Standard results and Code based results

or how the ubiquitous codes set a hierarchy of methods

Étienne Forest[1,2]

[1]High Energy Research Accelerator Organization Tsukuba, Japan

[2]Department of Accelerator Science
Graduate University for Advanced Studies, Hayama, Japan

December 2017

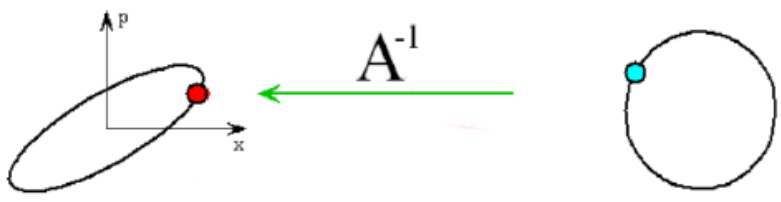
Outline

What is the purpose of this lecture

- Courant-Snyder theory, if presented in a correct framework, is a very general theory
  - linear motion
  - coupled motion
  - nonlinear motion
  - magnet modulation
  - linear/nonlinear phase slip
  - one-resonance theory
  - system with radiation
  - all the above with spin
  - etc. . .
- This theory elevates the tracking code on a pedestal because it is closest to the real machine
- it uses only things available to the code
- It assumes that the tracking code can produce approximate Taylor maps
- It is a primarily geometrical theory based on the orbits produced by the code
- It is a highly hierarchical theory well suited to modern computers and thus easily understandable
- It can make the connection with the world of analytical perturbation theory shown in text books
  - Formulae based on delta-function expansion perturbations of the ring,for example, $d\nu = \frac{1}{4\pi}\beta k\, ds$
  - Formulae based on a Fourier-expansion of a perturbation around the ring such as Guignard's theory

Why a Courant-Snyder theory?

From geometry



From equation of motion

$$x(s) = \sqrt{2\beta_s J}\cos\left(\int_0^s \frac{1}{\beta_s} ds + \phi_0\right)$$

For example we can answer the question: what is $\langle x^2 \rangle$ for a distribution of particles or a single particle (time average).

The advantage of the geometrical description will be its enormous generality. If we know A^{-1}, say from a tracking code, we are done.

Textbook results of Courant-Snyder Theory

What is really behind it?

The following results are derived using ODE theory in many textbooks: S.Y. Lee, H. Wiedemann, etc... They are really a parameterization.

- Equation of motion in the simplest case

$$\ddot{x} + k(s)x = 0 \tag{1}$$

- Assuming $k(s + C) = k(s)$ and periodicity
- The Courant-Snyder "solution" emerges

$$x(s) = \sqrt{2\beta_s J}\cos\left(\int_0^s \frac{1}{\beta_s} ds + \phi_0\right) \tag{2}$$

β_s is a periodic function ($\beta_{s+C} = \beta_s$) and J is a constant.

Why it is not a "solution"
It is a parameterization

The standard result is really a parameterization

- a parametrisation assuming stability

$$|x(s)| \leq \sqrt{2\beta_{max}J} \tag{3}$$

- Indeed if $k = 0$, it is not stable. The solution has a different form

$$x(s) = x_0 + sx_0' \tag{4}$$

- And if $k < 0$ and constant, it is exponentially unstable, and it has also a different form

$$x(s) = A\cosh ks + B\sinh ks \tag{5}$$

- Therefore Eq. (2) can represent only the stable linear motion of an accelerator in 1-d-f.

Problems with generalisation

- This analytical parametrisation assumes a simple analytical form : $\ddot{x} + k(s)x = 0$
- Can we use that easily in a code?
- Can it be **easily** generalised to coupled systems, to non-linear systems or to spin for example?

The answer to these questions is "**no**".

Most importantly, it is not easy to implement generally in a code. But there exists an approach which is easy to put in a code **no matter how complicated and discontinuous** the equations of motion are!

Goal of the slides that will follow

Sorry for the crowded transparency

Basic goal is to derive Eq. (2): $x(s) = \sqrt{2\beta_s J}\cos\left(\int_0^s \frac{1}{\beta_s} ds \; + \; \phi_0\right)$

1. Parameterize the motion using finite maps (matrices in this case)
2. Introduce the assumption of stability and area preservation(Hamiltonian flow)
3. "normalise the map", i.e., exploit stability
4. Consider the existence of two points of observation in a ring
5. Use an infinite number of points, i.e., Eq. (1): $\ddot{x} + k(s)x = 0$
6. Then using item 5, derive Eq. (2)

Finally we must convince ourselves that this is the only reasonable starting point for the more general theory (coupling, non-linear, spin, etc....) which is faithful to a code. That takes a bit of thought and work.

Examine first the one-turn map

Symplectic=Hamiltonian

One-turn map at $s = 0$ computed from the code:

$$\begin{pmatrix} x(C) \\ p(C) \end{pmatrix} = \underbrace{\begin{pmatrix} M_{11} & M_{12} \\ M_{21} & M_{22} \end{pmatrix}}_{M} \begin{pmatrix} x(0) \\ p(0) \end{pmatrix} \quad \text{where } p = x' \qquad (6)$$

The Hamiltonian nature of the flow implies, in the linear 1-d-f case, that

$$det \begin{pmatrix} M_{11} & M_{12} \\ M_{21} & M_{22} \end{pmatrix} = |M| = M_{11}M_{22} - M_{12}M_{21} = 1 \qquad (7)$$

Go back to stability

Examine first the one-turn map

$$det(M - \lambda I) = 0 \;\; , \;\; \det(M) = 1 \text{ and stability} \;\; \Rightarrow \;\; \left| \frac{1}{2} Tr(M) \right| \; < \; 1$$

$$\Downarrow$$

$$\lambda = \exp\left(: \pm i\mu :\right) = \cos\left(\mu\right) \; \pm \; i \sin\left(\mu\right) \quad \text{where} \quad \cos\left(\mu\right) = \frac{1}{2} Tr(M)$$

Normalisation

We look for

$$M = \begin{pmatrix} A_{11} & A_{12} \\ A_{21} & A_{22} \end{pmatrix} \begin{pmatrix} \cos(\mu) & \sin(\mu) \\ -\sin(\mu) & \cos(\mu) \end{pmatrix} \begin{pmatrix} A_{22} & -A_{12} \\ -A_{21} & A_{11} \end{pmatrix}$$

where $det(A) = 1$.

This insures, in 1-d-f, that the map A is Hamiltonian and the invariant J is indeed invariant. $2\pi J$ is the area circumscribed by the trajectory and it is preserved if A is area preserving.

The preservation of J is also true for nonlinear multidimensional maps if the map A is a nonlinear symplectic transformation

Choice for A

Notice that if

$$M = ARA^{-1}$$

$$\Downarrow$$

$$M = ArRr^{-1}A^{-1}$$

The rotation r commutes with R and can be chosen such that

$$Ar \implies \begin{pmatrix} A_{11} & 0 \\ A_{21} & A_{11}^{-1} \end{pmatrix}$$

Result for A
Invariant

$$A_{11} = \left| \frac{M_{12}}{\sin(\mu)} \right|^{1/2} \qquad A_{21} = \frac{\left(M_{21} A_{11} + \sin(\mu) A_{11}^{-1} \right)}{\cos(\mu) - M_{22}}$$

We can compute an invariant using A^{-1}:

$$\varepsilon = 2J = \underbrace{\left({A_{11}^{-1}}^2 + {A_{21}}^2 \right)}_{\gamma} x^2 + 2 \underbrace{\left(-A_{21} A_{11} \right)}_{\alpha} xp + \underbrace{A_{11}^2}_{\beta} p^2 \qquad (8)$$

We will see that ε is indeed $2J$ of $\mathrm{Eq.}$ (2)

Writing M in terms of A

Invariant

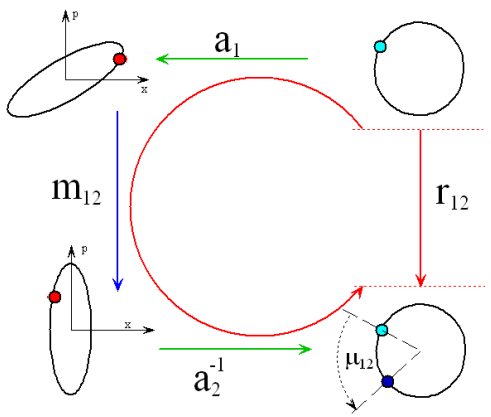
$$
\begin{aligned}
M &= \begin{pmatrix} A_{11} & A_{12} \\ A_{21} & A_{22} \end{pmatrix} \begin{pmatrix} \cos(\mu) & \sin(\mu) \\ -\sin(\mu) & \cos(\mu) \end{pmatrix} \begin{pmatrix} A_{22} & -A_{12} \\ -A_{21} & A_{11} \end{pmatrix} \\
&= \begin{pmatrix} \cos(\mu) - (A_{11}A_{21} + A_{12}A_{22})\sin(\mu) & (A_{11}^2 + A_{12}^2)\sin(\mu) \\ -\left(A_{22}^2 + {A_{21}}^2\right)\sin(\mu) & \cos(\mu) + (A_{11}A_{21} + A_{12}A_{22})\sin(\mu) \end{pmatrix} \\
&= \begin{pmatrix} \cos(\mu) + \alpha\sin(\mu) & \beta\sin(\mu) \\ -\gamma\sin(\mu) & \cos(\mu) - \alpha\sin(\mu) \end{pmatrix} \qquad (9)
\end{aligned}
$$

Notice that $1 + \alpha^2 = \beta\gamma$ because $det(A) = 1$

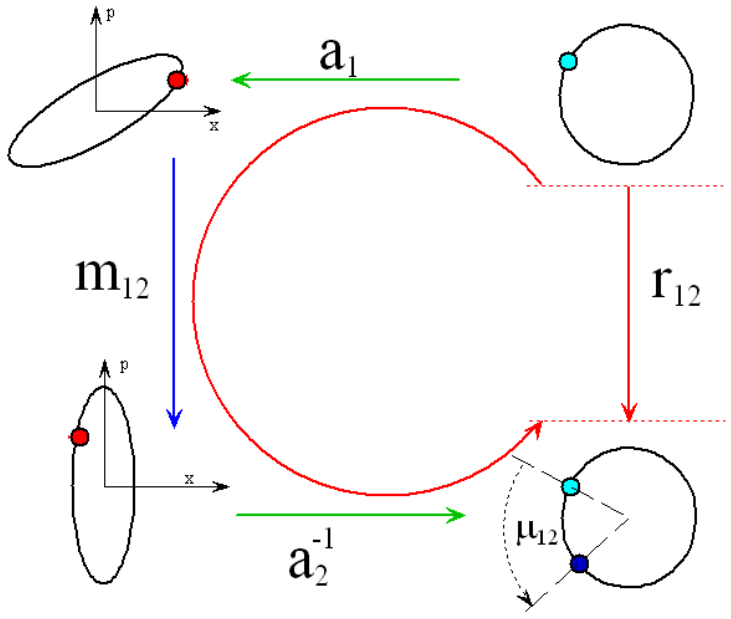
More than one point of observation in the ring

- Eq. (2), i.e., $x(s) = \sqrt{2\beta_s J}\cos\left(\int_0^s \frac{1}{\beta_s} ds + \phi_0\right)$, assumes more than one location
- in fact an infinite number since s is a continuous variable: It cannot be compatible with a tracking code
- So we first derive Eq. (2) for two locations: 1 location is a trivial degenerate case

We reduce Eq. (2) to a diagram



Phase advance diagram



1. Assume m_{12}, a_1 and a_2 are area preserving
2. a_1 and a_2 turn the one-turn maps m_1 and m_2 into rotations
3. Then r_{12} must be a rotation if a_1 and a_2 are area preserving (symplectic in general)
4. In what follows we assume that they are linear maps

Here we go....

$$M_1 = A_1 R A_1^{-1} \qquad M_2 = A_2 R A_2^{-1}$$

$$\Downarrow$$

$$M_{12} = A_2 R_{12} A_1^{-1} \quad \leftarrow \text{ Using phase advance figure}$$

$$A_k = \begin{pmatrix} \sqrt{\beta_k} & 0 \\ -\alpha_k/\sqrt{\beta_k} & 1/\sqrt{\beta_k} \end{pmatrix} \qquad k = 1, 2$$

$$M_{12} = \begin{pmatrix} \cos(\mu_{12}) & \sin(\mu_{12}) \\ -\sin(\mu_{12}) & \cos(\mu_{12}) \end{pmatrix}$$

- Notice that μ_{12} depends on the choice for A_k and also depends on M_{12}.
- In a code, we cannot say much more.
- But to compare to Eq. (2), we compute x_2 in terms of x_1.

x_2 in terms of x_1

$$x_2 = \sqrt{\beta_2}\underbrace{\left\{\frac{1}{\sqrt{\beta_1}}x_1\right\}}_{E}\cos(\mu_{12}) + \sqrt{\beta_2}\underbrace{\left\{\frac{\alpha_1}{\sqrt{\beta_1}}x_1 + \sqrt{\beta_1}p_1\right\}}_{F}\sin(\mu_{12})$$

$$x_2 = \sqrt{\beta_2\left(E^2+F^2\right)}\cos(\mu_{12}+\phi_1) \tag{10}$$

Bonus result p_2:

$$p_2 = -\frac{1}{\sqrt{\beta_2}}\sqrt{\left(E^2+F^2\right)}\left\{\alpha_2\cos(\mu_{12}+\phi_1) + \sin(\mu_{12}+\phi_1)\right\}$$

$$\varepsilon_1 = E^2 + F^2 = 2J_1 = \gamma_1 x_1^2 + 2\alpha_1 x_1 p_1 + \beta_1 p_1^2 \tag{11}$$

$$\text{where}\quad \cos(\phi_1) = \frac{E}{\sqrt{E^2+F^2}} \quad \text{and} \quad \sin(\phi_1) = -\frac{F}{\sqrt{E^2+F^2}}$$

What if $\ddot{x} + k(s)x = 0$?

Continuous case

- Eqs. (10) and (11) are correct for any 1-d-f system and therefore applicable to any code
- μ_{12} in a code depends on details unavailable to the usual user: fringe fields, misalignments, bizarre magnets, etc...
- Therefore $\mu_{12} = \int_{s_1}^{s_2} \frac{ds}{\beta(s)}$ is valid for our choice of A_s for the underlying equation $\ddot{x} + k(s)x = 0$

Let us derive $\mu_{12} = \int_{s_1}^{s_2} \frac{ds}{\beta(s)}$ by assuming that position 1 and 2 are separated by an infinitesimal length of ds

Phase advance using maps

Continuous case

$$M_{12} = A_2 R_{12} A_1^{-1} \implies M_{12} A_1 R_{12}^{-1} = A_2$$

$$\Downarrow$$

$$M_{12} = \begin{pmatrix} 1 & ds \\ -k ds & 1 \end{pmatrix} \quad A_1 = \begin{pmatrix} \sqrt{\beta_s} & 0 \\ -\alpha_s/\sqrt{\beta_s} & 1/\sqrt{\beta_s} \end{pmatrix} \quad R_{12}^{-1} = \begin{pmatrix} 1 & -d\mu \\ d\mu & 1 \end{pmatrix}$$

$$\Downarrow$$

$$A_2 = \begin{pmatrix} \sqrt{\beta_s} - ds\,\alpha_s/\sqrt{\beta_s} & -d\mu\sqrt{\beta_s} + ds/\sqrt{\beta_s} \\ -\alpha_s/\sqrt{\beta_s} + ds/\beta_s^{3/2} - k\sqrt{\beta_s} ds & 1/\sqrt{\beta_s} + ds\,\alpha_s/\beta_s^{3/2} \end{pmatrix} + O(ds, d\mu)^2$$

Our choice for A_2 requires that the $A_{2;12}$ be zero:

$$-d\mu\sqrt{\beta_s} + ds/\sqrt{\beta_s} = 0 \implies \frac{d\mu}{ds} = \frac{1}{\beta_s} \implies \mu_{12} = \int_{s_1}^{s_2} \frac{ds}{\beta_s}$$

Q.E.D.

Time evolution of lattice functions

Corollary

- $A_{11} = \sqrt{\beta_s} - ds\ \alpha_s/\sqrt{\beta_s}$

$$\Longrightarrow \frac{d\beta_s}{ds} = -2\alpha_s$$

- $A_{21} = -\alpha_s/\sqrt{\beta_s} + ds/\beta_s^{3/2} - k\sqrt{\beta_s}ds$

$$\Longrightarrow \frac{d\alpha_s}{ds} = \frac{1}{2}\frac{\alpha_s}{\beta_s}\frac{d\beta_s}{ds} + k\beta_s - \frac{1}{\beta_s} = k\beta_s - \gamma_s$$

- Finally,

$$\frac{d\gamma_s}{ds} = 2\alpha_s k$$

Hamiltonian in our example code

The following Hamiltonian is implemented in our small code and in PTC:

$$H = \frac{p^2}{2\,(1+\delta)}\,(1 + e_1 k_q) + b_0 \left(x + h\frac{x^2}{2} \right) - hx\,(1+\delta) + \frac{k_q}{2}x^2 + \frac{k_o}{4}x^4 \qquad (12)$$

1. e_1 is added to show later than a code based theory is impervious to "false" physics
2. δ is the momentum deviation from the design momentum k_q and k_o are the quadrupole and octupole strength respectively
3. The Hamiltonian of Eq. (12) with $e_1 = 0$ represents motion in the small angle approximation in the presence of a quadrupole-octupole system.

The Courant-Snyder small angle Hamiltonian

$$H = \frac{p^2}{2(1+\delta)}(1+e_1 k_q) + b_0\left(x + h\frac{x^2}{2}\right) - hx(1+\delta) + \frac{k_q}{2}x^2 + \frac{k_o}{4}x^4 \tag{13}$$

$$\frac{dx}{ds} = \frac{(1+e_1 k_q)}{(1+\delta)}p \tag{14}$$

$$\frac{dp}{ds} = -b_0(1+hx) + h(1+\delta) - k_q x - k_o x^3 \tag{15}$$

$$\Downarrow$$

$$\ddot{x} + \frac{(1+e_1 k_q)k_q}{1+\delta}x = \frac{(1+e_1 k_q)k_o}{1+\delta}\left(-b_0(1+hx) + h(1+\delta) - k_o x^3\right)$$

We now switch from our small code to PTC

Using the polymorph real_8

Code for the Taylor series in PTC

Using the polymorph real_8

Second order Symplectic Integrator Drift-Kick-Drift type

```
CASE(2)
   DH=EL%L/2.0_dp/EL%P%NST
   D=EL%L/EL%P%NST
   DD=EL%P%LD/2.0_dp/EL%P%NST



   CALL DRIFT(DH,DD,EL%P%beta0,k%TOTALPATH,EL%P%EXACT,k%TIME,X)
   if(e1_cas/=0.and.el%p%nmul>1) then
      X(1)=X(1)+DH*X(2)/SQRT(1.0_dp+2.0_dp*X(5)/EL%P%beta0+x(5)**2)*e1_cas*el%bn(2)
   endif
   CALL KICK (EL,D,X,k)
   CALL DRIFT(DH,DD,EL%P%beta0,k%TOTALPATH,EL%P%EXACT,k%TIME,X)
   if(e1_cas/=0.and.el%p%nmul>1) then
      X(1)=X(1)+DH*X(2)/SQRT(1.0_dp+2.0_dp*X(5)/EL%P%beta0+x(5)**2)*e1_cas*el%bn(2)
   endif
```

real(dp) —-> real_8 where dp=8

What can the code do?

1. You can view the code as a black box function giving you a propagator from point i to $i + 1$
2. i can denote any "Poincaré" surface of section in the lattice including the integration steps
3. The propagator propagates phase space $(x, p_x, \dots)$, spin, etc...
4. Through the magic of polymorphism, that code can propagate Taylor series
5. If you can propagate Taylor series, then you can propagate approximates maps such as linear maps, i.e., matrices

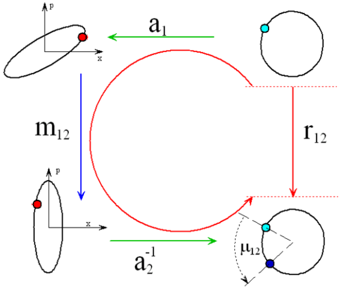
How to treat a library analysing Taylor maps

Normal form library

1. It will have normal formal form procedures
   1. *For linear maps*
   2. *nonlinear maps*
   3. *radiative maps*
   4. *spin motion*
   5. *etc. . .*
2. It will manipulate maps into different forms
   1. *Taylor series representations (so-called matrices)*
   2. *Lie representation*
   3. *Generating functions for Hamiltonian maps*
   4. *Various useful factored representations*
   5. *etc. . .*
3. Maps, computed by the code, manipulated into a normal form, are reinserted in the code: the Twiss algorithm
   1. *Propagates lattice functions, i.e., the results of the normal form*
   2. *Computes the phase advance if wanted*
   3. *The above things for linear, nonlinear, spin, etc. . .*

The Phase Advance Diagram
Complete loop

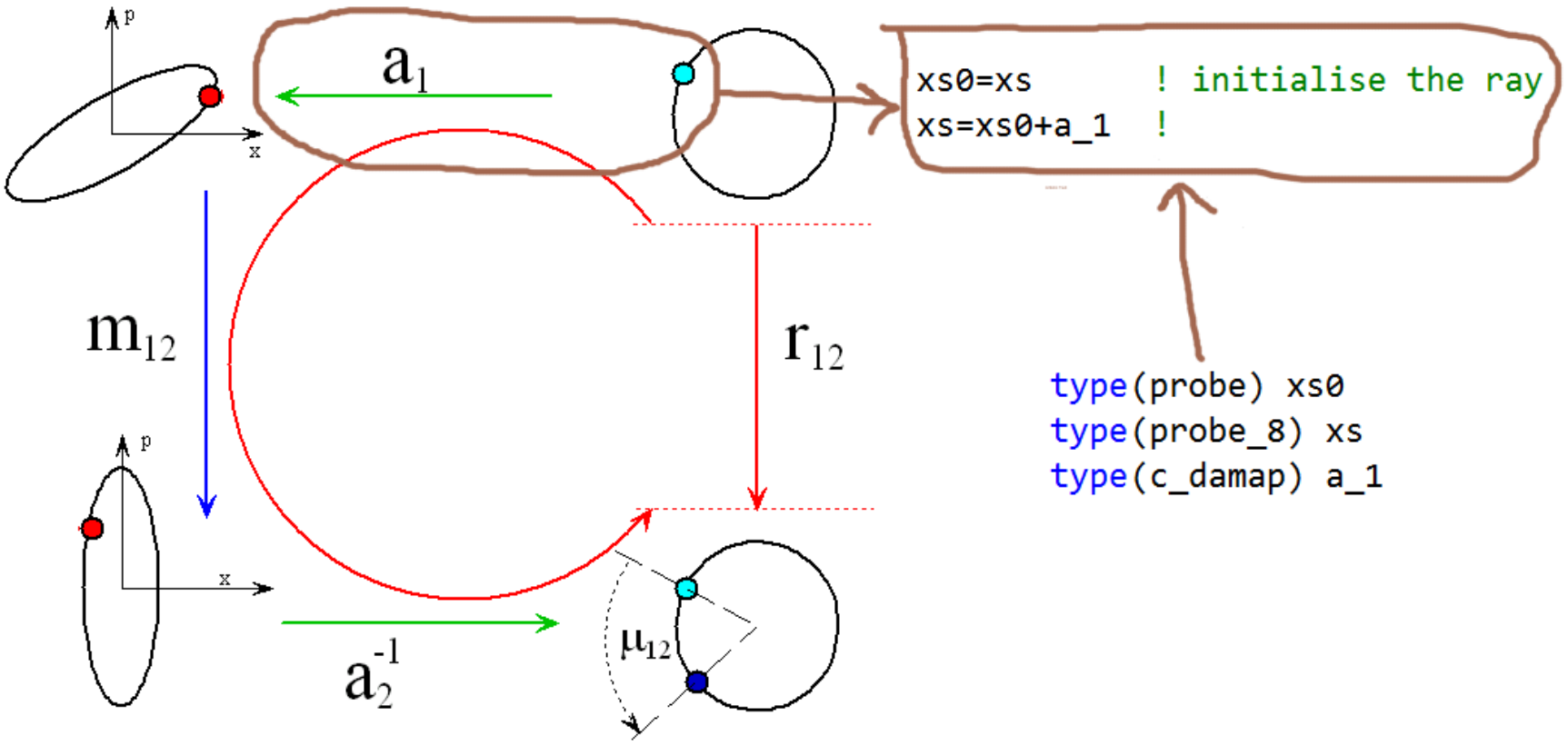
```
r=a_cs+r0

f=>ring%start

do i=1,ring%n

call propagate(ring,r,state,fibre1=i,fibre2=i+1)

a_cs=r
call c_full_canonise(a_cs,a_cs,a_spin, &
disp,A_L,A_NL,phase=phase,nu_spin=phase_spin)

r0=r
r=a_cs+r0

!!!!!!!!!!!!! Do something !!!!!!!!!!!!!!!!!!!!



!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!
```



c_full_canonise can be replaced by c_fast_canonise in a linear calculation without system parameters.

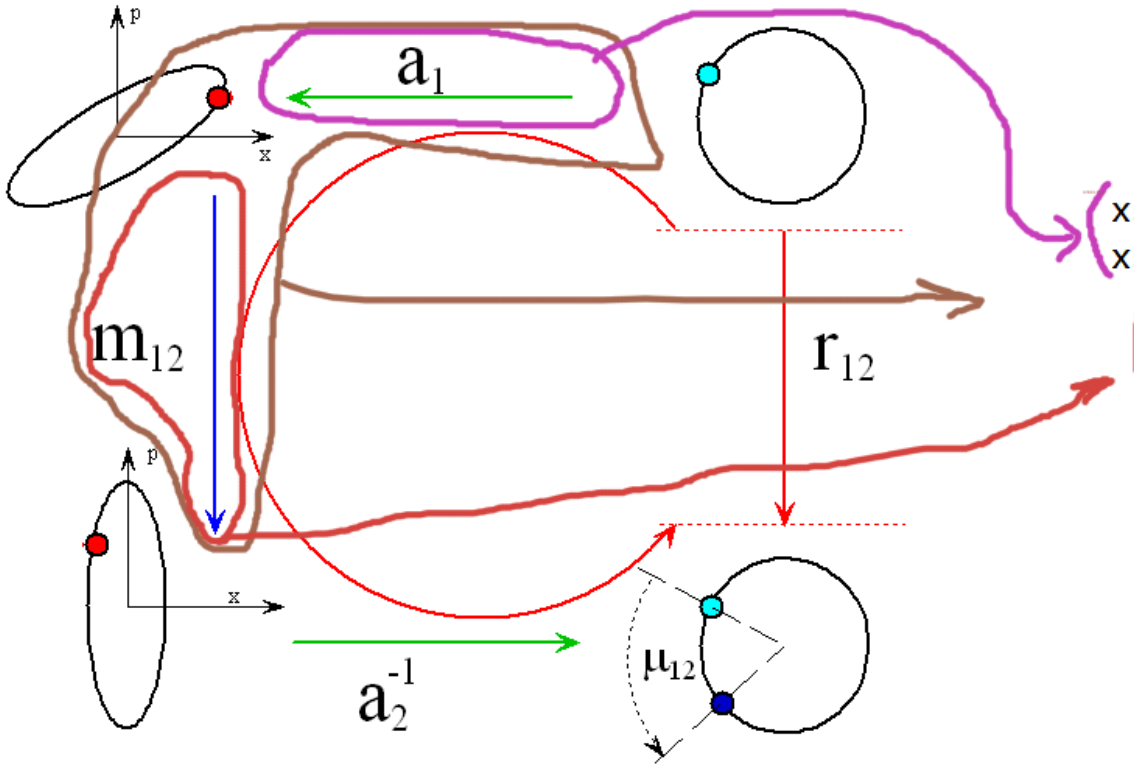
From the Advance Diagram

Initializing with a_1



From to the Advance Diagram

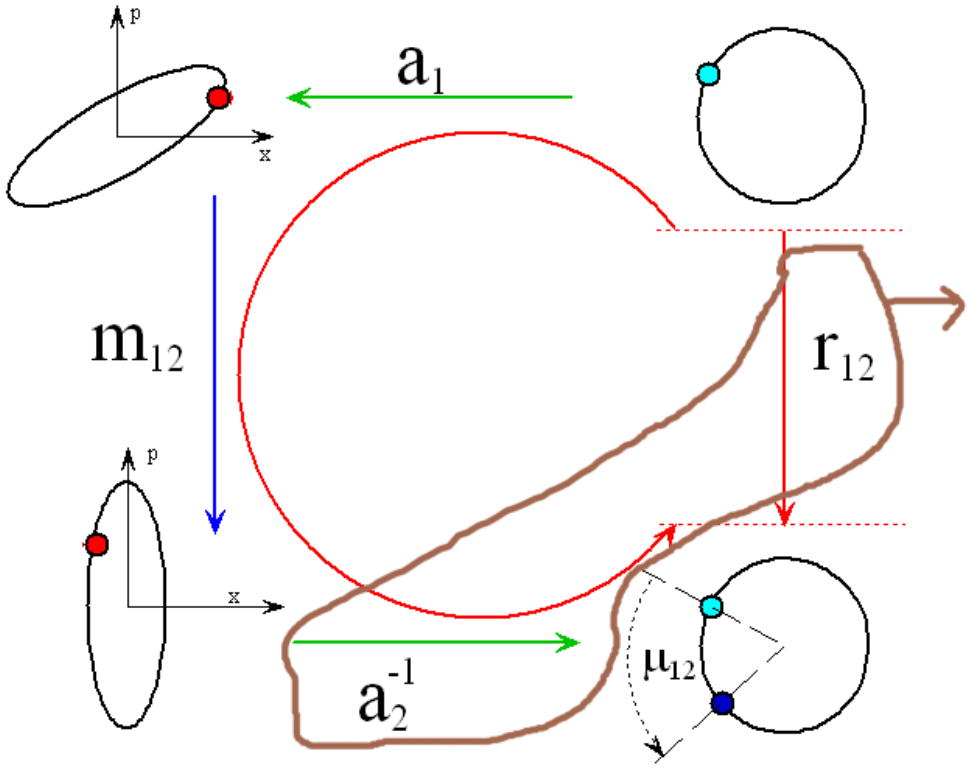
Tracking : computing m_{12}




```
xs0=xs       ! initialise the ray
xs=xs0+a_1   !
call track_fake(XS,i,i+1)
xs0=xs
u=xs    ! u=m_12 o a_1
```

From to the Advance Diagram

Finding a_2 and r_{12}



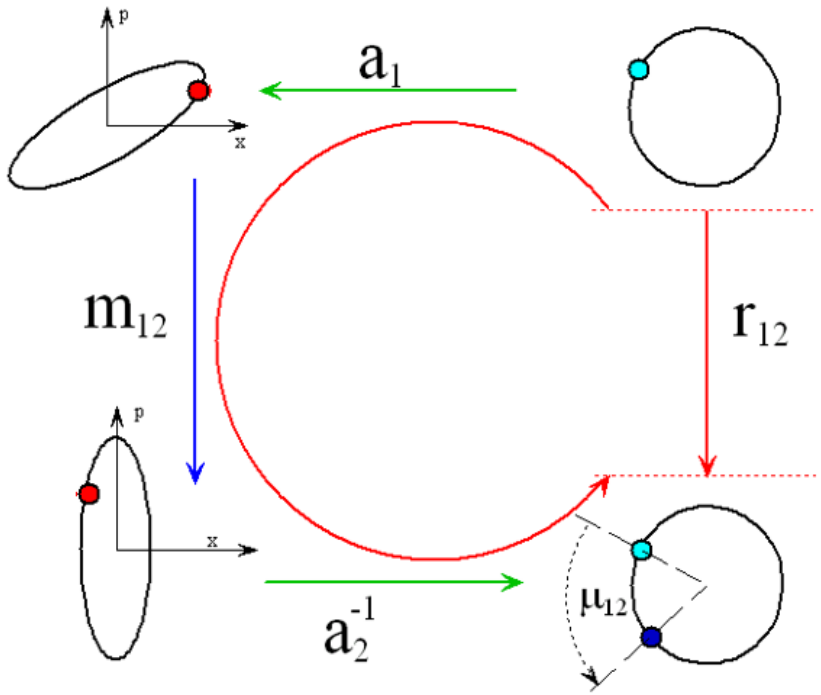

```
u=xs    ! u=m_12 o a_1
! u = a_2 o r_12(phase)
call c_full_canonise(u,a_2,phase=phase)
```

$$a_2 = \begin{pmatrix} \sqrt{\beta_2} & 0 \\ -\frac{\alpha_2}{\sqrt{\beta_2}} & \frac{1}{\sqrt{\beta_2}} \end{pmatrix}$$

The Phase Advance Diagram

Complete loop

```
r=a_cs+r0

f=>ring%start

do i=1,ring%n

call propagate(ring,r,state,fibre1=i,fibre2=i+1)

a_cs=r
call c_full_canonise(a_cs,a_cs,a_spin, &
disp,A_L,A_NL,phase=phase,nu_spin=phase_spin)

r0=r
r=a_cs+r0

!!!!!!!!!!!!! Do something !!!!!!!!!!!!!!!!!!!!

!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!
```



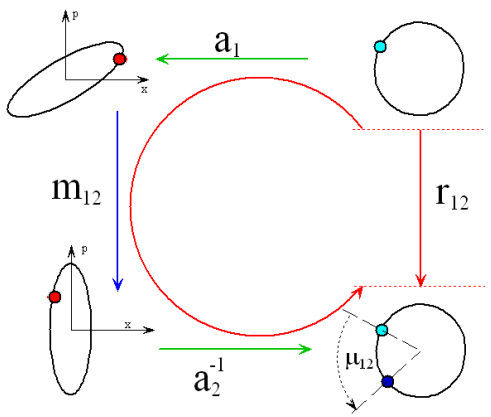
Same Twiss Loop from the Code "PTC"

A basic Twiss loop

`xs=xs0+a_1` $\Longleftarrow A_1 = \begin{pmatrix} \sqrt{\beta_{x;1}} & 0 \\ -\alpha_{x;1}/\sqrt{\beta_{x;1}} & 1/\sqrt{\beta_{x;1}} \end{pmatrix}$ $k = 1, 2$

```
do i=1,ring%n
 call propagate(ring,xs,state,fibre1=i,fibre2=i+1)
  !!!!!!!!!!!!!!!   doing something  !!!!!!!!!!!!!!!!
```

$\Longleftarrow$ This is the $m_{12} \circ a_1$ of the figure




```
  !!!!!!!!!!!!!!!  end of doing something  !!!!!!!!!!!!!!!!
enddo
```

What am I trying to convey?

- The difference between my approach and the standard approach is epistemological in nature
- It is about conceptual knowledge and its organisation
- What does it mean to "understand" something?
- Understand entails a hierarchical organisation of physics concepts
- The more the knowledge is hierarchical, the greater our comprehension
- This makes it easier to perform traditional calculations as well:
  - Analytic formulae such

$$\frac{\partial \langle \widetilde{x} \rangle}{\partial J} = \frac{\beta_s^{1/2}}{2\left(1 - \cos\left(\mu\right)\right)} \oint_0^C \left(-\sin\left(\mu_{s\sigma}\right) + \sin\left(\mu_{s\sigma} - \mu\right)\right) \beta_\sigma^{3/2} k_S(\sigma) d\sigma$$

  - Guignard-Deprit calculation using a mode decomposition in the variable around the ring (s or $\theta(s)$)

We can choose the appropriate methods not on the basis of our taste but on the basis of real understanding

(taste = wimpy euphemism for our incompetence or ignorance)

Example of non-hierarchical knowledge

1. Newton laws seem to predict accurately the motion of planets (Concept)
2. Mercury's orbit cannot be accounted
3. We conclude the following
    1. Some invisible planet must be present in the inner solar system (Newton Laws unchanged)
    2. Some other force must exist which acts on Mercury (Newton Laws complemented)
    3. Newton's Laws are not completely correct (Newton Laws superseded by a better set)

In the absence of evidence, Mercury's orbit is a perception of reality which is not derivable from Newton's Laws. It stands alone and it is not part of any hierarchy.

Therefore I say : we do not understand why Mercury's orbit precesses

Here comes Einstein!

Knowledge is now hierarchical

1. Einstein's laws seem to predict accurately the motion of planets (Concept)
2. Mercury's orbit can be accounted from Einstein's General Relativity
3. So the motion of Mercury is not part of a new concept which must be learnt separately

Now we have a hierarchy. Newtow's Law is a consequence of Einstein GR which accounts for Mercury's orbit. We now understand.

Now, I go back to the Twiss loop accessible to the code.

Same Twiss Loop from the Code “PTC”

Apparently unrelated things computed in a basic Twiss loop

```
 xs=xs0+a_1

do i=1,ring%n
 call propagate(ring,xs,state,fibre1=i,fibre2=i+1)

  !!!!!!!!!!!!!!!   doing something  !!!!!!!!!!!!!!!!


  !!!!!!!!!!!!!!!  end of doing something  !!!!!!!!!!!!!!!!
```

Same Twiss Loop from the Code "PTC"

Apparently unrelated things computed in a basic Twiss loop

`xs=xs0+a_1` $\longleftarrow A_1 = \begin{pmatrix} \sqrt{\beta_{x;1}} & 0 \\ -\alpha_{x;1}/\sqrt{\beta_{x;1}} & 1/\sqrt{\beta_{x;1}} \end{pmatrix}$ $k = 1, 2$

```
do i=1,ring%n
 call propagate(ring,xs,state,fibre1=i,fibre2=i+1)

  !!!!!!!!!!!!!!!   doing something  !!!!!!!!!!!!!!!!


  !!!!!!!!!!!!!!!  end of doing something  !!!!!!!!!!!!!!!!
```

Same Twiss Loop from the Code "PTC"

Apparently unrelated things computed in a basic Twiss loop

```
xs=xs0+a_1
```

$\longleftarrow A_1 = \begin{pmatrix} \sqrt{\beta_{x;1}} & 0 \\ -\alpha_{x;1}/\sqrt{\beta_{x;1}} & 1/\sqrt{\beta_{x;1}} \end{pmatrix} \qquad k = 1, 2$

```
do i=1,ring%n

 call propagate(ring,xs,state,fibre1=i,fibre2=i+1)


  !!!!!!!!!!!!!!!   doing something  !!!!!!!!!!!!!!!!
 ! Computation of beta function defined as ∂ ⟨x^2⟩ /∂J ⟹ A_11^2 + A_12^2
```

```
  !!!!!!!!!!!!!!!  end of doing something  !!!!!!!!!!!!!!!!
```

Same Twiss Loop from the Code "PTC"

Apparently unrelated things computed in a basic Twiss loop

```
xs=xs0+a_1
```

$\longleftarrow A_1 = \begin{pmatrix} \sqrt{\beta_{x;1}} & 0 \\ -\alpha_{x;1}/\sqrt{\beta_{x;1}} & 1/\sqrt{\beta_{x;1}} \end{pmatrix} \qquad k = 1, 2$

```
do i=1,ring%n

 call propagate(ring,xs,state,fibre1=i,fibre2=i+1)


  !!!!!!!!!!!!!!!   doing something  !!!!!!!!!!!!!!!!
 ! Computation of beta function defined as ∂ ⟨x²⟩ /∂J ⟹ A²₁₁ + A²₁₂
    a_1=xs
    betax_1=(a_1%v(1).index.1)**2+(a_1%v(1).index.2)**2









  !!!!!!!!!!!!!!!  end of doing something  !!!!!!!!!!!!!!!!
```

Same Twiss Loop from the Code "PTC"

Apparently unrelated things computed in a basic Twiss loop

```
xs=xs0+a_1
```

$\longleftarrow A_1 = \begin{pmatrix} \sqrt{\beta_{x;1}} & 0 \\ -\alpha_{x;1}/\sqrt{\beta_{x;1}} & 1/\sqrt{\beta_{x;1}} \end{pmatrix} \quad k = 1, 2$

```
do i=1,ring%n

 call propagate(ring,xs,state,fibre1=i,fibre2=i+1)


  !!!!!!!!!!!!!!!   doing something  !!!!!!!!!!!!!!!!
  ! Computation of beta function defined as ∂ ⟨x^2⟩ /∂J ⟹ A_11^2 + A_12^2
     a_1=xs
     betax_1=(a_1%v(1).index.1)**2+(a_1%v(1).index.2)**2
  ! Modulating the strength of magnet (a bend here) : useful in spin experiments I am told by the experts
  ∂ ⟨x^2⟩ /∂J_m ⟹ A_15^2 + A_16^2



  !!!!!!!!!!!!!!!  end of doing something  !!!!!!!!!!!!!!!!
```

Same Twiss Loop from the Code "PTC"

Apparently unrelated things computed in a basic Twiss loop

```
xs=xs0+a_1
```

$\longleftarrow A_1 = \begin{pmatrix} \sqrt{\beta_{x;1}} & 0 \\ -\alpha_{x;1}/\sqrt{\beta_{x;1}} & 1/\sqrt{\beta_{x;1}} \end{pmatrix} \qquad k = 1, 2$

```
do i=1,ring%n

 call propagate(ring,xs,state,fibre1=i,fibre2=i+1)


  !!!!!!!!!!!!!!!   doing something  !!!!!!!!!!!!!!!!
  ! Computation of beta function defined as ∂ ⟨x^2⟩ /∂J ⟹ A_11^2 + A_12^2
     a_1=xs
     betax_1=(a_1%v(1).index.1)**2+(a_1%v(1).index.2)**2
  ! Modulating the strength of magnet (a bend here) : useful in spin experiments I am told by the experts
  ∂ ⟨x^2⟩ /∂J_m ⟹ A_15^2 + A_16^2
     betax_m=(a_1%v(1).index.5)**2+(a_1%v(1).index.6)**2




  !!!!!!!!!!!!!!!  end of doing something  !!!!!!!!!!!!!!!!
```

Same Twiss Loop from the Code "PTC"

Apparently unrelated things computed in a basic Twiss loop

```
xs=xs0+a_1
```

$\longleftarrow A_1 = \begin{pmatrix} \sqrt{\beta_{x;1}} & 0 \\ -\alpha_{x;1}/\sqrt{\beta_{x;1}} & 1/\sqrt{\beta_{x;1}} \end{pmatrix} \qquad k = 1, 2$

```
do i=1,ring%n

 call propagate(ring,xs,state,fibre1=i,fibre2=i+1)


  !!!!!!!!!!!!!!!   doing something  !!!!!!!!!!!!!!!!
  ! Computation of beta function defined as ∂ ⟨x^2⟩ /∂J ⟹ A_11^2 + A_12^2
     a_1=xs
     betax_1=(a_1%v(1).index.1)**2+(a_1%v(1).index.2)**2
  ! Modulating the strength of magnet (a bend here) : useful in spin experiments I am told by the experts
  ∂ ⟨x^2⟩ /∂J_m ⟹ A_15^2 + A_16^2
     betax_m=(a_1%v(1).index.5)**2+(a_1%v(1).index.6)**2
  ! Computation of invariant spin field n⃗ and ∂n/∂p_y
  ! Stuff needed in lattice design with spin and in the Derbenev-Kondratenko formula



  !!!!!!!!!!!!!!!  end of doing something  !!!!!!!!!!!!!!!!
```

Same Twiss Loop from the Code "PTC"

Apparently unrelated things computed in a basic Twiss loop

```
xs=xs0+a_1
```

$\longleftarrow A_1 = \begin{pmatrix} \sqrt{\beta_{x;1}} & 0 \\ -\alpha_{x;1}/\sqrt{\beta_{x;1}} & 1/\sqrt{\beta_{x;1}} \end{pmatrix} \qquad k = 1, 2$

```
do i=1,ring%n

 call propagate(ring,xs,state,fibre1=i,fibre2=i+1)


  !!!!!!!!!!!!!!!   doing something  !!!!!!!!!!!!!!!!
  ! Computation of beta function defined as ∂ ⟨x^2⟩ /∂J ⟹ A_11^2 + A_12^2
     a_1=xs
     betax_1=(a_1%v(1).index.1)**2+(a_1%v(1).index.2)**2
  ! Modulating the strength of magnet (a bend here) : useful in spin experiments I am told by the experts
  ∂ ⟨x^2⟩ /∂J_m ⟹ A_15^2 + A_16^2
     betax_m=(a_1%v(1).index.5)**2+(a_1%v(1).index.6)**2
  ! Computation of invariant spin field n⃗ and ∂n/∂p_y
  ! Stuff needed in lattice design with spin and in the Derbenev-Kondratenko formula
     a_1%s=a_1%s*a_1**(-1)
     isf=a_1%s*e_y
     dnx_dpy=isf%v(1).index.4
  !!!!!!!!!!!!!!!  end of doing something  !!!!!!!!!!!!!!!!
```

Same Twiss Loop from the Code "PTC"

Apparently unrelated things computed in a basic Twiss loop

```
xs=xs0+a_1
```

$\longleftarrow A_1 = \begin{pmatrix} \sqrt{\beta_{x;1}} & 0 \\ -\alpha_{x;1}/\sqrt{\beta_{x;1}} & 1/\sqrt{\beta_{x;1}} \end{pmatrix} \qquad k = 1, 2$

```
do i=1,ring%n
 call propagate_ring xs state fibre1=i fibre2=i+1,
```

What about MAD?

```
 !!!!!!!!!!!!!!!   doing something   !!!!!!!!!!!!!!!!
 ! Computation of beta function defined as ∂⟨x²⟩/∂J ⟹ A²₁₁ + A²₁₂
    a_1=xs
    betax_1=(a_1%v(1).index.1)**2+(a_1%v(1).index.2)**2
 ! Modulating the strength of magnet (a bend here) : useful in spin experiments I am told by the experts
 ∂⟨x²⟩/∂J_m ⟹ A²₁₅ + A²₁₆
    betax_m=(a_1%v(1).index.5)**2+(a_1%v(1).index.6)**2
 ! Computation of invariant spin field n⃗ and ∂n/∂p_y
 ! Stuff needed in lattice design with spin and in the Derbenev-Kondratenko formula
    a_1%s=a_1%s*a_1**(-1)
    isf=a_1%s*e_y
    dnx_dpy=isf%v(1).index.4
 !!!!!!!!!!!!!!!   end of doing something   !!!!!!!!!!!!!!!!
```

Same Twiss Loop from the Code "PTC"

Apparently unrelated things computed in a basic Twiss loop

```
xs=xs0+a_1
```

$\longleftarrow A_1 = \begin{pmatrix} \sqrt{\beta_{x;1}} & 0 \\ -\alpha_{x;1}/\sqrt{\beta_{x;1}} & 1/\sqrt{\beta_{x;1}} \end{pmatrix} \qquad k = 1, 2$

```
do i=1,ring%n
```

call propagate(my_ring,A_script_probe,default,node1=nodePtr%pos,node2=nodePtr%pos+1)

What about MAD?

```
 !!!!!!!!!!!!!!!   doing something  !!!!!!!!!!!!!!!!
```

! Computation of beta function defined as $\partial \left\langle x^2 \right\rangle / \partial J \Longrightarrow A_{11}^2 + A_{12}^2$

```
   a_1=xs
   betax_1=(a_1%v(1).index.1)**2+(a_1%v(1).index.2)**2
```

! Modulating the strength of magnet (a bend here) : useful in spin experiments I am told by the experts

$\partial \left\langle x^2 \right\rangle / \partial J_m \Longrightarrow A_{15}^2 + A_{16}^2$

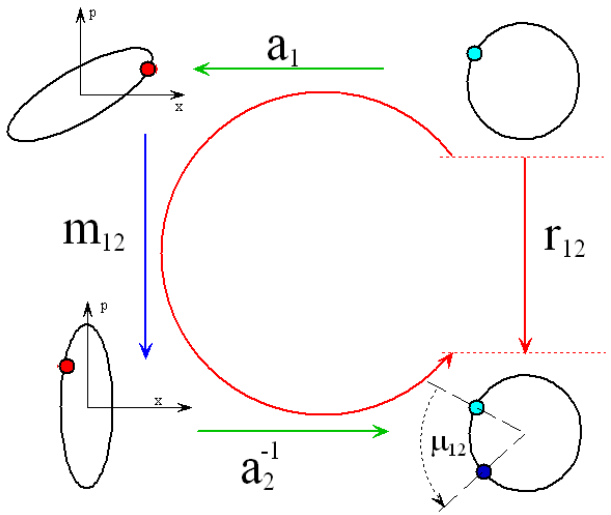
```
   betax_m=(a_1%v(1).index.5)**2+(a_1%v(1).index.6)**2
```

! Computation of invariant spin field $\vec{n}$ and $\partial n / \partial p_y$

! Stuff needed in lattice design with spin and in the Derbenev-Kondratenko formula

```
   a_1%s=a_1%s*a_1**(-1)
   isf=a_1%s*e_y
   dnx_dpy=isf%v(1).index.4
 !!!!!!!!!!!!!!!  end of doing something  !!!!!!!!!!!!!!!!
```

The linear 1-d-f case



What can we notice from the above graph?

- The phase space plots are turned into circles
- The circles have the same size (a_i's are area preserving in 1-d-f)
- Allegedly this can be extended to nonlinear maps

Now we make a bold claim

What if everything turns into circles?

$$M = a \circ n \circ a^{-1}$$

$$n = r_x \circ r_y \circ \cdots \circ r_{mod} \circ r_{spin}$$

- All the r_i's are rotations in the plane
- Their angles are the fractional tunes
- All these angles can depend on the radii of the other planes
- r_{mod} is a rotation describing the modulation frequency of a magnet family
- r_{spin} is a rotation of the spin. Its angle depends on all the other radii
- the transformation "a" acts on the orbital motion and on the spin. It transforms the spin differently at every point in phase space

This implies a **unique and universal** C-S theory

$$M = a \circ n \circ a^{-1}$$

$$n = r_x \circ r_y \circ \cdots \circ r_{mod} \circ r_{spin}$$

- All the r_i's are rotations in the plane
- Their angles are the fractional tunes
- All these angles can depend on the radii of the other planes
- r_{mod} is a rotation describing the modulation frequency of a magnet family
- r_{spin} is a rotation of the spin. Its angle depends on all the other radii
- the transformation "a" acts on the orbital motion and on the spin. It transforms the spin differently at every point in phase space

The do-loop presented before is universal.

- One simply tracks the canonical transformation a_s from $s = 0$ to some s of interest
- All imaginable and unimaginable lattice functions are gotten by manipulating s
- The phase advance is defined as the difference between the tracked a_s and a special form for a_s chosen by the physicist— Courant-Snyder form for example.

Other kind of normal form? Only circles?

- Circles + drift
  - Only circles (presented already)
  - Circles and a drift in the longitudinal plane: relevant in the absence of RF-cavities
- One-resonance normal form : a discrete rotational symmetry
  - Ordinary orbital resonance in a single plane: $3\nu_x = k$,$4\nu_x = k \ldots$
  - Coupled resonances: orbital and spin
    - Orbital: $2\nu_x + \nu_y = k$, etc ...
    - Spin : $\nu_x + \nu_s = k$, etc ... (always linear in spin)
- Spiraling sinks in the case of radiation:
  - Orbital (+ spin)
  - Orbital + stochastic moments: Synchrotron integral theory