Weekly Meeting

Aug. 30th, 2018

Overview:

RDT's are lattice dependent:

$$h_{jklm} = c\sum_{i=1}^{N} S_2 \beta_{xi}^{(j+k)/2} \beta_{yi}^{(l+m)/2} e^{i[(j-k)\mu_{xi}+(l-m)\mu_{yi}]}$$

The plan:

1) Just make up a (reasonable number) − the x, px dependence is more important
2) Construct the map
3) Track the map and study the phase space

The full RDT is:

$$h^{(1)} \equiv \sum_{|\overline{I}|=n} h_{\overline{I}} h_x^{+i_1} h_x^{-i_2} h_y^{+i_3} h_y^{-i_4} \delta^{i_5}$$

With eigenvectors:

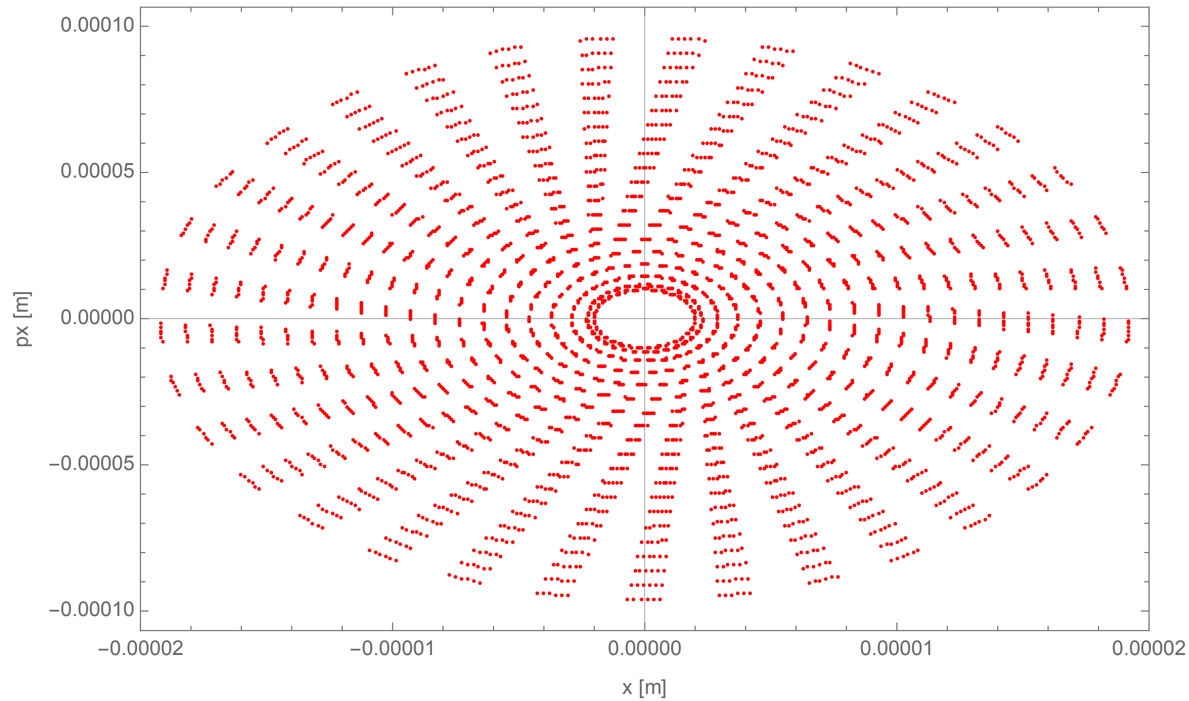
$$h_x^{\pm} \equiv \sqrt{2J_x} e^{\pm i\phi_x} = \sqrt{2J_x}\cos\phi_x \pm i\sqrt{2J_x}\sin\phi_x = x \mp ip_x$$

Progress

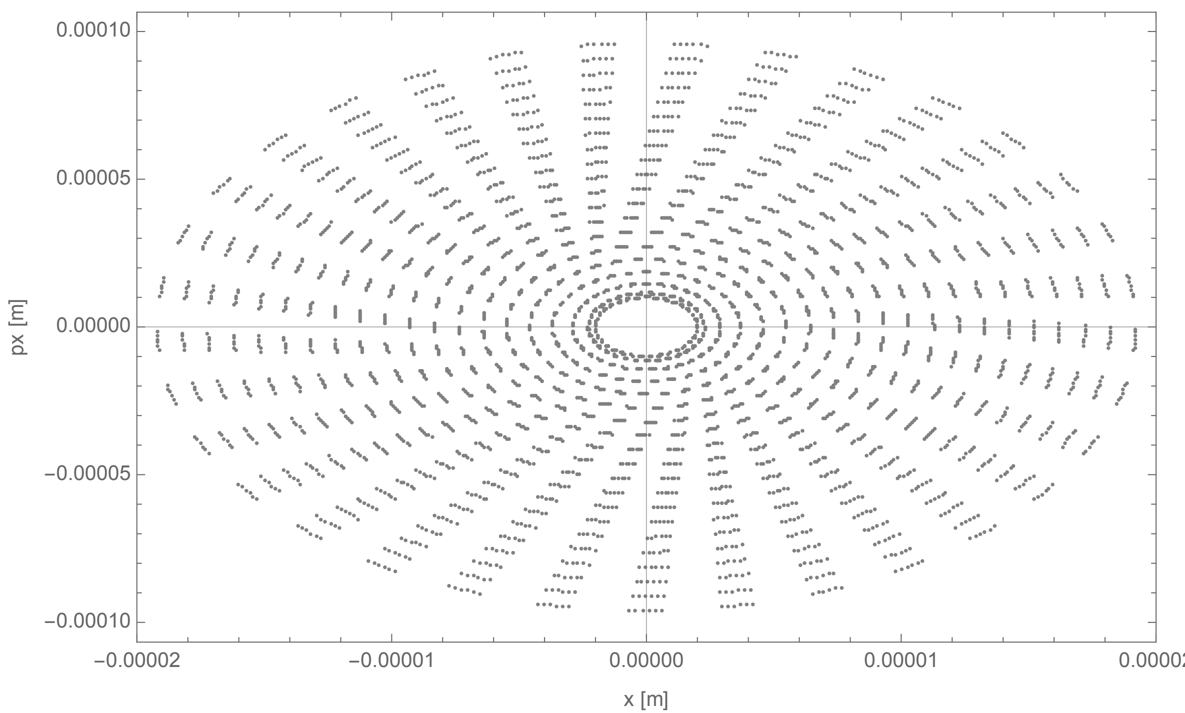
```
map[{xn_, pn_}] :=
 Module[{AMat, AInv, RMat, hxp, hxm, finalMap},
  hxp = Evaluate[x + I p];
  hxm = Evaluate[x - I p];
  AMat = {{√β, 0}, {-α/√β, 1/√β}};
  AInv = {{1/√β, 0}, {α/√β, √β}};
  RMat = {{Cos[μ], Sin[μ]}, {-Sin[μ], Cos[μ]}};
  finalMap =
   AInv.((RMat.AMat.{xn, pn}) + PoissonBracket[h*δ^n*hxp^j*hxm^k, RMat.AMat.{xn, pn}, x, p] +
      1/2 PoissonBracket[h*δ^n*hxp^j*hxm^k, PoissonBracket[h*δ^n*hxp^j*hxm^k,
         RMat.AMat.{xn, pn}, x, p], x, p])]
```

- Truncated to 2^{nd} order in the Lie Transformation
- Takes un-normalized coordinates and returns un-normalized coordinates

```
j = 1;
k = 1;
n = 1;
δ = 10^-3;
μ = 0.19;
β = 5;
α = 0;
h = -2.5;
iter = 200;
xInit = 0.01 * 10^-5;
xEnd  = 2 * 10^-5;
stepSize = 10^-6;
pInit  = 10^-5;
```




```
j = 1;
k = 1;
n = 1;
δ = 10^-3;
μ = 0.19;
β = 5;
α = 0;
h = -200.5;
iter = 200;
xInit = 0.01 * 10^-5;
xEnd  = 2 * 10^-5;
stepSize = 10^-6;
pInit  = 10^-5;
```



Something is wrong!

- Doesn't change with δ (energy spread) or h (RDT coefficient) as it should!
- Only changes for different μ (phase advance)

To solve this equation h has to be decomposed into two parts: one part independent of the angle variables[34] and the remaining

$$h^{(1)} = h^{(1)}_{\mathrm{Ker}}\left(\overline{J}\right) + h^{(1)}_{\mathrm{Im}}\left(\overline{J},\overline{\phi}\right) \tag{79}$$

so that

$$k^{(1)} = h^{(1)}_{\mathrm{Ker}}\left(\overline{J}\right), \qquad g^{(1)} = -\frac{1}{1-\mathcal{R}_{0\to n}} h^{(1)}_{\mathrm{Im}}\left(\overline{J},\overline{\phi}\right) \tag{80}$$

which leads to

$$\begin{aligned} e^{:h:}\mathcal{R}_{0\to n} &= e^{:-g\left(\overline{J},\overline{\phi}\right):} e^{:k\left(\overline{J}\right):}\mathcal{R}_{0\to n} e^{:g\left(\overline{J},\overline{\phi}\right):} \\ &= e^{:\frac{1}{1-\mathcal{R}_{0\to n}}h^{(1)}_{\mathrm{Im}}\cdots:}\mathcal{R}_{0\to n} e^{:h^{(1)}_{\mathrm{Ker}}+h^{(2)}_{\mathrm{Ker}}-\frac{1}{2}\left[h^{(1)}_{\mathrm{Im}},\frac{1}{1-\mathcal{R}_{0\to n}}h^{(1)}_{\mathrm{Im}}\right]_{\mathrm{Ker}}\cdots:} \\ &\quad \times e^{:-\frac{1}{1-\mathcal{R}_{0\to n}}h^{(1)}_{\mathrm{Im}}\cdots:} \end{aligned} \tag{81}$$

where $\left[\overline{J},\overline{\phi}\right] \equiv [J_x, \phi_x, J_y, \phi_y]$ are the action-angle variables. This can strictly

Synopsis of Meeting:

- Realized that there are two pieces: the code is only calculating the linear part of the map
- There will be issues calculating the imaginary part via the current formulation of the code, so Stas and I will reforumalate.