SAD

K. Oide

KEK, High Energy Accelerator Research Organization

Oho, Tsukuba, Ibaraki 305, Japan

July 7, 1998

Topics:

- Overview
- Elements and Keywords
- Beam line
- Transformation
- Optical functions
- Matching
- Extension of SAD

Elements and Keywords

Definition Syntax:

Main Level:

$$\begin{aligned} \textit{element\_type element} &= (\textit{keyword} = \textit{expr}. \ \ ...) \\ \textit{element1} &= (\textit{keyword} = \textit{expr}. \ \ ...) \\ ... &\ \ ; \end{aligned}$$

Example:

```
QUAD QF1 = (L = 0.5 K1 = 0.1)
     QD1 = (L = 0.5 K1 = -0.1);
```

Function:

`SetElement[`*element*, *type*, { *keyword*`->`*value*, ...}`]`

Example:

```
SetElement["QF1", "QUAD", {"L"->0.5, "K1"->0.1}];
```

element		keywords
APERT	aperture	COUPLE DP DX1 DX2 DY1 DY2 JDPX JDPY
BEAMBEAM	beam-beam	AX AY AZ BX BY COUPLE DP DPX DPY DX DY DZ EMITX EMITY EPX EPY EX EY NP R1 R11 R12 R13 R14 R15 R16 R2 R22 R23 R24 R25 R26 R3 R33 R34 R35 R36 R4 R44 R45 R46 R55 R56 R66 SIGZ SLICE STURN XANGLE ZPX ZPY ZX ZY
BEND	dipole	ANGLE COUPLE DISFRIN DISRAD DX DY E1 E2 EPS F1 FRINGE K0 K1 L RANKICK ROTATE
CAVI	rf cavity	COUPLE DPHI DX DY FREQ HARM L LWAKE PHI RANPHASE RANVOLT ROTATE TWAKE V02 V1 V11 V20 VOLT
COORD	coordinate transformation	CHI1 CHI2 CHI3 COUPLE DIR DX DY DZ
DECA	decapole	COUPLE DISFRIN DISRAD DX DY K4 L ROTATE
DODECA	dodecapole	COUPLE DISFRIN DISRAD DX DY K5 L ROTATE

Table 1: Keywords of SAD elements.

element		keywords
DRIFT	drift space	COUPLE DISKIN L RADIUS
INS	insertion	AX AY BX BY COUPLE DIR DPX DPY DX DY EPX EPY EX EY PSIX PSIY R1 R2 R3 R4
MAP	external map	COUPLE
MARK	marker	AX AY AZ BX BY COUPLE DDP DP DPX DPY DX DY DZ EMITX EMITY EPX EPY EX EY GEO JDPX JDPY JDPZ JDX JDY JDZ OFFSET PSIX PSIY R1 R2 R3 R4 SIGZ
MONI	monitor	COUPLE DX DY OFFSET ROTATE
MULT	universal multipole	CHI1 CHI2 COUPLE DISFRIN DISRAD DPHI DX DY DZ EPS F1 F2 FREQ FRINGE HARM K0 K1 K10 K11 K12 K13 K14 K15 K16 K17 K18 K19 K2 K20 K21 K3 K4 K5 K6 K7 K8 K9 L PHI RADIUS ROTATE SK0 SK10 SK11 SK12 SK13 SK14 SK15 SK16 SK17 SK18 SK19 SK2 SK20 SK21 SK3 SK4 SK5 SK6 SK7 SK8 SK9 VOLT W1

Table 2: Keywords of SAD elements (cont'd).

`OCT`	octupole	`COUPLE DISFRIN DISRAD DX DY K3 L ROTATE`
`PHSROT`	phase space rotation	`AX AY AZ B11 B12 B13 B14 B15 B16 B22 B23 B24 B25 B26 B33 B34 B35 B36 B44 B45 B46 B55 B56 B66 BX BY BZ COUPLE D11 D12 D13 D14 D15 D16 D21 D22 D23 D24 D25 D26 D31 D32 D33 D34 D35 D36 D41 D42 D43 D44 D45 D46 D51 D52 D53 D54 D55 D56 D61 D62 D63 D64 D65 D66 DP DZ EMITX EMITY EMITZ EPX EPY EX EY JDPY JDY PSIX PSIY PSIZ R1 R2 R3 R4 SIGZ ZPX ZPY ZX ZY`
`QUAD`	quadrupole	`ACHROMA COUPLE DISFRIN DISKIN DISRAD DX DY EPS F1 F2 FRINGE K1 L ROTATE`
`SEXT`	sextupole	`COUPLE DISFRIN DISRAD DX DY K2 L ROTATE`
`SOL`	solenoid	`BOUND BZ CHI1 CHI2 CHI3 COUPLE DBZ DPX DPY DX DY DZ F1 GEO L`
`TCAVI`	transverse cavity	`COUPLE DX DY FREQ HARM K0 L LWAKE PHI RANKICK RANPHASE ROTATE TWAKE`

Table 3: Keywords of SAD elements (cont'd).

overlapped element

In the real world, many elements are placed overlapping to each other. For instance,

- quadrupoles in nonuniform solenoid (e.g. Belle & QCS).
- Quads, dipoles, solenoids on accelerating structure (e.g. Linac).

Though these components can be expressed using `SOL` and `MULT` in the current version of SAD, they are uneasy to handle.

A `BEND` element with "multipoles" or acceleration is not possible to express yet. Even it is not impossible to define "multipoles" in the curved coordinate, but it will be impractical to use such quantity for magnet measurements which are usually done in Cartesian system.

Beam Line

Definition Syntax:

Main Level:

`LINE` *beamline* `= (`*element1, element2, ...*`);`

Example:

```
LINE L1 = (START QF1 QD1);
```

Function:

`BeamLine[`*element1, element2, ...*`]`

Example:

```
l = BeamLine["START", "QF1", "QD1"];
FFS["USE l"];
```

construction of beam line

A beam line is a series of elements.

- Elements are appended to the previous one, with the local coordinate at the exit of the previous element.
- A `BEND` element rotates the local coordinate accroding to its value of `ANGLE`.
- A general coordinate transformation is possible by `COORD` element.

local coordinate

- The local coordinate is a right-hand system.
- The s-axis points the direction of the beam line.
- A `BEND` element rotates the local coordinate around the y-axis by $-$`ANGLE`, when `ROTATE` is zero.
- For any elements, the keyword `ROTATE` rotates the element (and the local coordinate) around the local s-axis by $-$`ROTATE` at the entrance, and rotates back at the exit.

- The rotation is done after taking out the offset given by (`DX`, `DY`) at the entrance, and before resetting the offset at the exit.

- At the entrance of `SOL` the coordinate is automatically set to the axis of `SOL`. At the exit it resets to the design orbit. In both cases, The angle χ_3 (see below) is set to zero after the transformation.

geometry coordinate

The relation between the local coordinate (x, y, s) at each element and the global geometric coordinate (ξ, η, ζ) is shown by `DISPLAY GEOMETRY` (abbrev. `DISP G`) command.

- The global coordinate defaults its origin at the beginning of the beam line, and the axes are $(\xi, \eta, \zeta) = (s, -x, -y)$.

- The global coordinate can be changed by `ORG` command.

The rotation of the local coordinate is expressed by three angles as shown in Fig. 1.

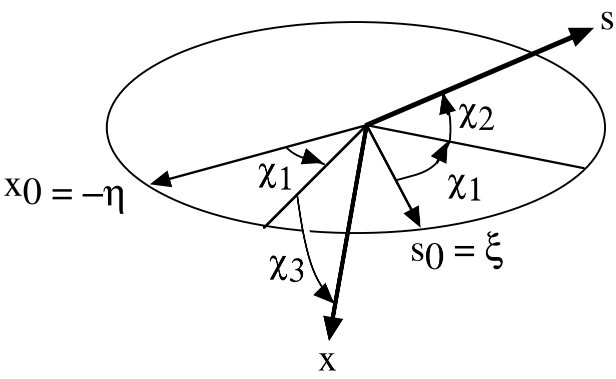


Figure 1: Rotation of the local coordinate is expressed by angles χ_1, χ_2, and χ_3.

coordinate and orbit

- The coordinate and the orbit are different things.
- In usual cases the coordinate is placed on the design orbit, but they become different by using `SOL`, misaligned elements, elements with `K0` as the "design", or `COORD` elements.
- To avoid confusion, there is a flag `GEOFIX` (default: OFF).

- When `GEOFIX` is ON, the coordinate is fixed by changing alignment, etc.
- The design momentum $p_0(s)$ works as a part of the coordinate system.
- `GEOFIX` also fixes $p_0(s)$. This is important in a linac.

Transformation

Different transformations are used in TRACK, EMIT, and FFS. FFS uses the same routines for orbit and matrix calculation as EMIT's, but uses its 4 by 5 submatrix for the optics parametrization.

	TRACK	EMIT	FFS
orbit tracking	6D symplectic	6D symplectic	6D symplectic
matrix	–	6D	4 by 5
radiation loss	when RAD	when RADCOD	–
rad. diffusion	classical(TRPT), Gaussian(RING)	as beam matrix	–
acceleration	OK	OK	OK
wake field	(obsolete)	–	OK
space charge	static approx.	–	–
intrabeam	–	beam matrix	–

Table 4: Comparison of transformations in TRACK, EMIT, and FFS.

DRIFT

Transformation in DRIFT is done analytically (without parallel or ultra-relativistic approximations).

BEND

- The body of `BEND` is treated analytically even with the edge angles, when `K1` is zero.
- The non-analytic part consists of the linear fringe (leak of the field from the edge), non-linear fringe at the first order, and field index (`K1`).

The entire transformation is:

```
(drift to the entrance face)
  x2  = x1/(cos(psi1) - sin(psi1) (px1/pz1))
  px2 = px1 cos(psi1) + pz1 sin(psi1)
  y2  = y1 + (py1/pz1) x2 sin(psi1)
  z2  = z1 - (p1 /pz1) x2 sin(psi1) ,
  where psi1 = ANGLE * E1;

(linear fringe at entrance face)
  x2  = x1 + dxfr (p1 - p0)/p1
  py2 = py1 + dyfr y1/p1^2
  z2  = z1 + (dxfr px1 + dyfr y1^2/(2 p1))/p1
  where dxfr = F1^2/(24 rhob) ,
        dyfr = F1/(6 rhob^2) ,
        rhob = L'/(ANGLE + K0) ,
        L'   = L - (ANGLE F1)^2 /(24 L)
```

```
            * sin(ANGLE (1 - E1 - E2)/2)/sin(ANGLE/2)

(nonlinear fringe at entrance)
  x2  = x1  + y1^2 p1^2/(2 rhob (p1^2 - px1^2)^(3/2))
  py2 = py1 - px1 y1/(p1 rhob sqrt(p1^2 - px1^2))
  z2  = z1  - px1 y1^2 p1/(2 rhob (p1^2 - px1^2)^(3/2))

(body of bend)
  px2 = -rho0/rhob (sin(psi2) + sin(omega + psi1))
        + sin(omega) pz1 + cos(omega) px1
        - x1/rhob sin(omega)
  x2  = x1 cos(omega)
        + rhob (pz2 - cos(omega) pz1 + sin(omega) px1)
        + rho0 (cos(omega+psi1) - cos(psi2))
  y2  = y1 + py1/sqrt(p1^2 - py1^2) s
  z2  = z1 - s p1/sqrt(p1^2 - py1^2) + v1/v0 L'
  where rho0  = L'/ANGLE
        omega = ANGLE - psi1 - psi2
        s     = rhob ANGLE (arcsin(px1/sqrt(p1^2 - py1^2))
                - arcsin(px2/sqrt(p2^2 - py2^2)) + omega)

(nonlinear fringe at exit)
  x2  = x1  - y1^2 p1^2/(2 rhob (p1^2 - px1^2)^(3/2))
  py2 = py1 + px1 y1/(p1 rhob sqrt(p1^2 - px1^2))
  z2  = z1  + px1 y1^2 p1/(2 rhob (p1^2 - px1^2)^(3/2))
```

```
(linear fringe at entrance face)
  x2  = x1 - dxfr (p1 - p0)/p1
  py2 = py1 + dyfr y1/p1^2
  z2  = z1 + (-dxfr px1 + dyfr y1^2/(2 p1))/p1

(drift from the exit face)
  px2 =  cos(psi2) px1 + sin(psi2) pz1
  x2  = x1 (cos(psi2) + px2/pz2 sin(psi2))
  y2  = y1 + py2/pz2 x1 sin(psi2)
  z2  = z1 - x1 sin(psi2) p2/pz2
  where psi2 = ANGLE * E2;
```

If `K1` is nonzero, the effects from `E1` and `E2` are approximated by thin quadrupoles. Then the body is subdivided into

```
1 + Floor[Sqrt[Abs[K1 L']/(12 10^-5 EPS)]]
```

slices

QUAD

- As there is no analytical solution of the transformation for the body of a quadrupole, SAD splits the Hamiltonian into the linear and residual nonlinear parts.
- A quadrupole body is sliced, and for each slice, the linear part is tracked analytically, and the residual part is applied as a kick:

```
(nonlinear fringe at entrance)
   canonical transformation by a generating function

G(x1, px2, y1, py2, p1)
     = H0(x1, px2, y1, py2, p1)
     + (D[H0, x1] D[H0, px2] + D[H0, y1] D[H0, py2])/2

   where H0  =  px2 dx1 + py2 dy1
         dx1 =  x1 (a/3 + b)
         dy1 = -y1 (a + b/3)
         a   =  K1 x1^2/p1/4
         b   =  K1 y1^2/p1/4 .

(linear fringe at entrance)
   px2 = exp(-a) px1
   py2 = exp(a)  py1
```

```
   x2  = exp(a)  x1 + b px1
   y2  = exp(-a) y1 - b py1
   z2  = z1 - (a x1 + b (1 + a/2) px2) px1
            + (a y1 + b (1 - a/2) py2) py1
   where a = -K1 F1 abs(F1)/(24 p1 L)
         b =  K1 F2/L .
```

F1 and F2 are parameters to characterize the slope of the field at the edges defined as:

```
     F1 = SIGN(Sqrt[a],a),    a = 24(I_0^2/2 - I_1),
     F2 = I_2 - I_0^3/3
with
     I_n = Integrate[(s-s0)^n K1[s]/K1_0,
             {s,-Infinity,Infinity}],
```

(body of quad)

The body is subdivided in

```
   n = 1 + Floor[10 Abs[(K1 L)/EPS]
   (EPS = 1 is used when EPS = 0),
```

then a transversely linear transformation exp(:H:) is done in each slice with

```
   H = ((-p + (px^2 + py^2)/(2 p) + E/v0) L
       + K1 (x^2 - y^2)/2)/n .
```

```
   Between slices applied is the correction exp(:dH:)
   for the kinematical term with

   dH = (-sqrt(p^2 - px^2 -py^2) + p
        - (px^2 + py^2)/(2 p)) L/n .

   In a solenoid, the forms of H and dH are modified.

(linear fringe at exit)
   px2 = exp( a)  px1
   py2 = exp(-a) py1
   x2  = exp(-a) x1 + b px1
   y2  = exp( a) y1 - b py1
   z2  = z1 + (a x1 - b (1 - a/2) px2) px1
            - (a y1 - b (1 + a/2) py2) py1
   where a = -K1 F1 abs(F1)/(24 p1 L)
         b =  K1 F2/L .

(nonlinear fringe at exit)
   canonical transformation by a generating function

   G(x1, px2, y1, py2, p1)
        = H0(x1, px2, y1, py2, p1)
        + (D[H0, x1] D[H0, px2] + D[H0, y1] D[H0, py2])/2
```

```
where H0   =  px2 dx1 + py2 dy1
      dx1  =  x1 (a/3 + b)
      dy1  = -y1 (a + b/3)
      a    = -K1 x1^2/p1/4
      b    = -K1 y1^2/p1/4 .
```

SEXT, OCT, DECA, DODECA

The transformation in a 2(n+1)-pole is given as

```
exp(:Fin:)exp(:a L:)exp(:Hn/2:)exp(:b L:)
*exp(:Vn:)exp(:a L:)exp(:Hn/2:)exp(:b L:)exp(:Fout:) ,
```

where L and Hn are Hamiltonians of a drift of length L and a thin 2(n+1)-pole kick of integrated strength Kn:

```
Hn = Kn/(1+n)! Re((x - I y)^(1+n)) ,
```

respectively. The coeffients are a = 1/2 - 1/sqrt(12) and b = 1/2 - a.

Terms exp(:Fin:) and exp(:Fout:) are transformations for

```
entrance and exit nonlinear fringes.

The term exp(:Vn:) is a correction to adjust
the third-order terms in L:

   Vn = (SUM over j=(x,y), k=(x,y)) [
              - beta/2 (Hn,k)^2
              + gamma (Hn,j Hn,k Hn,j,k)] ,

where ,i represents the derivertive by x or y.

We have also introduces two
coefficients beta = 1/6 - 1/sqrt(48) and
gamma = 1/40 - 1/24/sqrt(3).
```

CAVI

- `CAVI` simulates an accelerating structure. It is basically a thin acceleration. When its length `L` is specified, `CAVI` is sliced into pieces, consisting drifts and thin accelerations.
- It does not represent any realistic field pattern or "rf fringe field" .

MULT

- MULT is a universal element to express an overlapped elements with multipoles and acceleration.
- The basic idea of the transformation is same as QUAD: Divide the body into slices, solve linear term analytically, correct nonlinear by kicks.

SOL

- The basic characterlistics of the transformation of elements above are applicable when an element is placed within a slolenoid field, if the body of the solenoid field is constant.
- Since SOL only accepts a constant BZ, when the solenoid field is non-uniform in s, one have to prepare a deck with many solenoids.
- The fringe field of the solenoid is applied automatically, as the continuity of the canonical momenta.

Optical Functions

In FFS, optics are represented by 20 optical functions listed in Table 5. Note that FFS calculates only 4 by 5 optics.

function		function	
AX	α_X	AY	α_X
BX	β_X	BY	β_Y
NX	ψ_X	NY	ψ_Y
EX	η_X	EY	η_Y
EPX	η_{PX}	EPY	η_{PY}
R1	r_1	R2	r_2
R3	r_3	R4	r_4
DX	x	DY	y
DPX	p_x	DPY	p_y
DZ	z	DDP	Δp

Table 5: Optical functions in FFS. The notation assmes the momenta $(p_x, p_y, \Delta p)$ to be normalized by the local design momentum $p_0(s)$.

The transformation from the physical coordinate to the normal coordinate is given by

$$\begin{pmatrix} X \\ P_X \\ Y \\ P_Y \end{pmatrix} = \begin{pmatrix} \mu & 0 & -r_4 & r_2 \\ 0 & \mu & r_3 & -r_1 \\ r_1 & r_2 & \mu & 0 \\ r_3 & r_4 & 0 & \mu \end{pmatrix} \begin{pmatrix} x \\ p_x \\ y \\ p_y \end{pmatrix} - \begin{pmatrix} \eta_X \\ \eta_{PX} \\ \eta_Y \\ \eta_{PY} \end{pmatrix} \Delta p \ , \tag{1}$$

where $\mu^2 + (r_1 r_4 - r_2 r_3) = 1$.

acceleration

When the design coordinate involves acceleration such as in a linac, the parametrization is done for a scaled coordinate:

$$\left(x/\sqrt{\beta\gamma(s)}, p_x\sqrt{\beta\gamma(s)}, y/\sqrt{\beta\gamma(s)}, p_y\sqrt{\beta\gamma(s)}\right)$$

where $\beta\gamma(s) = p_0(s)/(mc)$. Note that above is still a symplectic variables. The resulting Twiss parameter gives the usual relation:

$$\langle x(s)^2 \rangle = \beta_x(s)\varepsilon_x(s) \ , \ \text{etc.},$$

being $\varepsilon_x(s)$ the physical emittance at s.

physical dispersion

The dispersion functions in Eq. 1 are dispersion in the normal coordinate. Sometimes the physical dispersions

$$\begin{pmatrix} \eta_x \\ \eta_{px} \\ \eta_y \\ \eta_{py} \end{pmatrix} \equiv \begin{pmatrix} \mu & 0 & r_4 & -r_2 \\ 0 & \mu & -r_3 & r_1 \\ -r_1 & -r_2 & \mu & 0 \\ -r_3 & -r_4 & 0 & \mu \end{pmatrix} \begin{pmatrix} \eta_X \\ \eta_{PX} \\ \eta_Y \\ \eta_{PY} \end{pmatrix}$$

are more convenient. The physical dispresions are denoted by `PEX`, `PEPX`, `PEY`, `PEPY`, respectively.

Matching

Matching of optics by SAD/FFS has the following characterintics:

- Using multi dimension, multi variable Newton's method with Singular Value Decomposition (SVD) as the main method, supplemented by the steepest descent method.
- approprate choice of functions. For instance, matches $\log \beta_x$ instead of β_x.
- matches geometry of a beam line together with optical functions.
- fuzzy logic to determine the local minimum and switching the methods.

- off-momentum matching.
- finite-amplitude matching.
- boosted by various SADScript functions.

Newton's method with SVD

- For mathching functions f_i and variables x_k, solve

$$\Delta f_i = \sum_k \frac{\partial f_i}{\partial x_k} \Delta x_k \ , \tag{2}$$

  using SVD.
- Search the minimum along the vector Δx_k using prediction with cubic interpolation.
- The derivatives are obtained either analytically or numerically.

SADScript functions used in matching

Mathing by FFS has become more powerful by using various SADScript functions:

name	purpose
`ElementValues`	to specify dependences beween variables
`FitFunction`	to match any number of any function
`FitValue`	to change the goal; to set minimum or maximum of the function.
`FitWeight`	to change the weight of functions
`InitialOrbits`	to set the initial condition of many orbits
`MatchingAmplitude`	finite-amplitude matching
`OpticsEpilog`	to do additional task after calculation
`OpticsProlog`	to do additional task before calculation
`VariableRange`	to set the range of variables

Table 6: SADScript functions for matching.

off-momentum matching

Off-momentum matching is the method of chromaticity correction in SAD.

- If a matching condition is give as

  $$\textit{function value } n \ ,$$

  matching is done for $n = 2m + 1$ off-momentum points

  $$\Delta p = \texttt{DP0} + \texttt{DP}\ k/m \qquad (k = -m, m) \ , \tag{3}$$

  when n is odd.

- When $n = 2m$ is even, the off-momenta are same as the case $n = 2m + 1$, Eq. 3, but the $k = 0$ is excluded.

- The function `FitValue` can change the goal value of matching for each momentum.

- FFS uses no perturbation to calculate the off-momentum optics.

finite-amplitude matching

Finite-amplitude matching is an extension of off-momentum matching to the transverse phase space.

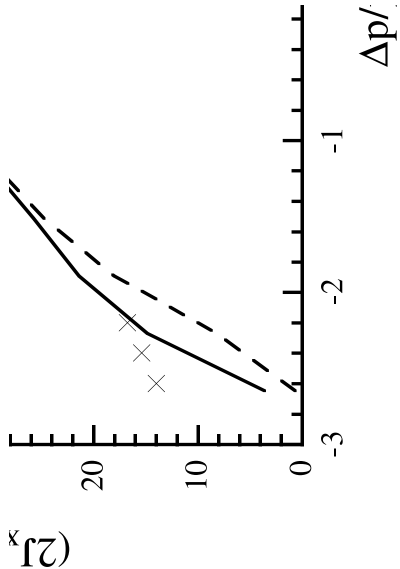
$$\texttt{MatchingAmplitude} = \{\{\Delta p_1, n_x, n_y\}, ...\};$$

sets matching conditions for the orbits on $\Delta p = \Delta p_1$, with initial offset

$$(x, p_x, y, p_y) = \left\{ \begin{array}{c} (x_k \cos\phi_x, x_k \sin\phi_x, 0, 0) \\ (0, 0, y_k \cos\phi_y, y_k \sin\phi_y) \end{array} \right. , \quad (4)$$

where $\phi_{x,y} = (0, 2\pi/3, 4\pi/3)$ and $(x_k, y_k) = (n_x, n_y)\sqrt{2\beta_{x,y}(\varepsilon_x + \varepsilon_y)}$.

- The orbits with the initial offsets never close at the end of the ring, but it is just ignored.
- x-y coupled initial conditions can be given by `Initial Orbit`.


Δp/
-1
-2
-3
0
10
20
(2J

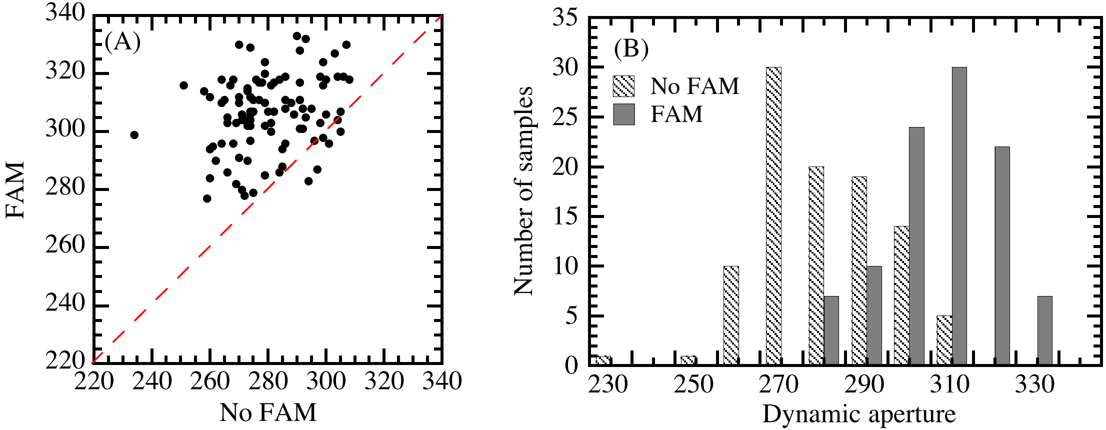


Figure 3: (A): Comparison of the dynamic aperture before/after FAM for 100 samples of sextupole settings. (B): Distribution of the dynamic apertures (Oide, Koiso, Ohmi, 1996).

Extension of SAD

There are several ways to extend SAD for match one's needs. What follows are list of them, from easier to harder.

- Write your own SADScript functions. This is the easiest unless you need very fast simulation.
- If you need hard simulation, but if the interaction between SAD is small, write an interface to your code in SADScript. This is easy, too. This was done for DA Taylor map and E. Forest's code, or to import results of TRANSPORT. It would be also done to revitalize SODOM.
- Write a new compiled function for SADScript. This is hard, but the rules are not so many.
- Add a new element for SAD. You have to write different routines for tracking, emittance, and matching.

These difficulties will be solved in various ways, hopefull not much far from now.