Symplectic integration and re-splitting in PTC

PTC generally attempts to integrate the maps of each magnet using a symplectic integrator. There are two exceptions at the moment to this: the traveling wave cavity used in linacs and the pancake type for fitted B-fields. The list could be extended (Exact wigglers and exact helical dipoles, exact solenoids, etc…) but this is where it stands now.

So the present discussion will be restricted to elements of PTC amenable to explicit symplectic integration.

The Talman philosophy

It was believed in the early days of "kick" codes that the drift-kick-drift (d-k-d) kick model was massively inadequate because it requires a large number of steps to achieve decent convergence and thus the correct tunes. Such codes were restricted to special applications such radiation and spin calculations--- Chao's code SLIM.

The reason for this state of affair was two-fold. Firstly, a technical reason: we did not have high order symplectic integrators. Secondly a philosophical reason: people did not understand their very own approaches which contradicted the inadequacy of kick codes.

The technical issue was resolved by Ruth and later by a cabal of authors including Neri, Forest and finally Yoshida. We can use high order symplectic integrators on our usual "MAD8" Hamiltonian for the body of the magnet.

Almost simultaneously, the code TEAPOT emerged from the suffocating entrails of the SSC-CDG. TEAPOT integrated exactly using a d-k-d method, or more accurately, a PROT-KICK-PROT method the combined function S-Bend. It was a second order code restricted to one step of integration or 4 strange steps for the IR quadrupoles! (PROT is a drift in polar coordinates used in S-Bend integration and in patching, see our discussion

on patching.),

How could that work? Talman, the principal person pushing TEAPOT, realized quite correctly that in accelerator physics the integration step is part of the model. For example, people who objected to the D-K-D integration method would themselves always use a single kick for the sextupoles. They would then adjust chromaticities not based on the thick sextupole model but based on their semi-serious one-kick representation. It is remarkable that the results are for most machines acceptable.

Talman realized that if he fitted each cell of the SSC to exactly 90 degree, the results with one kick per quadrupole, would be nearly identical to that of a "matrix" code. Better even, Talman's code any , once the magnets are refitted to achieve a correct machine, could potentially produce the correct physics for small rings while the standard matrix-kick-matrix" codes could not as they relied on the small angle approximation.

The Talman philosophy of symplectic integration can be summarized by giving an actual pseudo-algorithm for its implementation in the D-K-D case. We will refer to this as the Talman (pseudo-)algorithm:

a) Cut the lattice using some method
b) Fit all the stuff you would normally fit using your matching routines
c) Examine the resulting lattice functions and also perhaps some short term dynamic aperture
d) If all is fine, reduce the number of cuts and/or the sophistication of the method and go back to 1.
e) If something sucks, increase the number of cuts and/or the sophistication of the method and go back to 1.
f) After a having oscillated between d) and e), make up your mind and call that "the lattice."

For your particular lattice, store all the information at step f) so that you do not have to do that again!

In this note, we describe step a) in detail. The other steps are too "user"

dependent.

Recutting the lattice

THIN_LENS_RESTART(R1*,FIB,USEKNOB*)
This is routine resets all magnets to second order integration with one step.

THIN_LENS_RESPLIT(R*,THIN,EVEN,LIM(1:2),LMAX0,XBEND,FIB,USEKNOB*)

The optional parameter FIB and USEKNOB are explained later. They permit us to restrict the action of THIN_LENS_RESTART and THIN_LENS_RESPLIT to specific fibres or group of magnets.

Global parameters :
1) resplit_cutting (0,1, or 2)
2) sixtrack_compatible (true or false)
3) radiation_bend_split (true or false)
4) fuzzy_split (default to 1.0)

The parameter THIN describes an approximate integrated quadrupole strength for which a single thin lens should be used. For example, the example associated with this document, the quadrupoles QF and QD have an integrated strength of

KF L = 0.279309637578521 KD L=-0.197159744173073

Now let make following call

```
CALL MOVE_TO(R1,QF,"QF",POS)
CALL MOVE_TO(R1,QD,"QD",POS)
THIN=0.01D0
LIMITS(1:2)=100000

CALL THIN_LENS_RESPLIT(R1,THIN,LIM=LIMITS)
```

```
WRITE(6,*) QF%MAG%NAME,QF%MAG%P%METHOD,QF%MAG%P%NST
WRITE(6,*) QD%MAG%NAME,QD%MAG%P%METHOD,QD%MAG%P%NST
```

The results of example 1 are :

```
QF                              2           27
QD                              2           19
B                               2           15
```

We notice that KF/THIN is about 27 and KD/THIN is about 19. So this behaves as expected: it splits according to quadrupole strength. The method stayed method=2, i.e., second order integrator. This is not efficient if the number of steps must be large. Therefore we must teach the respliting algorithm to switch to the 4^{th} and 6^{th} integrator. This is now explained.

In the bend, we also look at the amount of natural horizontal and/or vertical focusing. In this example, a small ring with 36 degrees bend, the bends are unusually strong. Other parameter can be used to affect the bends: radiation_bend_split and xbend. They are discussed later.

The lim optional parameter:

PTC has, in the D-K-D case, 3 methods of integrations: the naïve second order method, the Ruth-Neri-Yoshida 4th order method and the Yoshida 6^{th} order method.
The optional array lim(2) determines when PTC should choose 4^{th} or 6^{th} order method.

1) lim(1) >KL/THIN : 2^{nd} order method
2) lim(2)>KL/THIN>lim(1): 4^{th} order method.
3) lim(2)>KL/THIN>lim(1), then PTC choose the 4^{th} order mether

Let us rerun the previous case with a different limit array:

```
THIN=0.01D0
LIMITS(1)=8
LIMITS(2)=24
```

```
CALL THIN_LENS_RESPLIT(R1,THIN,LIM=LIMITS)
WRITE(6,*) QF%MAG%NAME,QF%MAG%P%METHOD,QF%MAG%P%NST
WRITE(6,*) QD%MAG%NAME,QD%MAG%P%METHOD,QD%MAG%P%NST
```

The results of example 2 are :

QF	**6**	**3**
QD	**4**	**6**
B	**4**	**5**

The number of thin lenses for method 6 is 7 per steps. Method 4 has 3 kicks per steps. So we can see the number of kicks has remained more or less constant while the method has increased in order.

An example of the Talman Algorithm

```
!!!! TALMAN ALGORITHM !!!!! !!!!!  EXAMPLE 3
WRITE(6,*) "  "
WRITE(6,*) "!!!! TALMAN ALGORITHM !!!!! !!!!!  EXAMPLE 3"
WRITE(6,*) "  "

! PART 1A
THIN=0.001D0   ! CUT LIKE CRAZY
CALL THIN_LENS_RESPLIT(R1,THIN,LIM=LIMITS)
!!! PTC COMMAND FILE: COULD BE A MAD-X COMMAND OR WHATEVER
! COMPUTING TUNE AND BETA AROUND THE RING
! PART 1B
CALL READ_PTC_COMMAND77("FILL_BETA0.TXT")
WRITE(6,*) "  "
WRITE(6,*) " NOW REDUCING THE NUMBER OF STEPS AND REFITTING "
WRITE(6,*) "  "
! PART 2
DO I=0,2
 THIN=0.01D0+I*0.03
! PART 2A
 CALL THIN_LENS_RESPLIT(R1,THIN,LIM=LIMITS) ! REDUCING NUMBER OF CUTS
```

```
 ! FITTING TO PREVIOUS TUNES
! PART 2B
 CALL READ_PTC_COMMAND77("FIT_TO_BETA0_RESULTS.TXT")
 ! COMPUTING DBETA/BETA AROUND THE RING
! PART 2C
 CALL READ_PTC_COMMAND77("FILL_BETA1.TXT")
ENDDO
```

Now these are the result of part 1A:

```
METHOD 2          40         40
METHOD 4           0          0
METHOD 6          30       6230
number of Slices        6270
Total NST         930
Total NST due to Bend Closed Orbit          0
Biggest ds   0.115885454545455
```

The quadrupoles and the dipoles are cut using method 6 with a grand total of 6240 cuts.

In part 1B then compute the tunes and betas around the ring. The fractional tunes are:

```
Tunes   0.142192063715077      0.854078356314425
```

And the betas are stored for future use.

In part 2A, the magnet is recut less and less finely, the tune is refitted and finally the delta beta around the ring is estimated as a measure of the cutting adequacy.

The results are:

Thin = 0.01

```
<DBETA/BETA> =   4.025629639896395E-006
 MAXIMUM OF DBETA/BETA =   6.034509389788590E-006
```

Thin = 0.04

```
<DBETA/BETA> =   2.065365900468319E-003
```

```
 MAXIMUM OF DBETA/BETA =   4.014532098810251E-003
```
Thin = 0.07
```
<DBETA/BETA> =   5.779446473894797E-003
 MAXIMUM OF DBETA/BETA =   1.068563516341058E-002
```

One can also look at the chromaticities. For example, the chromaticities at the beginning were:

Chromaticities = -2.87009384223620 -2.03916389491785
and they are the end:

Chromaticities = -2.86244921970493 -2.03671376872069

At this point, we can freeze the lattice for good. Of course refitting the tune was only an example: in other lattices we may want to rematch a more complete set of properties: dispersion, alphas, phase advances, etc…

Recutting the lattice when exact=.true.: usage of the parameter XBEND

The lattice uses sector bends. Because this is an ideal lattice, the tune with the option exact_model=true or false should lead to the same tune.
Therefore we will fit to the same tune as before, passing it through the same algorithm.

```
WRITE(6,*) "  "
WRITE(6,*) "!!!! SBEND ORBIT SMALL PROBLEM !!!!! !!!!!  EXAMPLE 4"
WRITE(6,*) "  "


CALL APPEND_EMPTY_LAYOUT(M_U)  ! NUMBER 2
CALL SET_UP(M_U%END)
R1=>M_U%END

EXACT=.TRUE.
METHOD=DRIFT_KICK_DRIFT
```

```
CALL RUN_PSR(R1,EXACT,METHOD)

WRITE(6,*) "  "
WRITE(6,*) " NOW REDUCING THE NUMBER OF STEPS AND REFITTING "
WRITE(6,*) "  "
DO I=0,2
 THIN=0.01D0+I*0.03
 CALL THIN_LENS_RESPLIT(R1,THIN,LIM=LIMITS) ! REDUCING NUMBER OF CUTS
 ! FITTING TO PREVIOUS TUNES
  CALL READ_PTC_COMMAND77("FIT_TO_BETA0_RESULTS_2.TXT")
  ! COMPUTING DBETA/BETA AROUND THE RING
CALL READ_PTC_COMMAND77("FILL_BETA1_2.TXT")
ENDDO
```

The results of the last iteration are:

closed orbit

-3.129618316516444E-002 3.357101188909470E-003 0.000000000000000E+000
0.000000000000000E+000 0.000000000000000E+000 0.000000000000000E+000
stability T
Tunes 0.142192063715077 0.854078356314425
<DBETA/BETA> = 1.184251326377263E-002
MAXIMUM OF DBETA/BETA = 2.032348481520253E-002

We notice that the closed orbit is wild and that the fluctuation of the beta functions is twice as big as before.

This problem can be alleviated by enforcing a finer cut of the bend. This is done by setting the parameter xbend to some small value which corresponds approximately to the norm of the residual orbit.

In example 5, the respliting line is replaced by this one:

```
CALL THIN_LENS_RESPLIT(R1,THIN,LIM=LIMITS,XBEND=1.D-4)
```

The results of the final iteration steps are:

closed orbit

-3.377915412576128E-003 3.670253104757456E-004 0.000000000000000E+000
0.000000000000000E+000 0.000000000000000E+000 0.000000000000000E+000
stability T
Tunes 0.142192063715077 0.854078356314425
<DBETA/BETA> = 3.542348342695508E-003
MAXIMUM OF DBETA/BETA = 5.551255781973659E-003

EVEN Keyword (Example 6)

There are 3 possibilities:

EVEN=true enforces even split.
EVEN=false enforces odd split.
EVEN is not at all in the call statement: even or odd split is acceptable

Even=true is important if the motion must be observed at the centre of a magnet. This is useful during matching procedures in particular.

Name	method	#steps	T1%pos	TM%pos	T2%pos
QF	6	3	35	35	41
QD	4	6	52	57	61
B	4	5	67	67	75
QF	6	4	39	43	46
QD	4	6	59	64	68
B	4	6	75	80	84

The red data shows the “indifferent” splitting. T1 is a pointed to the integration node at the beginning of the fibre. TM points to the centre ***if it exists***. And T2 points to the end of the fibre. In the red case, TM and T1 are identical when the number if steps are odd: there is no centre of the magnet and by default TM points to T1.

In the green case, TM is always different from T1 since we insure the existence of a centre using even=true.

The existence of TM permits to point directly at the centre of the magnet.

LMAX0 Keyword (example 7):

```
call move_to(r1,d1,"D1",pos)
call move_to(r1,d1_next,"D1",pos)
call move_to(r1,d2,1) !!! Put the pointer for search back at position 1
call move_to(r1,d2,"D2",pos)
call move_to(r1,qf,"QF",pos)
call move_to(r1,qd,"QD",pos)
call move_to(r1,b,"B",pos)
resplit_cutting=1
call THIN_LENS_restart(r1)
thin=0.01d0
CALL THIN_LENS_resplit(R1,THIN,even=.true.,lmax0=0.05d0,xbend=1.d-4)
```

The lmax0 keyword must be used in conjunction with either resplit_cutting=1 or resplit_cutting=2.
(resplit_cutting is normally defaulted to zero)

If resplit_cutting=1, the ordinary magnets are left as is. However, the drifts are cut so that the maximum length of an integration node cannot exceed lmax0.

The results of the above run are:

Name	method	#steps	T1%pos	TM%pos	T2%pos
D1	2	46	1	26	102
D1	2	46	103	26	152
D2	2	10	57	64	70
QF	6	4	95	99	102
QD	6	2	203	206	208
B	4	6	223	228	232

One notices that only the drifts D1 and D2 were cut.

If resplit_cutting=2, then all the magnets and the drifts are cut to achieve a

maximum length of lmax0. This is useful in space charge calculations.

Name	method	#steps	T1%pos	TM%pos	T2%pos
D1	2	46	1	26	166
D1	2	46	167	26	216
D2	2	10	67	74	80
QF	6	12	151	159	166
QD	6	12	267	275	282
B	4	52	297	325	352

Now all the magnets are split.

Now all the magnets are split.

FIB Keyword (example 8):

Continuing with the previous example, we make the following call:

```
WRITE(6,*) "!!!! FIB KEYWORD  !!!!! !!!!!  EXAMPLE 8"
CALL THIN_LENS_RESTART(R1)
CALL THIN_LENS_RESPLIT(R1,THIN,EVEN=.TRUE.,LMAX0=0.005D0,XBEND=1.D-4,FIB=D1_NEXT)
```

The results are:

Name	method	#steps	T1%pos	TM%pos	T2%pos
D1	2	46	1	26	50
D1	*2*	*458*	*167*	*26*	*628*
D2	2	10	67	74	80
QF	6	12	151	159	166
QD	6	12	679	687	694
B	4	52	709	737	764

Only the fibre D1_NEXT is affected. All the others are left intact. This allows the splitting of a single fibre.

USEKNOB Keyword (example 9):

One can also use the type pol_block to produce a cutting of the lattice. For example:

```
FAM=0

FAM%NAME="D1"

R1=FAM

CALL THIN_LENS_RESTART(R1) ! PUTS BACK METHOD =2 AND NST=1 EVERYWHERE

 CALL THIN_LENS_RESPLIT(R1,THIN,EVEN=.TRUE.,LMAX0=0.005D0,XBEND=1.D-4,USEKNOB=.TRUE.)
```

If useknob is true, then the magnets with the knob flag "on", are split assuming no other flags prevents it. In this particular case, resplit_cutting=2, therefore the drifts will be split on the basis of ***LMAX0=0.005D0*** .

Name	method	#steps	T1%pos	TM%pos	T2%pos
D1	2	458	1	232	462
D1	2	458	488	232	949
D2	2	1	468	468	472
QF	2	1	483	483	487
QD	2	1	1412	1412	1416
B	2	1	1422	1422	1426

If the same called is performed with useknob=.false., then the magnets with knob=true are masked and the other magnets are split.

```
CALL THIN_LENS_RESPLIT(R1,THIN,EVEN=.TRUE.,LMAX0=0.005D0,XBEND=1.D-4,USEKNOB=.FALSE.)
```

Name	method	#steps	T1%pos	TM%pos	T2%pos
D1	2	1	1	1	5
D1	2	1	924	1	928
D2	2	92	112	160	207
QF	6	102	818	871	923
QD	6	102	934	987	1039
B	4	510	1136	1393	1649

Other Global Parameters

1) sixtrack_compatible (true or false)
This enforces 2nd order integrator for all magnets.

2) radiation_bend_split (true or false)

Recuts bends with method two to improve radiation or spin results

3) fuzzy_split (default to 1.0)

If fuzzy_split is greater than 1.0, it lets some magnets with slightly too long integration steps go through, i.e., if ds<lmax0*fuzzy_split.