,1);
for k=1:nmat
  sigma=Racc(:,:,k)*sigma60*Racc(:,:,k)';
  data(k,1)=sqrt(sigma(1,1));
end
plot(spos,data(:,1),'k')
xlabel('s [m]'); ylabel('\beta [m]')
```

The beating in the 90-degree cells is minimized with the settings given in the code above.

Exercise 29

Include these matrices in `calcmat()` *and assign code* `5` *to them. Hint: write an external function that returns* $Q(l, k_1)$ *and call it from* `calcmat()`.

The code from Equation 13 and 14 is encoded in the following function which is stored in the file `QQ.m`

```
% QQ.m, 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
```

Moreover, we add the following code snippet in `calcmat()`

```
    case 5   % thick quadrupole
       k1=beamline(line,4);
       Rcurr=QQ(beamline(line,3),k1);
```

and are ready to use long quadrupoles in our simulations.

Exercise 30

Use the beam line from Exercise 27 (60 degrees/cell FODO) and replace the thin quadrupoles by long quadrupoles with a length of 0.2, 0.4, 1.0 m. Make sure the overall length and the phase advance of the FODO cell remains unchanged. By how much does the periodic beta function at the start of the cell change? Express the change in percent.

Here we first prepare a thin-lens FODO beam line `fodo0` and then define the strength `K1` of a quadrupole with length $0.2\,\mathrm{m}$ before preparing a beam-line description. Note that we have to adjust the length of the drift space adjacent to the quadrupole in order to keep the overall length of the cell constant. Then we use `calcmat2()`, which already contains the code for the long quadrupole from Exercise 29, and use `R2beta()` to give us the phase-advance per cell in "tune units." We then have to vary `K1` until the tune is $1/6$. The value in the code is the final result.

```
L=2; F=2;
fodo0= ... % same as Exercise 24 or 27
K1=2.588;  % fiddle until the tunes are right, 0.2 m value
fodo1=[1,  9, L/20,   0;   % thick lens lattice
       5,  1,  0.2, -K1;
       1, 18, L/20,   0;
       5,  1,  0.2,  K1;
       1,  9, L/20,   0;];
beamline=fodo1;                  % select FODO cell here
[Racc,spos]=calcmat2(beamline);  % calculate matrices
Rturn=Racc(:,:,end);             % full turn matrix
[Q,alpha,beta,gamma]=R2beta(Rturn);
beta=beta    % display the beta
Q=Q          % display the tune, should be 1/6
```

We emphasize that the length of the cell in all cases must be equal to the length of the beam line with the thin-lens quadrupoles. We find the quadrupole excitations k_1 by first setting the value such that their integrated strength $k_1 L \approx 1/f$ and then slightly varying the value until the tune is again $1/6$. These values of k_1 are reported in the following table

L [m]	0	0.2	0.4	1	Units
k_1	-	2.588	1.343	0.613	[1/m^2]
β	4.04	4.06	4.08	4.12	[m]
$\Delta\beta/\beta$	-	0.5	1	2	%

The two lower rows report the beta function β at the start of the cell and the relative difference $\Delta\beta/\beta$ to the value from the thin-lens beam line. We observe that replacing thin-lens by long quadrupoles only changes the beta functions very little. Therefore one often does a first design of a new beam line with thin-lens magnets and later lets a computer do the fine adjustments.

Exercise 31

Include this matrix in `calcmat()` *and assign the code* `4` *to it.*

The function `SB()` returns the transfer matrix for the horizontally deflecting sector dipole, given by Equation 15. Note that a turned-off dipole with $\phi = 0$ returns the matrix of a drift space with the same length as the dipole.

```
% SB.m, sector bend
function out=SB(L,phi);
out=eye(2);
if abs(phi)<1e-8
  out(1,2)=L;
else
  rho=L/phi;
  out=[cos(phi),rho*sin(phi); -sin(phi)/rho,cos(phi)];
end
```

We include the dipole in `calcmat()` by adding the following stanza to the case statement. Note that we have to convert the bending angle, which is specified in degrees, to radians.

```
    case 4   % sector bend
       phi=beamline(line,4)*pi/180;   % to radians
       Rcurr=SB(beamline(line,3),phi);
```

Exercise 32

Insert 1 m long dipoles in the center of the drift spaces of the FODO cells from Exercise 27 while keeping the length of the cell constant. To avoid splitting a dipole, start the beam-line description in the middle of a focusing quadrupole and investigate deflection angles of $\phi = 5, 10,$ and 20 degrees. Check by how much the periodic beta functions change. Why do they change? Explain! Can you compensate the phase advance μ by adjusting the strength or focal lengths of the quadrupoles?

In order to place dipoles in both drifts, we start the FODO cell in the middle of a focusing quadrupole; the first and last quadrupoles in the following beam-line description are one half of the split-in-the-middle quadrupole. For reference, we first define a cell without dipole `fodo0`, followed by a cell `fodo1` that has 1 m-long dipoles inserted in the middle of the drift spaces, which are accordingly shortened in order to maintain the overall length of the cell. We have split the dipole in ten slices, which makes the plot of the beam sizes look nicer, but this requires that each slices only bends by a tenth of the angle. After the beam lines are defined, we calculate the transfer matrices `Racc` and determine the periodic beam matrix `sigma0` that we subsequently propagate through the beam line and finally display.

```
L=2; F=2;
fodo0=[2,   1,    0,  2*F;
       1,  20, L/20,  0;
       2,   1,    0, -F;
       1,  20, L/20,  0;
       2,   1,    0,  2*F];
F=2.057;
phi=5/10;  % five degrees in ten slices
% F=2.262; phi=10/10;   % ten degrees in ten slices
% F=6.05;  phi=20/10;   % twenty degrees in ten slices
fodo1=[2,   1,    0,  2*F;
       1,   5, L/20,  0;
       4,  10, 0.1,   phi;   % ten slices
       1,   5, L/20,  0;
       2,   1,    0, -F;
       1,   5, L/20,  0;
       4,  10, 0.1,   phi;   % ten slices
       1,   5, L/20,  0;
       2,   1,    0,  2*F];
beamline=fodo1;                  % select FODO cell here
nmat=sum(beamline(:,2))+1;
[Racc,spos]=calcmat2(beamline);  % calculate matrices
Rturn=Racc(:,:,end);             % full turn matrix
[Q,alpha0,beta0,gamma0]=R2beta(Rturn);
Q=Q      % display the tune, vary F until it is 1/6
sigma0=[beta0, -alpha0; -alpha0,gamma0]
for k=1:nmat
   sigma=Racc(:,:,k)*sigma0*Racc(:,:,k)';
   beta(k)=sigma(1,1);
end
plot(spos,beta); xlabel('s [m]'); ylabel('\beta [m]')
```

The task is now to vary `F` until the tune is $1/6$ and then read off the initial beta function `beta0` and compare with the value from the beam line without dipole, which is 6.93 m. For a 5-degree dipole `beta0` is 6.80 m, for 10 degrees it is 6.40 m, and for a 20-degree it is 4.52 m. We observe that stronger dipoles reduce the beta function because they focus in their bending plane, where they act similar to an additional quadrupole. This is also the reason why the focal length of the quadrupoles in the 20-degree case is so weak. The dipole does much of the focusing, such that we can reduce the excitation of the quadrupole to maintain the tune at 1/6.

Exercise 33

Upgrade the software to consistently handle 3×3 matrices for drift space, quadrupoles, and sector dipoles.

The code is available from the CAS web site at https://cas.web.cern.ch/previous-schools. Check out the subdirectory named 3D.

Exercise 34, and 35

Build a beam line of six FODO cells with a phase advance of 60 degrees/cell (thin quadrupoles are OK to use) and add a sector bending magnet with length 1 m and bending angle $\phi = 10$ degrees in the center of each drift. You may have to play with the quadrupole values to make the phase advance close to 60 degrees. But you probably already did this in Exercise 32.
Use the starting conditions $(x_0, x_0', \delta) = (0, 0, 0)$ and plot the position along the beam line. Repeat this for $\delta = 10^{-3}$ and for $\delta = 3 \times 10^{-3}$. Plot all three traces in the same graph. Discuss what you observe and explain!

This script must be executed in the subdirectory 3D, which contains all the functions to handle 3×3 transfer matrices. In Exercise 34 we use the description `fodo1` with values for `phi` and tt F pertaining to the case with 10-degree dipoles and then repeat hat cell six times. Then we calculate all 3×3 transfer matrices `Racc`. The initial ray `x0` is then specified in terms of the momentum deviation δ and propagated through the beam line, before being plotted.

```
L=2; F=2.262;    % copy from ex32.m
phi=10/10;       % ten degrees in ten slices
fodo1= ...       % from ex32.m
beamline=fodo1;
nmat=sum(beamline(:,2))+1;
[Racc,spos]=calcmat2(beamline);  % calculate matrices
Rturn=Racc(:,:,end);             % full turn matrix
[Q,alpha0,beta0,gamma0]=R2beta(Rturn(1:2,1:2));  % only 2x2 transverse part
%.............................ex34.m above and ex35.m below
beamline=repmat(fodo1,6,1);      % ex34
nmat=sum(beamline(:,2))+1;       % total number of slices
[Racc,spos]=calcmat2(beamline);  % calculate matrices
xpos=zeros(nmat,1);  % allocate array to store values to display
delta=1e-3;          % <---change delta here
x0=[0;0;delta];      % change third component, which is dp/p
for k=1:nmat
  x=Racc(:,:,k)*x0;
  xpos(k)=x(1);
end
plot(spos,xpos); xlabel('s [m]'); ylabel('x [m]')
```

With $\delta = 0$ the particle's trajectory hugs the optical axis. For the two non-zero values of δ the trajectories look similar, though scaled by a factor of three. In every dipole, the particle receives a small kick that is proportional to δ, continues to propagates just like any other particle, and is affected by quadrupoles such that it performs betatron oscillations. We also observe that the trajectories are again close to the optical axis after the six cells, a behavior that repeats if the beam line is extended by another, say 10, cells. This periodicity reflects the value of the tune of the cell, which is 1/6, such that after six cells the tune is unity. Equivalently, the transfer matrix through the whole section is very close to a unit matrix.

Exercise 36 and 37

Work out the transverse components of the periodic beam matrix σ_0. Assume that the emittance is $\varepsilon_0 = 10^{-6}$ meter-rad. Furthermore, assume that the momentum spread $\sigma_0(3,3) = \sigma_p^2$ is zero and plot the beam size along the beam line.
Plot the beam size for for $\sigma_p^2 = 10^{-3}$ and for $\sigma_p^2 = 3 \times 10^{-3}$. What happens if you change the phase advance of the cell? Try out by slightly changing the focal lengths.

We first calculate the periodic beta functions for a single cell before preparing the beam line with six cells and calculating `Racc`. Then we prepare the initial beam matrix `sigma0` that contains the usual 2×2 beam matrix in the upper left corner and the momentum spread squared in the bottom right. In the loop over `k` we propagate `sigma0` through the beam line and records the beam size along the way, which is finally plotted.

```
          :
[Q,alpha0,beta0,gamma0]=R2beta(Rturn(1:2,1:2)); % for single cell
beamline=repmat(fodo1,6,1);
nmat=sum(beamline(:,2))+1;
[Racc,spos]=calcmat2(beamline);  % calculate matrices
%.........plot the beam sizes with different momentum spread
sigma0=eye(3);
eps0=1e-6;              % 1 mm-mrad emittance
sigma0(1:2,1:2)=eps0*[beta0, -alpha0; -alpha0,gamma0];
sigp=1e-3;              % momentum spread
sigma0(3,3)=sigp^2;     % momentum spread squared
sigmax=zeros(nmat,1);  % allocate array
for k=1:nmat
  sigma=Racc(:,:,k)*sigma0*Racc(:,:,k)';
  sigmax(k)=sqrt(sigma(1,1));
end
plot(spos,sigmax)
xlabel('s [m]'); ylabel('\sigma_x [m]')
```

Varying the focal length slightly changes the observed beating of the dispersion.

Exercise 38

Determine the periodic dispersion at the start of the cell. Then plot the dispersion in the cell.

We first calculate the periodic dispersion `D0` and turn it into a 3×1 vector with a 1 in the third position and then use `Racc` to propagate this vector through the beam line. We save the first component, which is the dispersion in `Dx`, and display it once the loop has finished.

```
          :
D0=(eye(2)-Rturn(1:2,1:2))\Rturn(1:2,3)  % periodic dispersion
D0=[D0;1];           % patch a '1' into the third component
```

```
Dx=zeros(nmat,1);    % allocate array to store values to display
for k=1:nmat
  x=Racc(:,:,k)*D0;
  Dx(k)=x(1);
end
plot(spos,Dx); xlabel('s [m]'); ylabel('D_x [m]')
```

Exercise 39

Convert the code to use 4×4 matrices, where the third and fourth columns are associated with the vertical plane. Create a separate subdirectory for the calculations with the 4×4 matrices.

The code is available from the CAS web site. Check out the subdirectory named 4D.

Exercise 40 and 41

Start from a single FODO cell with 60 degrees/cell you used earlier. Insert sector bending magnets with a bending angle of $\phi = 10$ degrees in the center of the drift spaces. The bending magnets will spoil the phase advance in one plane. Now you have two phase advances and need to adjust both quadrupoles (by hand to 2 significant figures) such that it really is 60 degrees in both planes.
Use the result from Exercise 2 and adjust the two quadrupoles such that the phase advance in the horizontal plane is 90 degrees, cell, while it remains 60 degrees/cell in the vertical plane.

This as all following exercises must be executed in the subdirectory named 4D. We first prepare the FODO cell with 10-degree dipoles, but now have to give different focal lengths to the focusing and defocusing quadrupoles, because the focusing from the dipole affects the horizontal plane only, but once we use QF to compensate it, also the tune in the vertical plane is affected. We therefore have to play with both `FF` and `FD` until `Qx` and `Qy` are at their desired value of 1/6. At least they should be close. The values specified achieve that.

```
L=2;
FF=2.18;    % start with FF=FD=2
FD=2.06
phi=10/10;  % ten degree in ten slices
fodo=[2,   1,    0,  2*FF;
      1,   5, L/20,  0;
      4,  10, 0.1,   phi;
      1,   5, L/20,  0;
      2,   1,    0, -FD;
      1,   5, L/20,  0;
      4,  10, 0.1,   phi;
      1,   5, L/20,  0;
      2,   1,    0,  2*FF];
beamline=fodo;
nmat=sum(beamline(:,2))+1;
[Racc,spos]=calcmat2(beamline);  % calculate matrices
Rturn=Racc(:,:,end);             % full turn matrix
[Qx,alphax0,betax0,gammax0]=R2beta(Rturn(1:2,1:2));  % only 2x2 horizontal part
[Qy,alphay0,betay0,gammay0]=R2beta(Rturn(3:4,3:4));  % only 2x2 vertical part
Q=[Qx,Qy]      % display the tune, should be 1/6, but 0.166xx is OK
```

Making the tune in the horizontal plane 1/4 while it stay 1/6 in the vertical plane is approximately achieved by setting `FF=1.54` and `FD=1.84` in the above example.

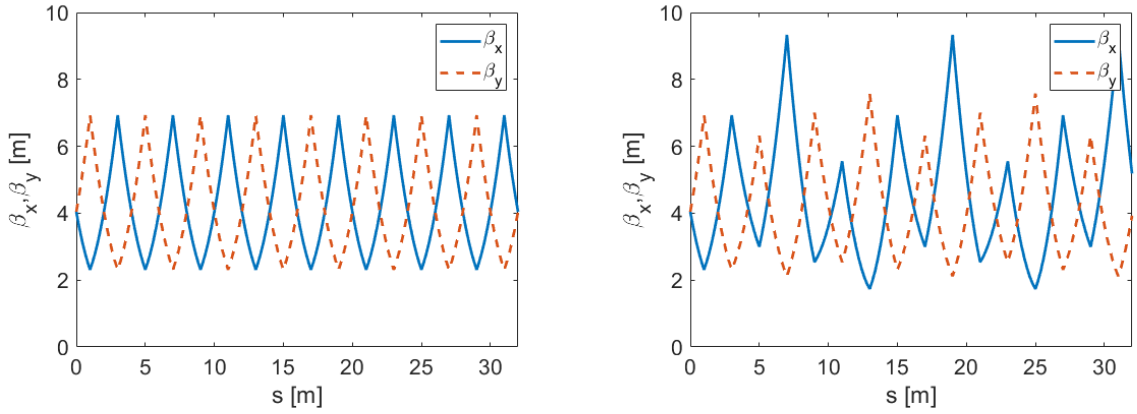

Fig. 1: Beta functions β_x (solid blue) and β_y (dashed red) for the beam line with the eight FODO cells from Exercise 42. The unperturbed values are shown on the left and those, where the second quadrupole at $s = 3\,\mathrm{m}$ has the wrong excitation, are shown on the right.

Exercise 42

Prepare a beam line with eight FODO cells without bending magnets and with 60 degrees/cell phase advance in both planes. (a) Prepare the periodic beam matrix sigma0 (4x4, uncoupled) as the initial beam and plot both beam sizes along the beam line. (b) Use sigma0 as the starting beam, but change the focal length of the second quadrupole by 10 % and plot the beam sizes once again. Discuss you observations.

We use the description of the FODO cell from Exercise 7 and calculate the matrices `Racc`, from which we determine the tune and the beta functions for the horizontal and vertical plane, respectively. The latter we then use to construct the 4×4 beam matrix `sigma0` that we use as input for the simulation. But first we prepare a bad fodo cell that contains a quadrupole with a focal length increased by 10 %, build a beam line with a single bad cell followed by seven good ones, and calculate the `Racc` for this perturbed beam line. Now we are ready to propagate `sigma0` through this beam line and record the beta functions along the way.

```
L=2; F=2;
fodo= ...  % from exercise 7
beamline=fodo;
[Racc,spos]=calcmat2(beamline); Rturn=Racc(:,:,end);
[Qx,alphax0,betax0,gammax0]=R2beta(Rturn(1:2,1:2));  % horizontal
[Qy,alphay0,betay0,gammay0]=R2beta(Rturn(3:4,3:4));  % vertical
Q=[Qx,Qy]      % display the tune, should be 1/4,1/6, but 0.25xxx,0.16xxx is OK
eps=1; sigma0=zeros(4,4);
sigma0(1:2,1:2)=eps*[betax0,-alphax0;-alphax0,gammax0];
sigma0(3:4,3:4)=eps*[betay0,-alphay0;-alphay0,gammay0];
fodobad=fodo;
fodobad(4,4)=fodobad(4,4)*1.1;  % increase focal length by 10 %
beamline=[fodobad;repmat(fodo,7,1)];
nmat=sum(beamline(:,2))+1;
[Racc,spos]=calcmat2(beamline);   % sets up all transfer matrices
beta=zeros(nmat,2);
for k=1:nmat
  sigma=Racc(:,:,k)*sigma0*Racc(:,:,k)';
  beta(k,1)=sigma(1,1);
```

```
  beta(k,2)=sigma(3,3);
end
plot(spos,beta(:,1),spos,beta(:,2),'--','LineWidth',2)
xlabel('s [m]'); ylabel('\beta_x,\beta_y [m]');
legend('\beta_x','\beta_y')
```

Figure 1 shows β_x (solid blue) and β_y (dashed red) for the the beam line consisting of eight good cells on the left for comparison. On the right we see β_x and β_y for and the perturbed beam line, which is messed up pretty badly. Instead of oscillating with equal amplitudes, the beta functions, and with it the beam sizes, are beating, which is why this effect is called *betabeat.* Since we perturbed a horizontally focusing magnet, the horizontal beta function is affected somewhat worse than the vertical.

Exercise 43

From the lecture about betatron coupling identify the transfer matrix for a solenoid and write a function that receives the longitudinal magnetic field B_s and the length of the solenoid as input and returns the transfer matrix. Then extend the simulation code to handle solenoids. Finally, define a beam line where you place the solenoid in the middle of a FODO cell and follow a particle with initial condition $(x_0, x_0', y_0, y_0') = (10^{-3}\,\mathrm{m}, 0, 0, 0)$. What do you observe? Is the motion confined to the horizontal plane?

Our first task is to implement the transfer matrix for a solenoid in `calcmat`, which is given elsewhere in these proceedings, and write a function that calculates it from the length and strength of the solenoid.

```
function out=SOL(L,ks)
if (abs(ks)<1e-10) ks=1e-10; end  % catch zero
phis=ks*L/2; c=cos(phis); s=sin(phis);
Qs=[c, 2*s/ks; -ks*s/2, c];
Rr=[c,0,s,0;
    0,c,0,s;
   -s,0,c,0;
    0,-s,0,c];
out=Rr*[Qs,zeros(2,2);zeros(2,2),Qs];
```

Once that is added as a stanza in the case statement in `calcmat` we prepare a beam-line description where the solenoid is added in the center of the drift and calculate all transfer matrices `Racc`. After defining the input ray `x0`, we propagate it through the beam line and display both horizontal and vertical position along the way.

```
F=2.5;   % focal length of the quadrupoles
fodo=[1,  5,  0.2,  0;    % 5* D(L/10)
      2,  1,  0.0, -F;    % QD
      1,  3,  0.2,  0;    % 3* D(L/10)
     20,  4,  0.2,  1;    % SOL
      1,  3,  0.2,  0;    % 3* D(L/10)
      2,  1,  0.0,  F;    % QF
      1,  5,  0.2,  0];   % 5* D(L/10)
beamline=fodo; Racc,spos,nmat,nlines]=calcmat2(beamline);
x0=[0.001;0;0;0];     % 1 mm offset at start
data=zeros(nmat,2);   % allocate memory
for k=1:nmat
  x=Racc(:,:,k)*x0;
  data(k,1)=x(1);  % store the horizontal position
  data(k,2)=x(3);  % vertical position
end
```

```
plot(spos,1e3*data(:,1),'k',spos,1e3*data(:,2),'k--')
xlabel('s [m]'); ylabel(' x,y [mm]'); legend('x','y')
```

The plot shows that the vertical position stays on the optical axis until the solenoid, where it starts betatron oscillations. The reason is that the quadrupole just upstream of the solenoid gives the trajectory a horizontal angle, such that it enters the solenoid at an angle, which causes a helical orbit inside the solenoid. After the solenoid ends the trajectory is non-zero in both horizontal and vertical planes and continues to oscillate.