The *Lattice* Package

A Maple package for charged-particle optics and beam-line analysis.

H.-Ulrich (Uli) Wienands
SLAC National Accelerator Laboratory
Menlo Park, CA 94025, USA
uli@slac.stanford.edu

Date of this manual: February 10, 2016.

Introduction

The *Lattice* package is a Maple package to design and analyze charged-particle beam lines and circular machines. It employs a beam-line description using the standard elements (dipoles, quadrupoles and so on) and retains the algebraic power of Maple. Beam-line elements are described using the equations governing the particle motion in algebraic form. In this way it is possible to compute expressions (rather than numbers) for beam-line parameters like Twiss functions, dispersion and such, for beam lines or rings, and to perform analysis on these expressions using the full power of Maple. The user can—within the limits of Maple's capabilities and of memory limitations—demonstrate analytically that certain characteristics can in fact be obtained using a given beam line, or not, as the case may be. Numeric calculations are possible as well to find values for magnet settings, track particle coordinates, generate lattice plots etc.

The *Lattice* package takes a hybrid approach to the computations for a beam line. First-order analysis is done by using 6x6 TRANSPORT[1] matrices. Thus most algorithms and examples found in the literature can be modeled essentially unchanged with the *Lattice* package. For particle tracking and modeling of nonlinear elements a tracking function (map) is included for each element. The tracking function of a beam line is computed by composition of the tracking functions for each element of the beam line and no truncation is done. This ensures accurate and symplectic tracking as long as the tracking function for each element is symplectic. A beam can be defined for use in tracking, including its first-order defining $\sum$ matrix[2]. Plotting commands are provided to allow for simple plotting of lattice functions and phase-space portraits. Output in a format suitable for input to MAD8[3] can be generated.

Since the full power of Maple is available to the user, the package does not have special matching or fitting routines but rather relies on the extensive capabilities of the Maple programming language to facilitate such operations. Likewise, operations like series expansion to a desired order can be performed using the *series*, *taylor* and *mtaylor* functions of Maple.

The coordinates the *Lattice* package uses are positions and angles in the beam-following Frenet-Serret coordinate system, i.e. a particle's coordinates in 6-d phase space are described by $\langle x,x',y,y',l,dp\rangle$. These are not canonical in the Hamiltonian sense.

The *Lattice* package implements the commonly used methods to model beam lines and circular machines. Common beam line elements are available: **Drift**, **Quad**, **Bend**, **Sextupole**, **RfCavity** as well as **Solenoid**, **GKick**, **Foil**, **Wire.** These elements are implemented as Maple *Record*s, which are

instantiated upon calling the respective element type procedures that return the *Record* implementing the element.

Beam lines are built using the **DefineLine** procedure which returns a *Record* implementing a whole beam line. For certain calculations one may desire each element instantiated separately; this can be done using the **ExpandLine** procedure which returns a Maple *Vector* of the elements of the beam line.

The individual parameters of each element or beam line are available using the standard Maple :- ("member of") operator. This allows quick extraction e.g. of the first-order TRANSPORT matrix *R* to perform operations on using Maple's LinearAlgebra package or other specialized operations. In certain cases it is also possible to replace the value of parameters in a given element, although this requires care to avoid an element record having inconsistent parameters and a better mechanism is provided by the **Subs** operation. The parameters available for each element are described in the next section.

Procedures are available for computing the matched Twiss functions and dispersion: **twiss** returns a maple *Vector* with the 6 Twiss functions, **dispersion** returns a *Vector* with the 4 dispersion functions. Functions for the individual quantities also exist: **betax**, **alphax**, **gammax** and their vertical counterparts as well as **etax**...**etapy**.
Tunes and cosines of the phase advance are computed by **cosmux**, **cosmuy** and **nux**, **nuy**.
Twiss functions and dispersion can be propagated through a lattice using **TwissTran** and **EtaTran**.

Particle tracking is supported by providing a function **DefineBeam** that sets up the data structure for a particle beam, which is characterized by its $\sum$ matrix and can have particles, as well as the **Track** procedure that actually performs the tracking. Due to speed limitations this is not practical for large-scale multi-turn tracking studies (e.g. to find the dynamic aperture of a ring) but rather for exploratory limited studies. However, it is possible to extract and convert the tracking function to a numeric polynomial map the evaluation of which is actually quite efficient. Using *mtaylor*, this map can be limited to a desired order for further increase in tracking speed.

Lattice function plot structures can be generated using the **LatticePlot** procedure and put on the screen using *plots:-display*.
Beam distribution and phase-space ellipse plot structures are generated with **BeamPlot**.
Synchrotron-radiation integrals are computed by the functions **I1**, **I2**, **I3**, **I4x** and **I5x**. These act on *ExpandedLine*s.
The procedure **Subs** can be used to substitute variables on elements and beam lines of the *Lattice* package. It changes all occurrences of a variable in the given element or beam line. It recursively changes any sublines as well. Note that this is preferable to changing members of the element record as Subs ensures that all occurrences of a variable are changed and the element record remains consistent.
The symplectic 6x6 Matrix **J** is defined in *Lattice* and can be used together with *VectorCalculus:-Jacobian* to check symplecticity of an operation or element or beam-line map.
Output for the lattice program MAD8 can be produced using **Mad8Form**.

Most of the accelerator physics formulae used in *Lattice* are from Ref. [4]. Units used in *Lattice* are meter, radian, MeV, MV, Tesla unless otherwise noted.

The *Lattice* package is compatible with Maple versions 15 and later.

Implementation details

The machine elements are represented by Maple *Record*s. Each element has a somewhat different set of parameters stored in its record, but a number of parameters are common to all elements:

- *l* - the length of the element. This length is understood to be the path length unless otherwise noted. Procedures like **DefineLine** or **ExpandLine** keep track and recalculate the length as appropriate.
- *R* - the first-order 6x6 R matrix. This is used for first-order calculations like twiss and dispersion. For elements with only higher-order terms, *R* implements a drift of the equivalent length. Some elements 0 length will have *R=IdentityMatrix*.
- *TF* - the trackFunction or map. This function describes the passage of particles through the element and can include to higher than first order. It is implemented with the Maple arrow operator. *TF* always acts on the column Vector with the 6 particle coordinates, *<x,xp,y,yp,l,dp>*. Elements like **Foil** and **Collimator** can add random values to the beam coordinates.
- *kind* - This is a name denoting the kind of the element. In general it is the same as the name of the constructor (**Drift**, **Quad** etc.).
- *Eref* - a reference energy. Not always assigned a value. Used if elements change the average energy of a beam, like synchrotron radiation in a bend.

A general note: *Record* elements in Maple and therefore in the *Element*s of *Lattice* can be changed. However, the parameter values of the elements in *Lattice* are interdependent; this interdependence gets lost if only one parameter value gets changed by assignment. As a result such assignments lead to erroneous results unless great care is taken. The **Subs** procedure is provided to allow change of parameter values that will maintain consistency across all parameters of an element. **Subs** also allows to change parameters in *BeamLines* in a correct and consistent manner.

Lattice defines a number of types that are used to restrict the operation of the procedures in *Lattice* to its own types. In a few instances this is used to overload Maple operations. The types defined in *Lattice* are:

- Element - An element like a **Drift**, **Quad** etc. Any *BeamLine* is also an *Element*. An *ExpandedLine* is a Vector of *Elements*.
- BeamLine - A *BeamLine* is a concatenation of several *Elements*. A *BeamLine* is distinct from a mere *Element* type by having a parameter *BL*, which is assigned the list of *Elements* making up the *BeamLine*. The other parameters of a *BeamLine* are the combination of the parameters from the *Elements* of the *BeamLine*, i.e. the length *l* is the total length, the R Matrix is the total R matrix and the trackFunction *TF* is the total trackFunction.
- ExpandedLine - a Vector of *Elements*. In general, the components of this Vector are *Elements* and not *BeamLines*. Sublines are expanded in the *ExpandLine* procedure. The *Elements* of an *ExpandedLine* have the following parameters:
  - *s* - the total distance from the beginning of the exit of the element
  - *R* - the R matrix of this element (not cumulative)
  - *TF* - the trackFunction of this element (not cumulative)
  - *kind* - the kind of this element
  - *Eact* - the actual beam energy at the exit of this element. (this allows evaluations of e.g. the energy sawtooth of an electron storage ring)
  - any other parameter the present element has.
- Machine - A *BeamLine* with the addition of a Title. Certain operations (like **Mad8Form**) require a *Machine* as argument.

Initialization

As usual with Maple packages, the package file Lattice.mla has to be placed in a directory (folder) where Maple can find it; the same is true for the help database Lattice.help (or Lattice .hdb for Maple versions before 18). The correct locations depend on the operating system used (Windows *vs* Mac OS X *vs* Linux) and the specific setup of the user. Loading *Lattice* using the *with* command will produce a confirmatory message even with its output suppressed.

restart

"Maple Initialization loaded..." **(3.1)**

with(*Lattice*)

"Lattice.mw, Version 1.0.2, 10-Feb-2016"

[`.`, *BeamPlot*, *Bend*, *DefineBeam*, *DefineElement*, *DefineLine*, *DefineMachine*, *Drift*, *Edefault*, *ElnamT*, *EtaTran*, *ExpandLine*, *Foil*, *GKick*, *H*, *Hx*, *HxTran*, *Hy*, *I1*, *I2*, *I3*, *I4x*, *I5x*, *J*, *LatticePlot*, *LumpLine*, *Mad8Form*, *Quad*, *QuadOld*, *RfCavity*, *SRotate*, *ST*, *Sextupole*, *Solenoid*, *Subs*, *Track*, *Tunes*, *TwissTran*, *Wire*, `^`, *alphax*, *alphaxT*, *alphay*, *alphayT*, *betax*, *betaxT*, *betay*, *betayT*, *cosmux*, *cosmuy*, *dET*, *dispersion*, *etapx*, *etapxT*, *etapy*, *etapyT*, *etax*, *etaxT*, *etay*, *etayT*, *gammax*, *gammay*, *nux*, *nuy*, *sinmux*, *sinmuy*, *twiss*, *twissx*, *twissy*] **(3.2)**

Element Details

In this section the individual machine elements provided by *Lattice* are described by example. An explicit listing of the tracking function is obtained by calling element:-TF(<*x,xp,y,yp,dl,dp*>) with unassigned names in the coordinate vector. Note that the name *l* is used for the length of the element and should not be used for the path-length difference in the particle vector.

Drift

drift := *Drift*(*len*)

$$Record\left(l = len,\ kind = Drift,\ dE = 0,\ Eref = 0,\ R = \begin{bmatrix} 1 & len & 0 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 & 0 & 0 \\ 0 & 0 & 1 & len & 0 & 0 \\ 0 & 0 & 0 & 1 & 0 & 0 \\ 0 & 0 & 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 0 & 0 & 1 \end{bmatrix},\ TF\right) \quad \textbf{(4.1.1)}$$

$$= trackFunction \Big)$$

drift:-*TF* (⟨ *x*, *xp*, *y*, *yp*, *dl*, *dp*⟩)

$$\begin{bmatrix} x + len\ xp \\ xp \\ y + len\ yp \\ yp \\ dl \\ dp \end{bmatrix} \quad \textbf{(4.1.2)}$$

len - length [m]
This element implements a field-free drift region. The track function presently is to first order only i. e. neglects the path-length change with angle.

Bend

dipole d *Bend* (*len*, theta, *n*, *E* [*ref*]);

$$Record\left(l = len,\ a = \theta,\ E1 = 0,\ E2 = 0,\ n = n,\ Eref = E_{ref},\ kind = Bend,\ \rho = \frac{len}{\theta},\ dE = \left(E \right.\right. \quad \textbf{(4.2.1)}$$

$$\rightarrow -\frac{177}{4000000}\ \frac{E^4\ ?\text{:-}a\ ?\text{:-}a}{?\text{:-}l\ \pi}\Big),\ R = \Big[\Big[\cos\left(\sqrt{1-n}\ \theta\right),$$

$$\frac{\sin\left(\sqrt{1-n}\,\theta\right) len}{\sqrt{1-n}\,\theta}, 0, 0, 0, \frac{len\left(1-\cos\left(\sqrt{1-n}\,\theta\right)\right)}{\theta\,(1-n)}\Bigg],$$

$$\left[-\frac{\sqrt{1-n}\,\theta\sin\left(\sqrt{1-n}\,\theta\right)}{len}, \cos\left(\sqrt{1-n}\,\theta\right), 0, 0, 0, \frac{\sin\left(\sqrt{1-n}\,\theta\right)}{\sqrt{1-n}}\right],$$

$$\left[0, 0, \cos\left(\sqrt{n}\,\theta\right), \frac{\sin\left(\sqrt{n}\,\theta\right) len}{\sqrt{n}\,\theta}, 0, 0\right],$$

$$\left[0, 0, -\frac{\sqrt{n}\,\theta\sin\left(\sqrt{n}\,\theta\right)}{len}, \cos\left(\sqrt{n}\,\theta\right), 0, 0\right],$$

$$\left[-\frac{\sin\left(\sqrt{1-n}\,\theta\right)}{\sqrt{1-n}}, -\frac{len\left(1-\cos\left(\sqrt{1-n}\,\theta\right)\right)}{\theta\,(1-n)}, 0, 0, 1,\right.$$

$$\left.\frac{len\left(\sqrt{1-n}\,\theta-\sin\left(\sqrt{1-n}\,\theta\right)\right)}{\theta\,(1-n)^{3/2}}\right],$$

$$\left.\left[0, 0, 0, 0, 0, 1\right]\right], TF = trackFunction\Bigg)$$

dipole:-TF ($\langle x, xp, y, yp, dl, dp\rangle$)

$$\left[\left[\cos\left(\sqrt{1-n}\,\theta\right) x + \frac{\sin\left(\sqrt{1-n}\,\theta\right) len\, xp}{\sqrt{1-n}\,\theta} + \frac{len\left(1-\cos\left(\sqrt{1-n}\,\theta\right)\right) dp}{\theta\,(1-n)}\right], \quad \textbf{(4.2.2)}$$

$$\left[-\frac{\sqrt{1-n}\,\theta\sin\left(\sqrt{1-n}\,\theta\right) x}{len} + \cos\left(\sqrt{1-n}\,\theta\right) xp + \frac{\sin\left(\sqrt{1-n}\,\theta\right) dp}{\sqrt{1-n}}\right],$$

$$\left[\cos\left(\sqrt{n}\,\theta\right) y + \frac{\sin\left(\sqrt{n}\,\theta\right) len\, yp}{\sqrt{n}\,\theta}\right],$$

$$\left[-\frac{\sqrt{n}\,\theta\sin\left(\sqrt{n}\,\theta\right) y}{len} + \cos\left(\sqrt{n}\,\theta\right) yp\right],$$

$$\left[-\frac{\sin\left(\sqrt{1-n}\,\theta\right) x}{\sqrt{1-n}} - \frac{len\left(1-\cos\left(\sqrt{1-n}\,\theta\right)\right) xp}{\theta\,(1-n)} + dl\right.$$

$$\left.+\frac{len\left(\sqrt{1-n}\,\theta-\sin\left(\sqrt{1-n}\,\theta\right)\right) dp}{\theta\,(1-n)^{3/2}}\right],$$

$$\left.\left[dp\right]\right]$$

len - the length [m]
theta - the bending angle [rad]
n - (optional) the field index, defaults to 0
E[ref] - (optional) the reference energy [MeV], defaults to 0

The **Bend** element implements a sector dipole magnet. The field index *n* allows to implement a gradient magnet. Edge focusing is taken into account.
The transfer function at present is first-order only and does not include and higher-order aberrations.

Quad

quadrupole d *Quad*(*len*, *k1l*);

$$Record\left(l = len,\ k1l = k1l,\ ns = 0,\ Eref = 0,\ dE = 0,\ kind = Quad,\ R = \left[\left[\cos\left(\sqrt{len}\ \sqrt{k1l}\right),\ \frac{\sin\left(\sqrt{len}\ \sqrt{k1l}\right)\sqrt{len}}{\sqrt{k1l}}, 0, 0, 0, 0\right],\ \left[-\frac{\sqrt{k1l}\ \sin\left(\sqrt{len}\ \sqrt{k1l}\right)}{\sqrt{len}}, \cos\left(\sqrt{len}\ \sqrt{k1l}\right), 0, 0, 0, 0\right],\ \left[0, 0, \cosh\left(\sqrt{len\ k1l}\right), \frac{\sinh\left(\sqrt{len\ k1l}\right)\sqrt{len}}{\sqrt{k1l}}, 0, 0\right],\ \left[0, 0, \frac{\sqrt{k1l}\ \sinh\left(\sqrt{len}\ \sqrt{k1l}\right)}{\sqrt{len}}, \cosh\left(\sqrt{len}\ \sqrt{k1l}\right), 0, 0\right],\ \left[0, 0, 0, 0, 1, 0\right],\ \left[0, 0, 0, 0, 0, 1\right]\right], TF = trackFunction\right) \quad \textbf{(4.3.1)}$$

quadrupole:-*TF*(⟨*x*, *xp*, *y*, *yp*, *dl*, *dp*⟩)

$$\begin{bmatrix} \cos\left(\sqrt{len}\ \sqrt{k1l}\right) x + \frac{\sin\left(\sqrt{len}\ \sqrt{k1l}\right)\sqrt{len}\ xp}{\sqrt{k1l}} \\ -\frac{\sqrt{k1l}\ \sin\left(\sqrt{len}\ \sqrt{k1l}\right) x}{\sqrt{len}} + \cos\left(\sqrt{len}\ \sqrt{k1l}\right) xp \\ \cosh\left(\sqrt{len\ k1l}\right) y + \frac{\sinh\left(\sqrt{len\ k1l}\right)\sqrt{len}\ yp}{\sqrt{k1l}} \\ \frac{\sqrt{k1l}\ \sinh\left(\sqrt{len}\ \sqrt{k1l}\right) y}{\sqrt{len}} + \cosh\left(\sqrt{len}\ \sqrt{k1l}\right) yp \\ dl \\ dp \end{bmatrix} \quad \textbf{(4.3.2)}$$

len - the length [m]
k1l - the integrated focusing strength [1/m]

The **Quad**rupole element implements a thick quadrupole. The focusing strength is the integrated strength *k1*len*. This allows to model a thin quadrupole by setting the length to 0 when calling **Quad**(). Modifying the length to 0 in a subsequent **Subs** operation, however, will lead to a divide-by-zero error and is to be avoided.
The trackFunction at present is to first order only and does not contain any chromatic or third-order aberrations. Chromaticity can be modelled by calling **Quad** with *k1l* being a function of *dp* (the 6th coordinate) of the particle.

Sextupole

sext d *Sextupole* (*len*, *k2l*)

$$Record\left(l = len,\ k2l = k2l,\ Eref = 0,\ kind = Sextupole,\ dE = 0,\ R = \begin{bmatrix} 1 & len & 0 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 & 0 & 0 \\ 0 & 0 & 1 & len & 0 & 0 \\ 0 & 0 & 0 & 1 & 0 & 0 \\ 0 & 0 & 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 0 & 0 & 1 \end{bmatrix},\ TF = trackFunction\right) \quad \textbf{(4.4.1)}$$

sext:-TF (⟨*x*, *xp*, *y*, *yp*, *l*, *dp*⟩)

$$\begin{bmatrix} x + \frac{1}{2}\,len\,xp + \frac{1}{2}\,len\left(xp + k2l\left(\left(y + \frac{1}{2}\,len\,yp\right)^2 - \left(x + \frac{1}{2}\,len\,xp\right)^2\right)\right) \\ xp + k2l\left(\left(y + \frac{1}{2}\,len\,yp\right)^2 - \left(x + \frac{1}{2}\,len\,xp\right)^2\right) \\ y + \frac{1}{2}\,len\,yp + \frac{1}{2}\,len\left(yp + 2\,k2l\left(x + \frac{1}{2}\,len\,xp\right)\left(y + \frac{1}{2}\,len\,yp\right)\right) \\ yp + 2\,k2l\left(x + \frac{1}{2}\,len\,xp\right)\left(y + \frac{1}{2}\,len\,yp\right) \\ l \\ dp \end{bmatrix} \quad \textbf{(4.4.2)}$$

len - the length [m]
k2l - the integrated sextupole strength [1/m^2]

The **Sextupole** element implements a thin sextupole with a pure quadratic field. The length is made

up by drift sections of half the element length up-and downstream of the thin sextupole element. The *R* matrix of the sextupole is that of a drift section.
The trackFunction implements a pure sextupolar field plus the first-order drift.
Sextupoles with more realistic behavior can be modelled using several sextupole-slices.

Solenoid

This element is not yet properly debugged. Use at your own risk!

sol := *Solenoid*(*len*, *ks*)

$$Record\left(l = len, a = a1, Eref = Er, kind = Solenoid, dE = (E \to 0), R = \left[\left[\cos(ks\ len)^2, \frac{\sin(ks\ len)\cos(ks\ len)}{ks}, \sin(ks\ len)\cos(ks\ len), \frac{\sin(ks\ len)^2}{ks}, 0, 0\right], \left[-ks\sin(ks\ len)\cos(ks\ len), \cos(ks\ len)^2, -ks\sin(ks\ len)^2, \sin(ks\ len)\cos(ks\ len), 0, 0\right], \left[-\sin(ks\ len)\cos(ks\ len), -\frac{\sin(ks\ len)^2}{ks}, \cos(ks\ len)^2, \frac{\sin(ks\ len)\cos(ks\ len)}{ks}, 0, 0\right], \left[ks\sin(ks\ len)^2, -\sin(ks\ len)\cos(ks\ len), -ks\sin(ks\ len)\cos(ks\ len), \cos(ks\ len)^2, 0, 0\right], \left[0, 0, 0, 0, 1, 0\right], \left[0, 0, 0, 0, 0, 1\right]\right], TF = trackFunction\right) \quad \textbf{(4.5.1)}$$

sol:-*TF*(⟨*x*, *xp*, *y*, *yp*, *l*, *dp*⟩)

$$\left[\left[\cos(ks\ len)^2 x + \frac{\sin(ks\ len)\cos(ks\ len)\ xp}{ks} + \sin(ks\ len)\cos(ks\ len)\ y + \frac{\sin(ks\ len)^2\ yp}{ks}\right], \left[-ks\sin(ks\ len)\cos(ks\ len)\ x + \cos(ks\ len)^2\ xp - ks\sin(ks\ len)^2\ y + \sin(ks\ len)\cos(ks\ len)\ yp\right], \left[-\sin(ks\ len)\cos(ks\ len)\ x - \frac{\sin(ks\ len)^2\ xp}{ks} + \cos(ks\ len)^2\ y\right] \quad \textbf{(4.5.2)}$$

$$+\frac{\sin(ks\ len)\cos(ks\ len)\ yp}{ks}\Bigg],$$

$$[ks\sin(ks\ len)^2\ x - \sin(ks\ len)\cos(ks\ len)\ xp - ks\sin(ks\ len)\cos(ks\ len)\ y$$

$$+\cos(ks\ len)^2\ yp],$$

$$[l],$$

$$[dp]]$$

len - the length [m]
ks - the solenoid strength [1/m]

The **Solenoid** element implements a solenoid without fringe fields. The strength *ks* is not integrated. The trackFunction implements the first-order equations only.

RfCavity

This element is not yet properly debugged. Use at your own risk!

cav d *RfCavity*(*len*, *freq*, *V0*, phi)

$$Record\left(l = len,\ kind = RfCavity,\ R = \begin{bmatrix} 1 & len & 0 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 & 0 & 0 \\ 0 & 0 & 1 & len & 0 & 0 \\ 0 & 0 & 0 & 1 & 0 & 0 \\ 0 & 0 & 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 0 & 0 & 1 \end{bmatrix},\ TF = trackFunction,\ Eref,\ dE\right) \quad \textbf{(4.6.1)}$$

$$= (E \rightarrow Vcav\ sphi)\Bigg)$$

cav:-*TF* (⟨*x*, *xp*, *y*, *yp*, *l*, *dp*⟩)

$$\begin{bmatrix} x + len\ xp \\ xp \\ y + len\ yp \\ yp \\ l \\ dp + \dfrac{V0 \sin\left(\phi + \dfrac{1}{299792458}\ \dfrac{l\ freq\ \pi}{BeamBetar}\right)}{BeamEnergy} \end{bmatrix} \qquad \textbf{(4.6.2)}$$

len - the length [m]
freq - the rf frequency [Hz]
V0 - the peak rf voltage [MV]
phi - the synchronous angle [rad]

The **RfCavity** element implements a single-cell rf cavity. The *R* matrix is that of the equivalent drift section.
The trackFunction adds the longitudinal energy change to the 6th coordinate of the particle. No change in path length due to the kick is calculated.

GKick

corr := *GKick*(*dx*, *dxp*, *dy*, *dyp*, *dl*, *ddp*)

$$Record\left(l = 0,\ kind = GKick,\ R = \begin{bmatrix} 1 & 0 & 0 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 & 0 & 0 \\ 0 & 0 & 1 & 0 & 0 & 0 \\ 0 & 0 & 0 & 1 & 0 & 0 \\ 0 & 0 & 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 0 & 0 & 1 \end{bmatrix},\ dx = dx,\ dxp = dxp,\ dy = dy,\ dyp = dyp,\ dl = dl,\ ddp = ddp,\ dE = 0,\ Eref = 0,\ TF = trackFunction\right) \qquad \textbf{(4.7.1)}$$

corr:-*TF*(⟨*x*, *xp*, *y*, *yp*, *l*, *dp*⟩)

$$\begin{bmatrix} x + dx \\ xp + dxp \\ y + dy \\ yp + dyp \\ l + dl \\ dp + ddp \end{bmatrix} \tag{4.7.2}$$

dx - horizontal offset [m]
dxp - horizontal angle kick [rad]
dy - vertical offset [m]
dyp - vertical angular kick [rad]
dl - longitudinal offset [m]
ddp - energy kick [1]

The **GKick** element implements a general kick. It can be used to model orbit correctors but also misalignments. No energy dependence of the kick is present. The element has zero length and the *R* Matrix is the *IdentityMatrix*.
The trackFunction adds the specified deflections and offsets to the particle coordinates.

Wire

w d *Wire*(*len*, *i*, *xs*, *ys*, *Eref*)

$$\begin{aligned} &Record\left(l = len,\ k1 = -\frac{0.149896271779928\ mu\_0\ i\ \left(xs^2 - ys^2\right)}{\pi\ \left(xs^2 + ys^2\right)^2\ Eref},\ kind = Wire,\ R = \left[\left[1.\right.\right.\right. \\ &\quad + \frac{0.0749481358899640\ mu\_0\ i\ \left(xs^2 - ys^2\right)\ len^2}{\pi\ \left(xs^2 + ys^2\right)^2\ Eref},\ \frac{1}{2}\ len \\ &\quad + \frac{1}{2}\ len\left(\frac{0.0749481358899640\ mu\_0\ i\ \left(xs^2 - ys^2\right)\ len^2}{\pi\ \left(xs^2 + ys^2\right)^2\ Eref} + 1\right),\ 0.,\ 0.,\ 0.,\ 0.\Big], \\ &\quad \left[\frac{0.149896271779928\ mu\_0\ i\ \left(xs^2 - ys^2\right)\ len}{\pi\ \left(xs^2 + ys^2\right)^2\ Eref},\ 1.\right. \\ &\quad + \frac{0.0749481358899640\ mu\_0\ i\ \left(xs^2 - ys^2\right)\ len^2}{\pi\ \left(xs^2 + ys^2\right)^2\ Eref},\ 0.,\ 0.,\ 0.,\ 0.\Big], \\ &\quad \left[0.,\ 0.,\ 1. - \frac{0.0749481358899640\ mu\_0\ i\ \left(xs^2 - ys^2\right)\ len^2}{\pi\ \left(xs^2 + ys^2\right)^2\ Eref},\ \frac{1}{2}\ len + \frac{1}{2}\ len\left(1\right.\right. \\ &\quad \left.\left. - \frac{0.0749481358899640\ mu\_0\ i\ \left(xs^2 - ys^2\right)\ len^2}{\pi\ \left(xs^2 + ys^2\right)^2\ Eref}\right),\ 0.,\ 0.\Big], \\ &\quad \left[0.,\ 0.,\ -\frac{0.149896271779928\ mu\_0\ i\ \left(xs^2 - ys^2\right)\ len}{\pi\ \left(xs^2 + ys^2\right)^2\ Eref},\ 1.\right. \end{aligned} \tag{4.8.1}$$

$$-\frac{0.0749481358899640\ mu\_0\ i\ \left(xs^2-ys^2\right)\ len^2}{\pi\ \left(xs^2+ys^2\right)^2\ Eref},\ 0.,\ 0.\Bigg],$$

$$\left[0.,\ 0.,\ 0.,\ 0.,\ 1.,\ 0.\right],$$

$$\left[0.,\ 0.,\ 0.,\ 0.,\ 0.,\ 1.\right]\Bigg],\ Eref = Eref,\ TF = trackFunction\Bigg)$$

w:-$TF\left(\langle x, xp, y, yp, l, dp\rangle\right)$

$$\left[\left[\left[x\right],\right.\right.$$

(4.8.2)

$$\left[xp+\frac{1}{Eref}\left(0.299792543559857\left(-\frac{1}{2}\ \frac{xs\ mu\_0\ i}{\pi\ \left(xs^2+ys^2\right)}\right.\right.\right.$$

$$+\frac{1}{2}\ \frac{\left(\frac{mu\_0\ i}{\pi}-\frac{2\ xs^2\ mu\_0\ i}{\pi\ \left(xs^2+ys^2\right)}\right)x}{xs^2+ys^2}-\frac{xs\ mu\_0\ i\ ys\ y}{\pi\ \left(xs^2+ys^2\right)^2}$$

$$+\frac{1}{2}\ \frac{\left(\frac{xs\ mu\_0\ i}{\pi\ \left(xs^2+ys^2\right)}-\frac{2\ mu\_0\ i\ \left(xs^2-ys^2\right)\ xs}{\pi\ \left(xs^2+ys^2\right)^2}\right)x^2}{xs^2+ys^2}$$

$$+\frac{1}{2}\ \frac{\left(-\frac{4\ ys\ mu\_0\ i\ xs^2}{\pi\ \left(xs^2+ys^2\right)^2}-\frac{2\ mu\_0\ i\ \left(xs^2-ys^2\right)\ ys}{\pi\ \left(xs^2+ys^2\right)^2}\right)y\ x}{xs^2+ys^2}$$

$$+\frac{1}{2}\ \frac{\left(\frac{xs\ mu\_0\ i}{\pi\ \left(xs^2+ys^2\right)}-\frac{4\ ys^2\ mu\_0\ i\ xs}{\pi\ \left(xs^2+ys^2\right)^2}\right)y^2}{xs^2+ys^2}$$

$$+\frac{1}{2}\ \frac{\left(\frac{mu\_0\ i\ \left(xs^2-ys^2\right)}{\pi\ \left(xs^2+ys^2\right)^2}-\frac{2\ mu\_0\ i\ \left(xs^3-3\ ys^2\ xs\right)\ xs}{\pi\ \left(xs^2+ys^2\right)^3}\right)x^3}{xs^2+ys^2}$$

$$+\frac{1}{2}\ \frac{1}{xs^2+ys^2}\left(\left(\frac{2\ ys\ mu\_0\ i\ xs}{\pi\ \left(xs^2+ys^2\right)^2}-\frac{2\ mu\_0\ i\ \left(6\ ys\ xs^2-2\ ys^3\right)\ xs}{\pi\ \left(xs^2+ys^2\right)^3}\right.\right.$$

$$- \frac{2\, mu\_0\, i \left(xs^3 - 3\, ys^2\, xs\right) ys}{\pi \left(xs^2 + ys^2\right)^3} \Bigg) y\, x^2 \Bigg)$$

$$+ \frac{1}{2} \frac{1}{xs^2 + ys^2} \Bigg(\Bigg(\frac{mu\_0\, i \left(xs^2 - ys^2\right)}{\pi \left(xs^2 + ys^2\right)^2} - \frac{2\, mu\_0\, i \left(-xs^3 + 3\, ys^2\, xs\right) xs}{\pi \left(xs^2 + ys^2\right)^3}$$

$$- \frac{2\, mu\_0\, i \left(6\, ys\, xs^2 - 2\, ys^3\right) ys}{\pi \left(xs^2 + ys^2\right)^3} \Bigg) y^2\, x \Bigg)$$

$$+ \frac{1}{2} \frac{\left(\frac{2\, ys\, mu\_0\, i\, xs}{\pi \left(xs^2 + ys^2\right)^2} - \frac{2\, mu\_0\, i \left(-xs^3 + 3\, ys^2\, xs\right) ys}{\pi \left(xs^2 + ys^2\right)^3} \right) y^3}{xs^2 + ys^2} \Bigg) len \Bigg) \Bigg],$$

$$\Bigg[y \Bigg],$$

$$\Bigg[yp + \frac{1}{Eref} \Bigg(0.299792543559857 \Bigg(-\frac{1}{2} \frac{ys\, mu\_0\, i}{\pi \left(xs^2 + ys^2\right)} - \frac{ys\, mu\_0\, i\, xs\, x}{\pi \left(xs^2 + ys^2\right)^2}$$

$$+ \frac{1}{2} \frac{\left(\frac{mu\_0\, i}{\pi} - \frac{2\, ys^2\, mu\_0\, i}{\pi \left(xs^2 + ys^2\right)} \right) y}{xs^2 + ys^2}$$

$$+ \frac{1}{2} \frac{\left(\frac{ys\, mu\_0\, i}{\pi \left(xs^2 + ys^2\right)} - \frac{4\, ys\, mu\_0\, i\, xs^2}{\pi \left(xs^2 + ys^2\right)^2} \right) x^2}{xs^2 + ys^2}$$

$$+ \frac{1}{2} \frac{\left(-\frac{2\, mu\_0\, i \left(-xs^2 + ys^2\right) xs}{\pi \left(xs^2 + ys^2\right)^2} - \frac{4\, ys^2\, mu\_0\, i\, xs}{\pi \left(xs^2 + ys^2\right)^2} \right) y\, x}{xs^2 + ys^2}$$

$$+ \frac{1}{2} \frac{\left(\frac{ys\, mu\_0\, i}{\pi \left(xs^2 + ys^2\right)} - \frac{2\, mu\_0\, i \left(-xs^2 + ys^2\right) ys}{\pi \left(xs^2 + ys^2\right)^2} \right) y^2}{xs^2 + ys^2}$$

$$+\frac{1}{2}\,\frac{\left(\frac{2\,ys\,mu\_0\,i\,xs}{\pi\left(xs^2+ys^2\right)^2}-\frac{2\,mu\_0\,i\left(3\,ys\,xs^2-ys^3\right)xs}{\pi\left(xs^2+ys^2\right)^3}\right)x^3}{xs^2+ys^2}$$

$$+\frac{1}{2}\,\frac{1}{xs^2+ys^2}\left(\left(\frac{mu\_0\,i\left(-xs^2+ys^2\right)}{\pi\left(xs^2+ys^2\right)^2}-\frac{2\,mu\_0\,i\left(-2\,xs^3+6\,ys^2\,xs\right)xs}{\pi\left(xs^2+ys^2\right)^3}\right.\right.$$

$$\left.-\frac{2\,mu\_0\,i\left(3\,ys\,xs^2-ys^3\right)ys}{\pi\left(xs^2+ys^2\right)^3}\right)y\,x^2\Bigg)+\frac{1}{2}\,\frac{1}{xs^2+ys^2}\left(\left(\frac{2\,ys\,mu\_0\,i\,xs}{\pi\left(xs^2+ys^2\right)^2}\right.\right.$$

$$\left.-\frac{2\,mu\_0\,i\left(-3\,ys\,xs^2+ys^3\right)xs}{\pi\left(xs^2+ys^2\right)^3}-\frac{2\,mu\_0\,i\left(-2\,xs^3+6\,ys^2\,xs\right)ys}{\pi\left(xs^2+ys^2\right)^3}\right)y^2\,x\Bigg)$$

$$+\frac{1}{2}\,\frac{\left(\frac{mu\_0\,i\left(-xs^2+ys^2\right)}{\pi\left(xs^2+ys^2\right)^2}-\frac{2\,mu\_0\,i\left(-3\,ys\,xs^2+ys^3\right)ys}{\pi\left(xs^2+ys^2\right)^3}\right)y^3}{xs^2+ys^2}\,len\Bigg)\Bigg],$$

$$\begin{bmatrix} \\ l \end{bmatrix},$$

$$\begin{bmatrix} \\ dp \end{bmatrix}\Bigg]$$

len - the length of the wire [m]
i - the excitation current [A]
xs - the horizontal distance to the central orbit [m]
ys - the vertical distance of the wire to the central orbit [m]
Eref - the beam energy [GeV]. Note: the energy has to be given for the kick to be computable.

The **Wire** element implements a current-carrying wire parallel to the beam orbit. Primary use is for beam-beam compensation. The model used is a 1/*r* drop in field of the wire (i.e. the wire is long compared to the distance from the beam). No end fields are considered. The *R* Matrix models the first-order kick (gradient) on the beam axis due to the wire.
The trackFunction implements the full nonlinear kick due to the wire, but without end effects.

Foil

Foils only work with numerical input data, so the following produces an error when trying to access the trackFunction:

window d *Foil*(*len*, *X0*, *Eref*)

$$\text{"Rms scattering angle is ",}\ \frac{13.6\sqrt{\frac{len}{X0}}\left(1+0.038\ln\left(\frac{len}{X0}\right)\right)}{Eref}$$

$$Record\left(l=len, kind=Foil, R=\begin{bmatrix}1&0&0&0&0&0\\0&1&0&0&0&0\\0&0&1&0&0&0\\0&0&0&1&0&0\\0&0&0&0&1&0\\0&0&0&0&0&1\end{bmatrix}, Eref=Eref, dE=0, TF\right) \quad \textbf{(4.9.1)}$$

$$=trackFunction\Bigg)$$

window:-*TF*(⟨*x*, *xp*, *y*, *yp*, *ldp*⟩)

```
Error, (in Statistics:-Sample) unable to evaluate 13.6*
(len/X0)^(1/2)*(1+0.38e-1*ln(len/X0))/Eref to floating-point
```

len - length of the foil [m]
X0 - radiation length of the material [m]
Eref - reference energy [MeV]. Used for the scattering calculation

The **Foil** element implements a scattering foil, e.g. a window in a beam line. As such items are invariably thin, the *R* Matrix is the unit matrix independent of length. Note that there is no energy loss calculated in the foil.
The trackFunction implements a Gaussian random-number generator and assigns angular kicks in both direction to the particle coordinates. The rms scattering angle is calculated using the PDG formula[5] and is printed when the **Foil** constructor is called.

Examples

Since the various functions are documented in the Help facility, this guide will introduce the use of *Lattice* by means of an example.
We reload *Lattice* as usual in Maple using with:
restart;

$with(Lattice):$

"Maple Initialization loaded..."

"Lattice.mw, Version 1.0.2, 10-Feb-2016" **(5.1)**

Example 1: Thin-lens FODO Lattice

Build a simple thin-lens FODO lattice and derive some formulae for its parameters.
The elements are defined thus:

$QFh := Quad\left(0, \frac{kf}{2}\right):$

$QDh := Quad\left(0, \frac{kd}{2}\right):$

$DRh := Drift\left(\frac{lcell}{4}\right):$

Build the beam line from these elements. This can be done in stages:

$FOD := DefineLine(QFh, DRh, DRh, QDh):$

$DOF := DefineLine(QDh, DRh, DRh, QFh):$

$FODO := DefineLine(FOD, DOF):$

Note that *DefineLine* works in a left-to-right fashion, like a beam-line definition in MAD, unlike matrix multiplication.

Find the cosine of the phase advance μ as function of the quadrupole strengths *kf* and *kd*:

$\cos(mux) = simplify(cosmux(FODO));$

$$\cos(mux) = -\frac{1}{2} kd\ lcell + \frac{1}{8} kd\ lcell^2\ kf + 1 - \frac{1}{2} lcell\ kf \quad \textbf{(5.1.1)}$$

$\cos(muy) = simplify(cosmuy(FODO))$

$$\cos(muy) = 1 + \frac{1}{2} lcell\ kf + \frac{1}{2} kd\ lcell + \frac{1}{8} kd\ lcell^2\ kf \quad \textbf{(5.1.2)}$$

which can be solved for the quad strengths in terms of the cosines in order to get the FODO cell parameters independent of the quadrupole strengths.

$convert\sim(solve([\textbf{(5.1.1)}, \textbf{(5.1.2)}], [kf, kd]), radical)$

$$\left[\left[kf = \frac{1}{lcell}\left(\frac{1}{2}\cos(muy) - \frac{1}{2}\cos(mux)\right.\right.\right. \quad \textbf{(5.1.3)}$$

$$\left. + \frac{1}{2}\left(\cos(muy)^2 - 2\cos(muy)\cos(mux) + \cos(mux)^2 - 16\cos(muy)\right.\right.$$

$$\left.\left. + 32 - 16\cos(mux)\right)^{1/2}\right), kd = \left(4\left(\frac{3}{2}\cos(mux) - 2 + \frac{1}{2}\cos(muy)\right.\right.$$

$$+\frac{1}{2}\left(\cos(muy)^2-2\cos(muy)\cos(mux)+\cos(mux)^2-16\cos(muy)\right.$$
$$\left.+32-16\cos(mux)\right)^{1/2}\Big)\Big)\Big/\Big(lcell\left(-4+\frac{1}{2}\cos(muy)-\frac{1}{2}\cos(mux)\right.$$
$$+\frac{1}{2}\left(\cos(muy)^2-2\cos(muy)\cos(mux)+\cos(mux)^2-16\cos(muy)\right.$$
$$\left.\left.+32-16\cos(mux)\right)^{1/2}\right)\Big)\Big]\Big]$$

Make the two phase advances equal for further simplification:

subs(*mux* = μ, *muy* = μ, **(5.1.3)**)[ ]

$$\left[kf=\frac{1}{2}\,\frac{\sqrt{-32\cos(\mu)+32}}{lcell},\ kd=\frac{4\left(2\cos(\mu)-2+\frac{1}{2}\sqrt{-32\cos(\mu)+32}\right)}{lcell\left(-4+\frac{1}{2}\sqrt{-32\cos(\mu)+32}\right)}\right] \quad \textbf{(5.1.4)}$$

and put this back into the cell:

cell := *Subs*(**(5.1.4)**[1], **(5.1.4)**[2], *FODO*):

#` `Note: *Subs* does not accept a list of replacement equations.

Compute the lattice functions for this cell:

tw := *twiss*(*cell*):

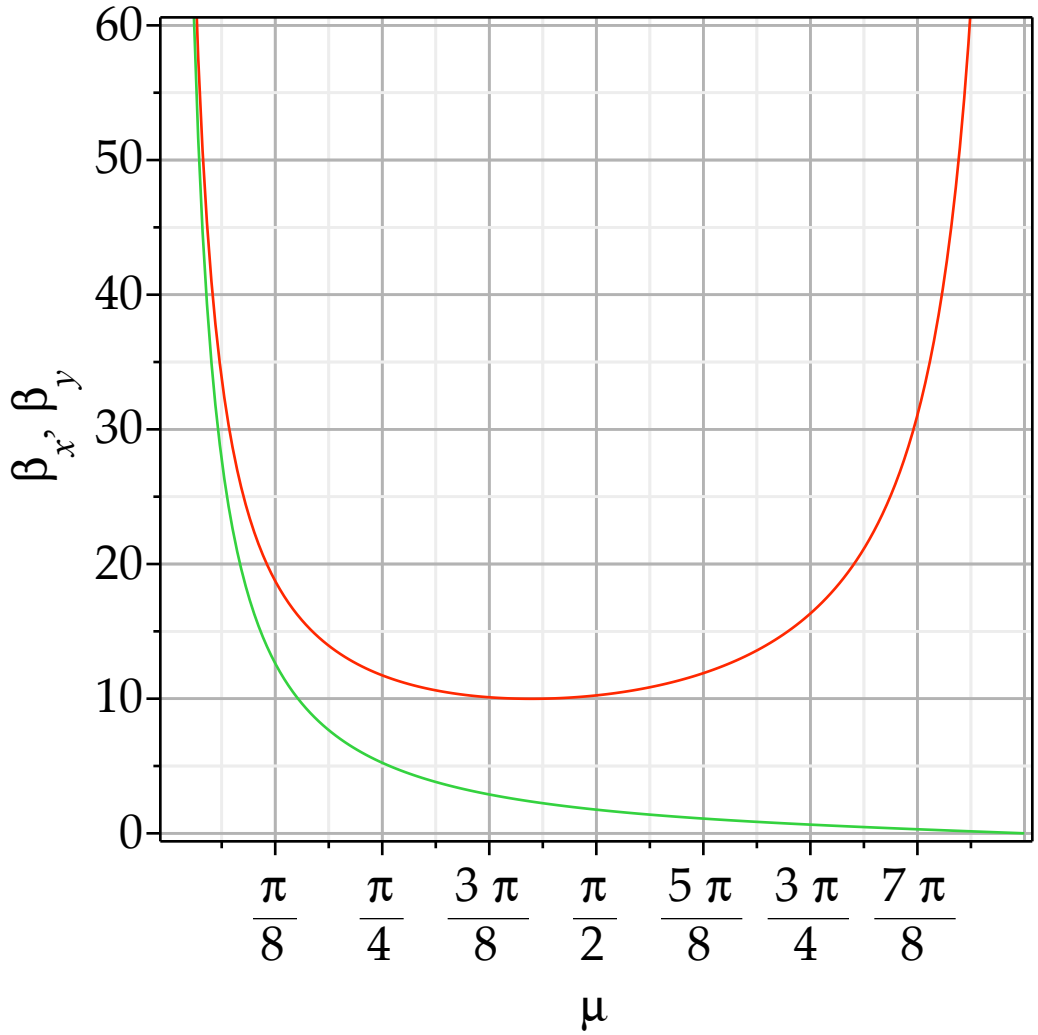
(5.1.5)

simplify~(*tw*) assuming 0 < mu, mu < Pi;

$$\left[\begin{array}{c}
-\dfrac{lcell\left(\cos(\mu)+1\right)}{\left(-2+\sqrt{-2\cos(\mu)+2}\right)\sin(\mu)} \\
0 \\
\dfrac{-2\cos(\mu)+\sqrt{-2\cos(\mu)+2}\,\cos(\mu)+2-\sqrt{-2\cos(\mu)+2}}{lcell\sin(\mu)} \\
\dfrac{lcell\left(\cos(\mu)-3+2\sqrt{-2\cos(\mu)+2}\right)}{\left(-2+\sqrt{-2\cos(\mu)+2}\right)\sin(\mu)} \\
0 \\
-\dfrac{2\cos(\mu)-2+\sqrt{-2\cos(\mu)+2}\,\cos(\mu)-\sqrt{-2\cos(\mu)+2}}{\sin(\mu)\,lcell}
\end{array}\right] \quad \textbf{(5.1.6)}$$

Plot this for a cell length of 6 m

plot(*subs*(*lcell* = 6, [**(5.1.6)**[1], **(5.1.6)**[4]]), mu = 0 ..Pi, *view* = [*default*, 0 ..60], *labels* = ['mu', *typeset*('beta[*x*]', '', '','beta[*y*]')]);



To plot the lattice functions specify values for the variables:

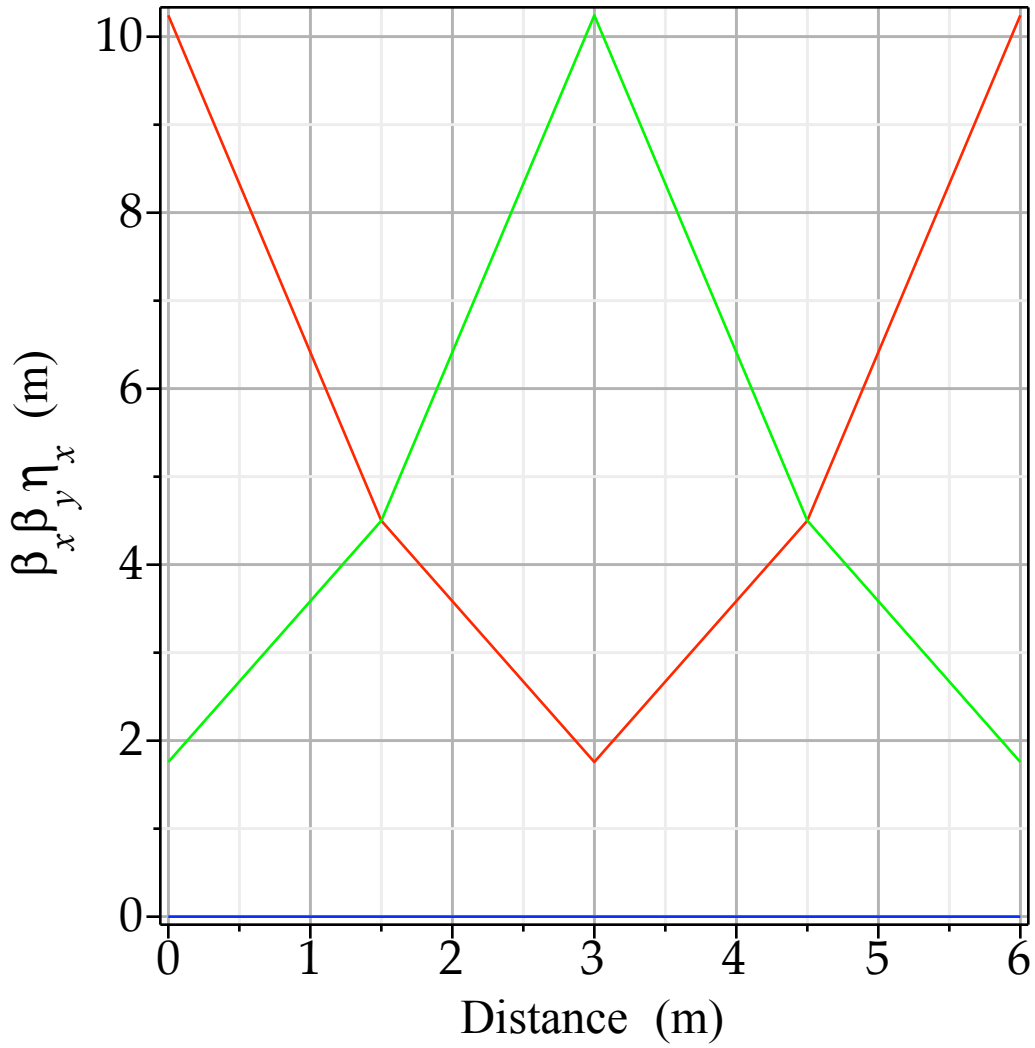
cell_numeric := *Subs*$\left(lcell = 6, \mathrm{mu} = \frac{\mathrm{Pi}}{2}, cell \right)$:

twiss_numeric := *evalf*~(*twiss*(*cell_numeric*));

$$\begin{bmatrix} 10.2426406871193 \\ 1.00000000000000\ 10^{-15} \\ 0.0976310729378179 \\ 1.75735931288072 \\ -1.50000000000000\ 10^{-15} \\ 0.569035593728850 \end{bmatrix} \quad \textbf{(5.1.7)}$$

latticePlot := *LatticePlot*(*cell_numeric*, *twiss_numeric*, ⟨0, 0, 0, 0⟩) :

plots:-*display*(*latticePlot*, *labels* = [*typeset*("Distance (m)"), *typeset*(beta[*x*], beta[*y*], eta[*x*], " (m)")])



A phase-space plot can be created by defining a beam and using *BeamPlot*. To make the plot more interesting, create some particles. Note that this does not involve actual tracking.

Beam := *DefineBeam*(*Electron*, 1, 1E−6, 1E−6, *twiss_numeric*, 0, 0, 0, 1000):

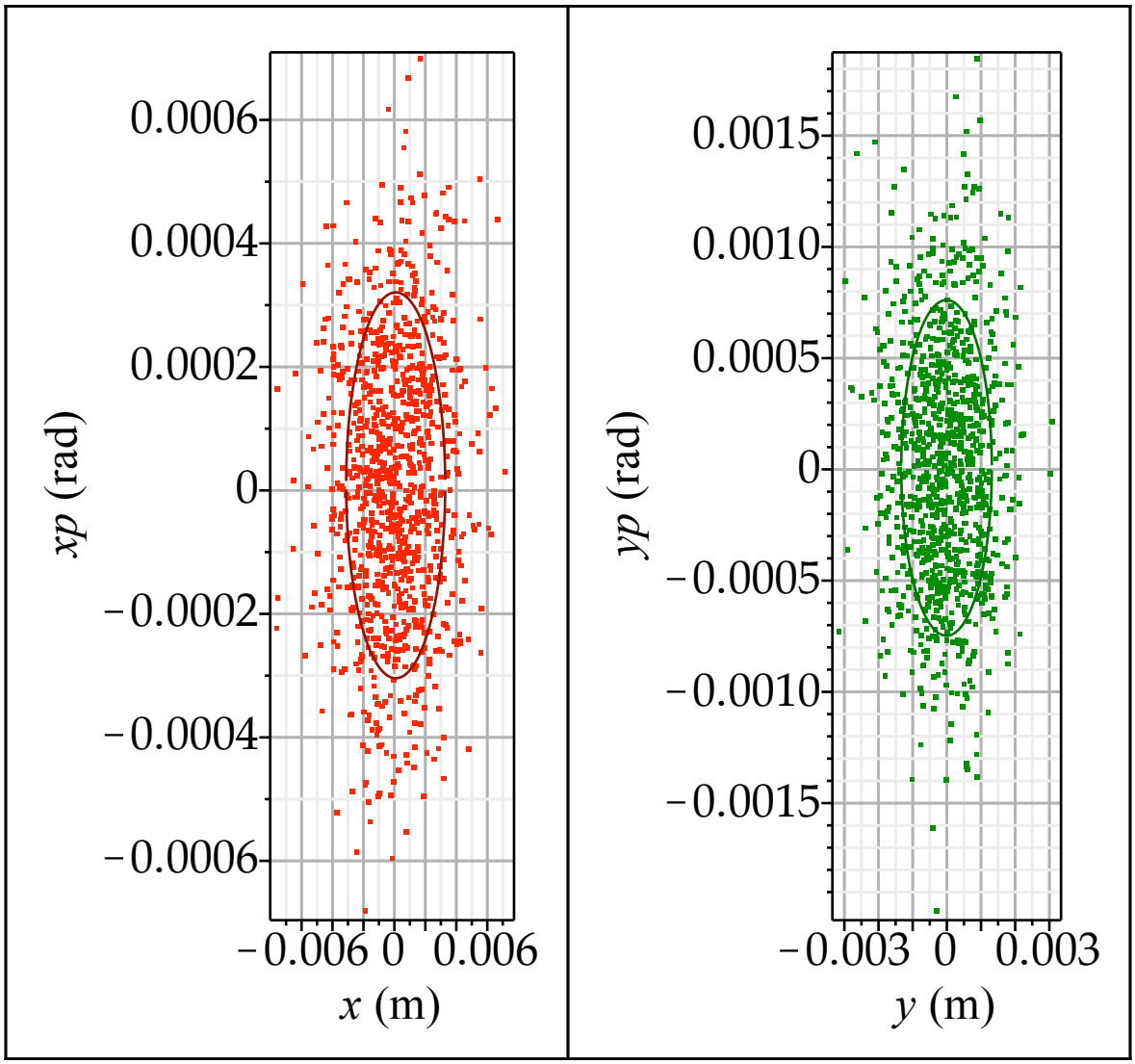
Beam:-Sigma

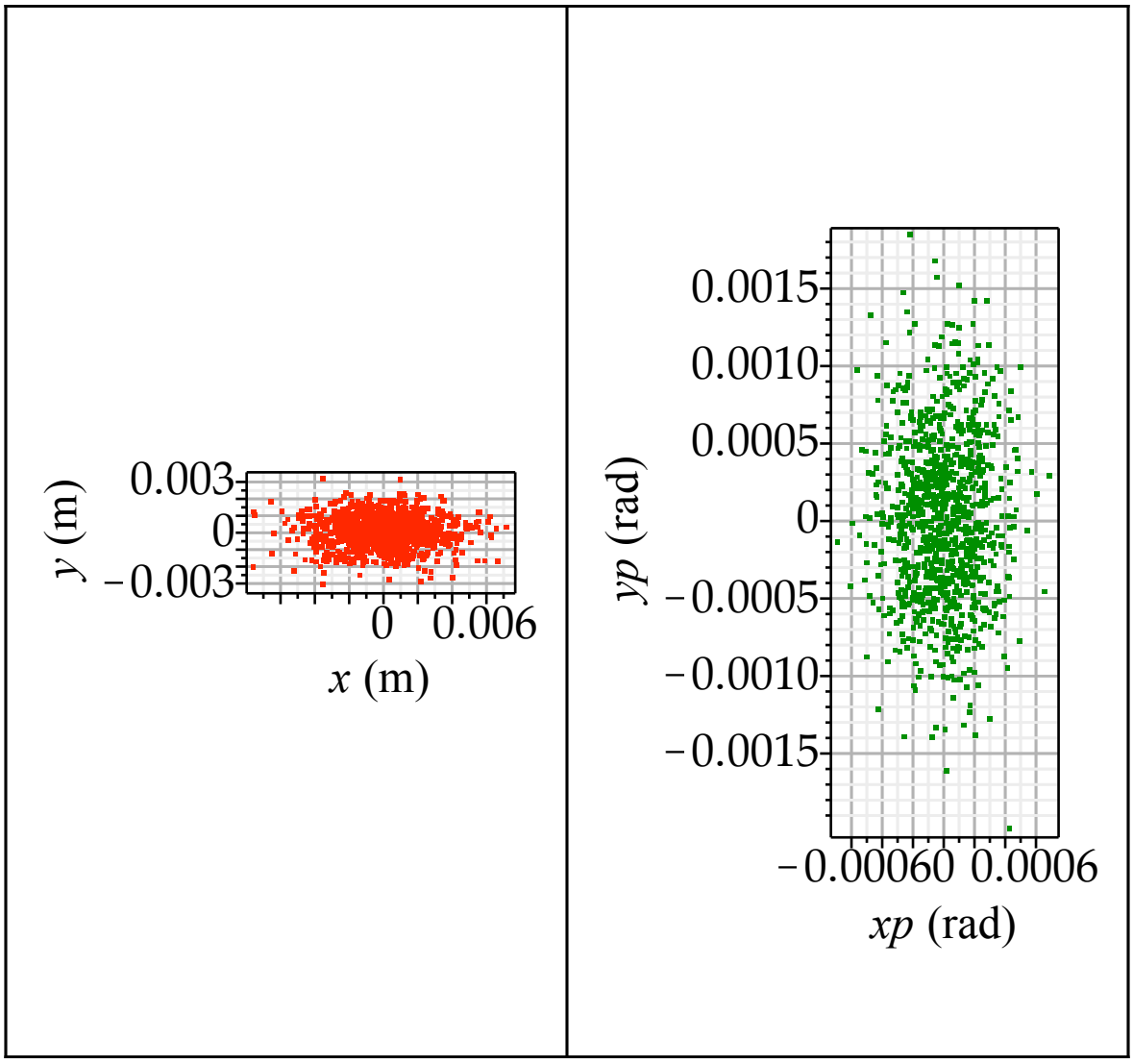
$$\begin{bmatrix} [0.0000102426406871193, -1.00000000000000\ 10^{-21}, 0, 0, 0, 0], \\ [-1.00000000000000\ 10^{-21}, 9.76310729378179\ 10^{-8}, 0, 0, 0, 0], \\ [0, 0, 0.00000175735931288072, 1.50000000000000\ 10^{-21}, 0, 0], \\ [0, 0, 1.50000000000000\ 10^{-21}, 5.69035593728850\ 10^{-7}, 0, 0], \\ [0, 0, 0, 0, 1, 0], \\ [0, 0, 0, 0, 0, 0] \end{bmatrix} \quad \textbf{(5.1.8)}$$

BeamPlot(*Beam*)

$$\begin{bmatrix} PLOT(\ldots) & PLOT(\ldots) \\ PLOT(\ldots) & PLOT(\ldots) \end{bmatrix} \quad \textbf{(5.1.9)}$$

plots:-*display*(%)


0.0006
0.0004
0.0002
0
−0.0002
−0.0004
−0.0006
xp (rad)
−0.006 0 0.006
x (m)
0.0015
0.0010
0.0005
0
−0.0005
−0.0010
−0.0015
yp (rad)
−0.003 0 0.003
y (m)



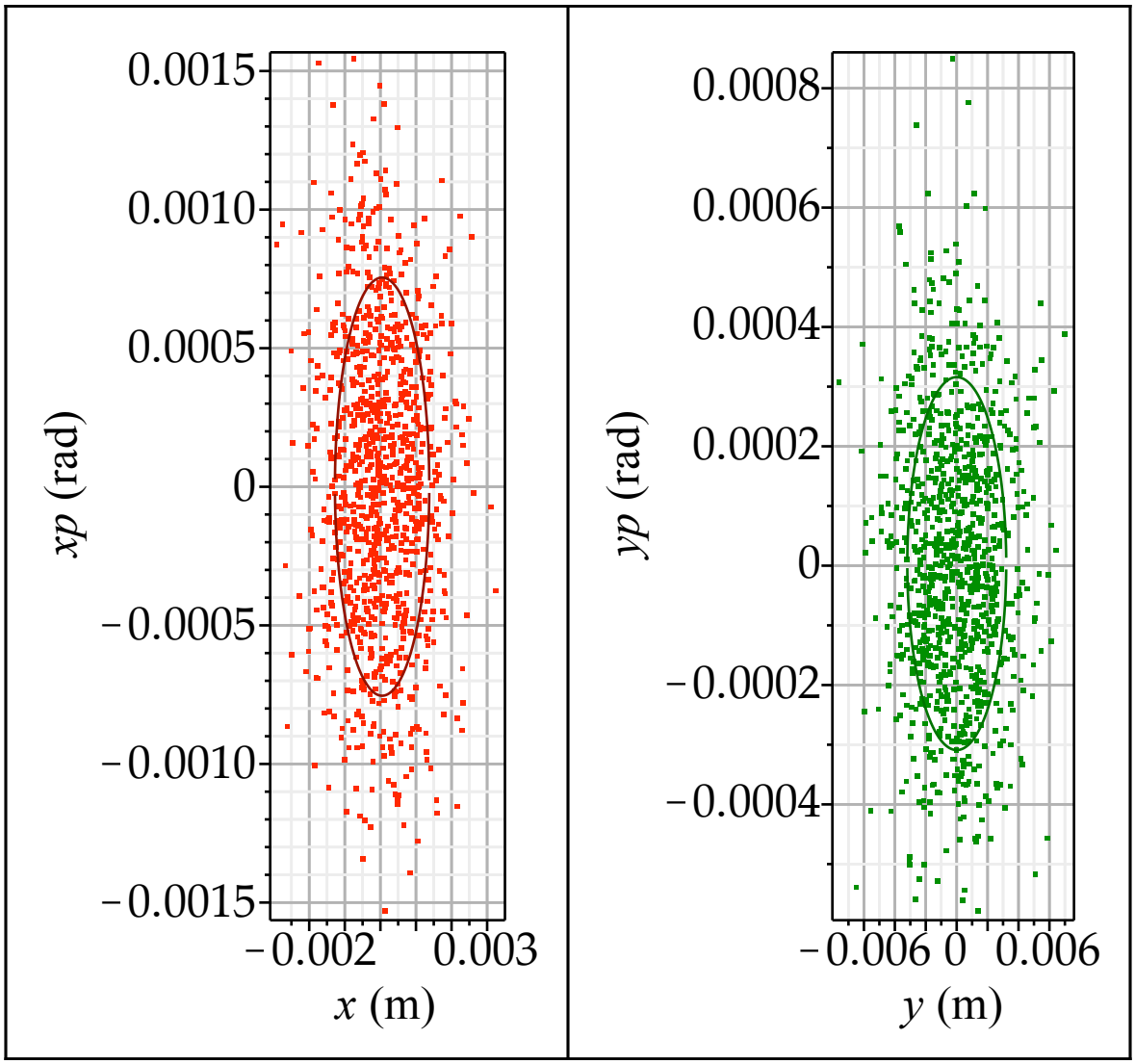
Note that in this version of *Lattice* the bottom two plots are only produced when particles exist in the beam.

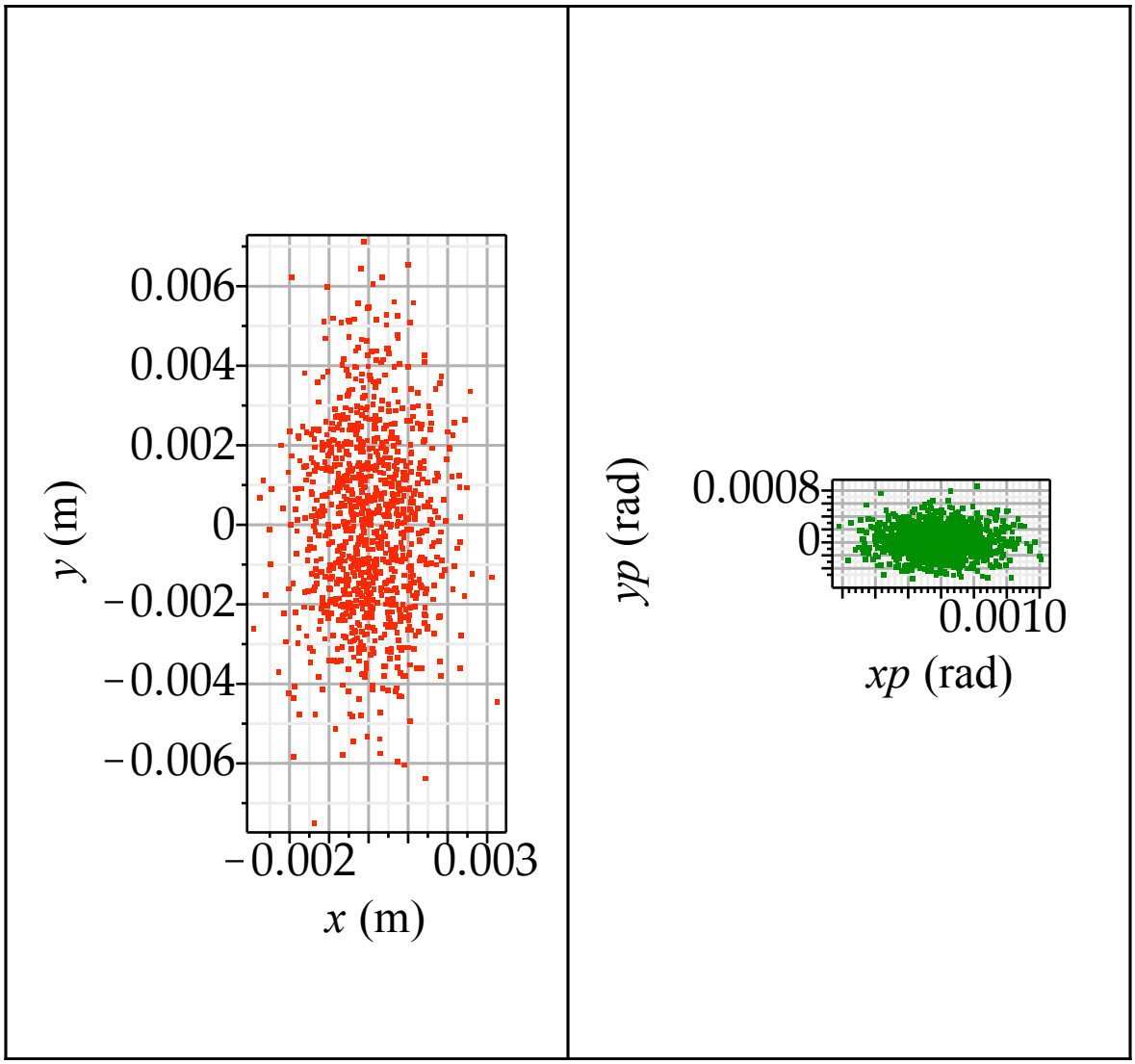
In order to see the phase space at the center of the cell, track the beam through the half-cell (FOD):

half_cell d *Subs*$\left(lcell = 6, \text{mu} = \frac{\text{Pi}}{2}, Subs(\textbf{(5.1.4)}[1], \textbf{(5.1.4)}[2], FOD) \right)$:

Beam_D d *Track*(*half_cell*, *Beam*) :

plots:-*display*(*BeamPlot*(*Beam_D*[2]));


0.0015
0.0010
0.0005
0
−0.0005
−0.0010
−0.0015
xp (rad)
−0.002
0.003
x (m)
0.0008
0.0006
0.0004
0.0002
0
−0.0002
−0.0004
yp (rad)
−0.006
0
0.006
y (m)



The result is as expected.

Tracks can be plotted by converting the line to an *ExpandedLine* and tracking that.

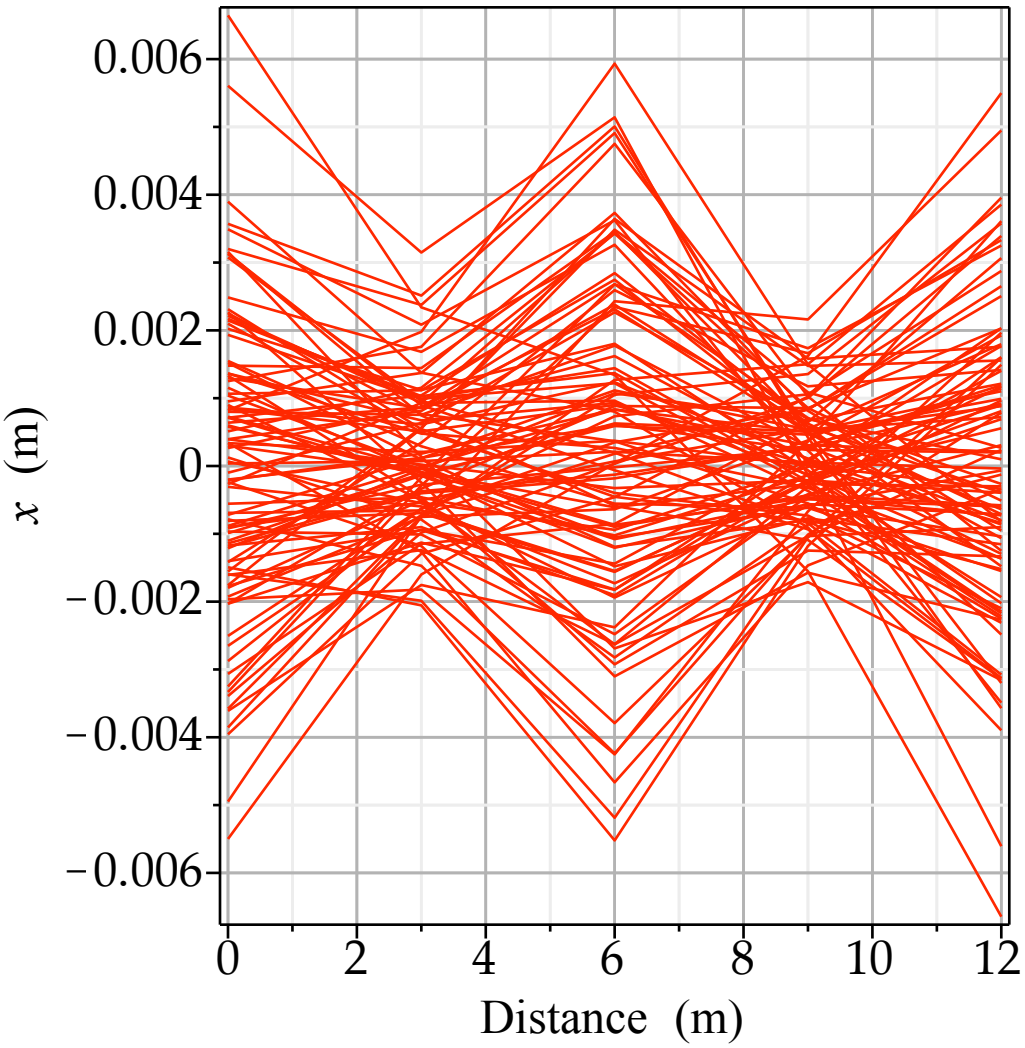
cell_e := *ExpandLine*(*cell_numeric*, *cell_numeric*) : # use 2 cells to make the plot more interesting

(5.1.10)

BeamVector := *Track*(*cell_e*, *Beam*) : #`here Track will create a Vector with *Beam* elements
. The first element is at the entrance so there are one more beam elements than elements
in the beam line.

s := ⟨0, *seq*(*cell_e*[*i*]:-*s*, *i* = 1 ..*numelems*(*cell_e*))⟩ :
#``get the distances from the line, but need to pre-pend 0

plots:-*display*(*seq*(*plot*(*s*, ⟨*seq*(*BeamVector*[*i*]:-*Coordinates*[j][1], *i* = 1 ..*numelems*(*s*))⟩), *j*
= 1 ..100),
labels = [*typeset*("Distance (m)"), *typeset*('*x*', " (m)")])



Note that the plotting uses two nested *seq*s: the inner one gets one track and plots it *vs s*; the outer one plots each trace. The *plots:-display* function puts all traces onto one graph.

Example 2: Multi-turn Tracking Example

This example demonstrates how a *Lattice* model can be used for multi-turn tracking by making a floating-point map out of the trackFunction.

The elements of a ring

$QFh := Quad(0, 27^{-1})$:

$QDh := Quad(0, -14^{-1})$:

$HBh := Bend(5\,2^{-1}, \pi\,8^{-1}, 0)$:

$FODO := DefineLine(QFh, HBh, HBh, QDh, QDh, HBh, HBh, QFh)$:

With exact parameters, Maple keeps everything exact:

$simplify(cosmux(FODO))$

(5.2.1)

$$\frac{10}{189}\quad\frac{13\,\pi-20}{\pi^2} \tag{5.2.1}$$

The trackFunction is a function acting on a 6-element Vector (particle vector or pv). It can be converted to an expression by specifying the elements of pv explicitly, which allows to unapply later in terms of pv and using the modified function as a map for tracking. It is then possible to convert all constants in the map to floating-point numbers and do other simplifications like series expansion and truncation. This is of advantage if tracking or other analysis is to be performed on the trackFunction of a larger structure and in many cases required to obtain a result in a reasonable amount of time.

$FODO\text{:-}TF\left(\langle pv_1, pv_2, pv_3, pv_4, pv_5, pv_6\rangle\right)$: # *a long expression, still exact*

In order to make tracking possible, we reduce the entries in FODO:-TF to floating-point numbers:

$$`~`_{evalf}\left(FODO\text{:-}TF\left(\langle pv_1, pv_2, pv_3, pv_4, pv_5, pv_6\rangle\right)\right)$$

$$\begin{bmatrix} 0.111725124208604\,pv_1 + 9.26108868488540\,pv_2 + 7.56530082140020\,pv_6 \\ -0.106630821733977\,pv_1 + 0.111725124208597\,pv_2 + 0.908158347416860\,pv_6 \\ 0.523809523809524\,pv_3 + 6.42857142857143\,pv_4 \\ -0.112874779541446\,pv_3 + 0.523809523809524\,pv_4 \\ -0.908158347416850\,pv_1 - 7.56530082140017\,pv_2 - 3.62039421017390\,pv_6 + pv_5 \\ pv_6 \end{bmatrix} \tag{5.2.2}$$

And then unapply to make it into a function again. This is then our map:

$Map := unapply(\textbf{(5.2.2)}, pv)$

$$\begin{aligned} Map := pv \rightarrow rtable\Big(&1..6, \{1 = 0.111725124208604\,pv_1 + 9.26108868488540\,pv_2 \\ &+ 7.56530082140020\,pv_6, 2 = -0.106630821733977\,pv_1 + 0.111725124208597\,pv_2 \\ &+ 0.908158347416860\,pv_6, 3 = 0.523809523809524\,pv_3 + 6.42857142857143\,pv_4, 4 \\ &= -0.112874779541446\,pv_3 + 0.523809523809524\,pv_4, 5 = \\ &-0.908158347416850\,pv_1 - 7.56530082140017\,pv_2 - 3.62039421017390\,pv_6 \\ &+ pv_5, 6 = pv_6\}, datatype = anything, subtype = Vector_{column}, storage = rectangular, \\ &order = Fortran\_order\Big) \end{aligned} \tag{5.2.3}$$

Now we can setup for a tracking loop:

$nturns := 100$:

$coords := \langle 0.001, 0, 0, 0.0001, 0, 0\rangle$:

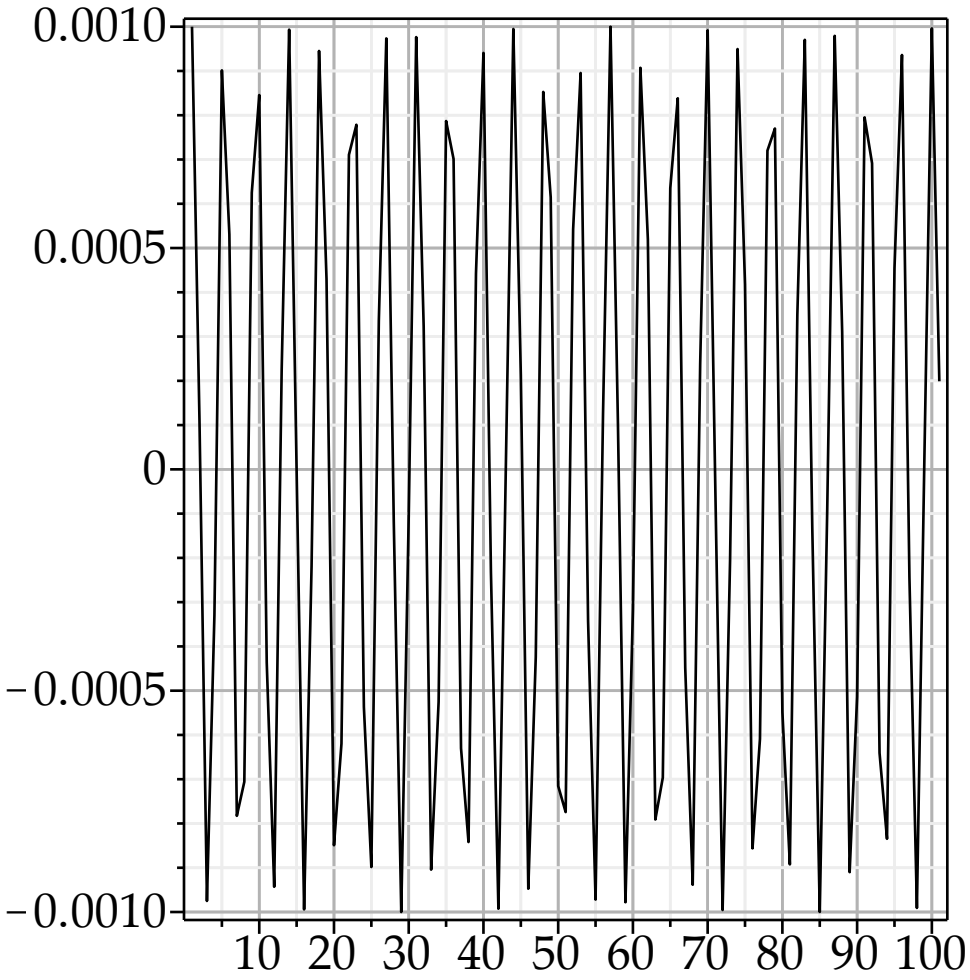
$pvV := Array(0..nturns)$:

$xv := Vector(nturns + 1, datatype = float)$:

$pvV_0 := coords$: $xv_1 := coords_1$:

And track:

```
> for turn to nturns do
    pvV[turn] := `~`[evalf](Map(pvV[turn-1]));
    xv[turn+1] := pvV[turn][1]
    end do:
  plots:-listplot(xv)
```



Installation

The *Lattice* package is distributed as a zipped archive containing the following files:

- `Lattice.mla` Maple library file, needs to be placed in a directory included in Maple's *libname* data structure.

  At present this file is generated using Maple 2015.
- `Lattice.mpl` Text file with the source of the *Lattice* package (code in Maple syntax).
- `Lattice.mw` A small Maple notebook run once to install the *Lattice* package.
- `Lattice.help` The Help database for Lattice (new style Help facility for Maple 18 and newer).
- `Lattice.hdb` The Help database for Lattice (old style Help facility for Maple 17 and older).

- `The Lattice Package Users Guide.mw` (this file).
- `The Lattice Package Users Guide.pdf` (pdf of this file).

In order to install *Lattice*, the user should decide where the library file should go. This could be a directory within the Maple installation directory tree, or a directory in the users home directory. Then pre-pend this path to the entries already in *libname*: `libname:="/Path/to/directory", libname:` The path is absolute.

The quick way to install is to then copy `Lattice.mla` to this directory. This is sufficient if a user does not plan to modify the package.

In order to be able to install modified versions, open `Lattice.mw`. Edit the path to `Lattice.mpl` in the first command (`read(…)`) to point to where `Lattice.mpl` is located (absolute path), which can be anywhere. Execute the file (*!!!* button). If no error is generated, `Lattice.mla` is generated and installed. Save the edited version of `Lattice.mw` for future use.

The Help database files go into the same directory as Lattice.mla.

Finally, test your installation with `restart;with(Lattice);` If all is well, the *Lattice* package will print an informational message during loading even when terminating the with command with a colon (:). Search for Lattice (with capital L) in Help to verify the Help database is installed properly.

Known Issues and Limitations as of Version 1.0.2, 10-Feb-2016 built.

Bends, **Quad**rupoles and **Drift**s are implemented to first order only, even in their tracking function.

Sextupoles are implemented as thin elements.

The **RfCavity** element is not properly debugged and should be considered experimental.

The **Solenoid** element is not properly debugged and should be considered experimental.

Keeping track of *Eref* is spotty and should be considered experimental.

The **Foil** scatters particles upon tracking but does not model energy loss. The ∑ matrix for the beam is not recalculated upon passing through the foil.

Subs will only accept a sequence of replacement equations, not a list. On occasion, *Subs* has failed to recurse into sublines but in sufficiently inconsistent and rare cases that I have not been able to uncover the underlying issue.

The *nux* and *nuy* functions do not correctly track the integer part of the tunes even if given an *ExpandedLine* as argument.

In older versions of Maple (specifically in Maple 15), the output from *Lattice* elements or *BeamLines* can confuse the Standard GUI and lead to lost output and output lines printing on top of each other.

This effect (which is a Maple problem, not a problem of the *Lattice* package) is mitigated by setting the typeset level to standard in Maple versions earlier than 16. In Maple 2015 this problem has not been observed. Note that there is no automatic way to change this setting back.

When calculating the tracking function (map) of a beam line, execution time rises steeply with the number of elements in a beam line. Do not attempt to find the tracking function of a full machine in the straightforward way as it most likely will not work and lock up Maple. This problem can be mitigated by using evalf to convert the coefficients into floating point format thereby greatly speeding up the calculations. See Example 2.

The *Track* operation works reasonably well for a single pass of many particles. For multi-turn operation, create the map and track by applying this map to the particle vector to avoid unreasonably long computing times. See Example 2.

The linkage to the Help files and their formatting is not always correct. Starting from the top *Lattice* help page usually works, however. The change of format of the Help database has not been of help to this project. The Help pages need more examples.

The *Lattice* package is compatible with Maple versions 15 and later (tested up to Maple 2015).

Acknowledgments

A part of this work was supported by the US Department of Energy under Contract No. DE-AC02 -76SF00515.

References

1. K.L. Brown, *A First- and Second-Order Matrix Theory for the Design of Beam Transport Systems and Charged particle Spectrometers*, SLAC-75, June 1982.
2. K.L. Brown and R.V. Servranckx, First and Second-order Charged Particle Optics, SLAC-PUB-3381, July 1984.
3. MAD Methodical Accelerator Design, H. Grote, CERN, http://mad.web.cern.ch/mad/ F.C. Iselin, *The MAD Program Physical Methods Manual*, CERN/SL/92.
4. A.W. Chao and M. Tigner, *Handbook of Accelerator Physics and Engineering*, 3rd. printing, World Scientific, Singapore, 2006.
5. Particle Data Group, http://pdg.lbl.gov/