Nonlinear Beam Dynamics Tools for Field Treatment, Symplectic Tracking and Spin in COSY INFINITY

Martin Berz

Kyoko Makino, Eremey Valetov, David Tarazona,

Adrian Weisskopf, He Zhang, …

Michigan State University

Work supported by the US Department of Energy

(Prehistorical) Introduction

- As a young graduate student, because I expressed some interest in mathematics, I was asked to cross check a set of very complicated formulas for aberrations (higher order terms) of particle optical systems
- Such formulas were the State of the Art to compute aberrations of inhomogeneous bending magnets, electrostatic deflectors, and their fringe fields, etc etc
- Some of the prominent codes were Transport, GIOS, TRIO, MaryLie, etc
- But nobody was quite sure if these formulas were really all correct (although as we saw later, amazingly they mostly were).
- My problem was, yes I liked Math, but I didn't have all that much stamina (or less forgivingly, I was a bit lazy)
- So how do I get out of this – I started to try out some Computer Algebra tools. Unfortunately, they quickly turned out to be hopelessly slow and underpowered.

What's the Problem to be Solved?

- The problem is conceptually simple: there is an iterative order-by-order scheme to calculate higher orders in terms of time integrals of already known lower orders.
- But: The difference between theory and practice is larger in practice than in theory. The complexity of the problem is horrendous, it grows exponentially with order.
- It was universally believed that higher than third order is practically impossible to derive.

Maps as Taylor Series

"The determination of terms of order higher than fourth is very laborious in all but the simplest cases. For this reason, algebraic calculations are usually restricted to the domain of the Seidel theory, supplemented where necessary by ray tracing".

Born-Wolf, *Principles of Optics, Pergamon 1989*

Some Power Series Particle Optics Codes:

What's the Problem to be Solved?

- The problem is conceptually simple: there is an iterative order-by-order scheme to calculate higher orders in terms of time integrals of already known lower orders.
- But: The difference between theory and practice is larger in practice than in theory. The complexity of the problem is horrendous, it grows exponentially with order.
- It was universally believed that higher than third order is practically impossible to derive.
- So I started to read up more on Computer Algebra. I found out that to solve integrals, ODEs, and PDEs, since they can't use "limit of x going to zero" of something or other, they use something called Differential Algebra
- Using these tricks, I built my own Differential Algebra-based computer algebra package, and computed analytic formulas for all common particle optical elements up to order five
- They were directly output in dense code: the worst one, the inhomogeneous electrostatic deflector, had something like 20,000 lines of code.

Differential Algebra – What's That?

- Key Idea: treat derivatives and integrals like algebraic functions ∂ and their inverse, alongside our well-known intrinsic functions and operators.
- Algebraic rules:

$$\partial(a + b) = \partial a + \partial b$$

$$\partial(a \cdot b) = (\partial a) \cdot b + a \cdot (\partial b)$$

- A simple illustrative example: Find a formula for the derivative of the root function $\sqrt{x}$.
- We know $x = \sqrt{x} \cdot \sqrt{x}$. Applying derivation operator:

$$1 = \partial(\sqrt{x} \cdot \sqrt{x}) = 2\sqrt{x}\ \partial(\sqrt{x})$$

- So

$$\partial(\sqrt{x}) = 1 / 2\sqrt{x}$$

From Wikipedia (Oct 2024)

- In mathematics, **differential algebra** is, broadly speaking, the area of mathematics consisting in the study of differential equations and differential operators as algebraic objects in view of deriving properties of differential equations and operators without computing the solutions, similarly as polynomial algebras are used for the study of algebraic varieties, which are solution sets of systems of polynomial equations. Weyl algebras and Lie algebras may be considered as belonging to differential algebra.
- More specifically, *differential algebra* refers to the theory introduced by Joseph Ritt in 1950, in which **differential rings**, **differential fields**, and **differential algebras** are rings, fields, and algebras equipped with finitely many derivations.[1][2][3]

Computational DA – What's That?

- The Differential Algebra for analytic aberrations:
- Polynomials in six phase space variables and t, sin(ωt), cos(ωt), sinh(ωt), cosh(ωt)
- This space is closed under addition, multiplication (i.e. it forms an algebra)
- The space is also closed under diff and integ (i.e. it forms a differential algebra)
- The Program HAMILTON for the Analytic Solution of the Equations of Motion in Particle Optical Systems through Fifth Order
  M. Berz, H. Wollnik, *Nuclear Instruments and Methods* **A258** (1987) 364-373

```
      SUBROUTINE elmm(L,Z,K01,K02,K03,K04,K05,K27,K32,NORDER,NG,ND)
*     ************************************************************
*
*     Subroutine to Compute Aberration Equations Equations
*     Magnetic Multipole to Fifth Order
*     Computer Generated by Program HAMILTON (C) M. Berz 1985
*
      IMPLICIT DOUBLE PRECISION (A - Z)
*
      INTEGER NORDER, NG, ND
*
      DOUBLE PRECISION L(0:461,7)
*
      K30    = 1./(1+K32)
      K31    = 1./(1+K32/2.)
*
      FX2 = -K01*K27
*
      FY2 = +K01*K27
*
      IF(FX2.LT.-1.D-8) THEN
         AFX = SQRT(-FX2)
         CX  = COS(AFX*Z)
         SX  = SIN(AFX*Z)/AFX
      ELSEIF(FX2.GT.1.D-8) THEN
         AFX = SQRT(FX2)
         EX  = EXP(AFX*Z)
         EEX = 1.D0/EX
         CX  = (EX + EEX)/2.D0
         SX  = (EX - EEX)/2.D0/AFX
      ELSE
         CX  = 1.D0
         SX  = Z
         FX2 = 1.D-8
      ENDIF
*
      IF(FY2.LT.-1.D-8) THEN
         AFY = SQRT(-FY2)
         CY  = COS(AFY*Z)
         SY  = SIN(AFY*Z)/AFY
      ELSEIF(FY2.GT.1.D-8) THEN
         AFY = SQRT(FY2)
         EY  = EXP(AFY*Z)
         EEY = 1.D0/EY
         CY  = (EY + EEY)/2.D0
```

```
         SY  = (EY - EEY)/2.D0/AFY
      ELSE
         CY  = 1.D0
         SY  = Z
         FY2 = 1.D-8
      ENDIF
*
*
      CS2 = CX
      CS3 = SX
      CS4 = CY
      CS5 = SY
      CS6 = Z
      KK2 = K31*K32
      KK3 = K30*K32
      FF2 = FX2
      TT2 = CS2
      TT3 = CS3
      TT4 = CS3*FF2
      TT5 = CS4
      TT6 = CS5
      TT7 = CS5*FF2
      TT8 = CS6
      TT9 = CS6*KK2
      TT10 = CS6*KK3
      L(1,1) = (+TT2)
      L(2,1) = (+TT3)
      L(1,2) = (+TT4)
      L(2,2) = (+TT2)
      L(3,3) = (+TT5)
      L(4,3) = (+TT6)
      L(3,4) = (-TT7)
      L(4,4) = (+TT5)
*
      IF(ND.EQ.0.AND.NG.EQ.0) GOTO 100
*
      L(6,6) = (+1)
      L(6,7) = (-0.5D+00*TT8-0.25D+00*TT9+TT10)
*
      IF(NG.EQ.0) GOTO 100
*
      L(5,5) = (+1)
      L(5,7) = (+0.5D+00*TT8+0.25D+00*TT9-TT10)
*
  100 IF(NORDER.EQ.1) GOTO 1000
*
      CS7 = CS3*CX
```

```
CS8 = CS3*SX
CS9 = CS4*CX
CS10 = CS4*SX
CS11 = CS5*CX
CS12 = CS5*SX
CS13 = CS5*CY
CS14 = CS5*SY
CS15 = CS6*CX
CS16 = CS6*SX
CS17 = CS6*CY
CS18 = CS6*SY
KK4 = KK2*K31*K32
KK5 = KK3*K31*K32
KK6 = K02*K27
FF3 = 1/FX2/FX2
FF4 = FF3*FX2
TT11 = KK6*FF4
TT12 = CS2*KK6*FF4
TT13 = CS8*KK6
TT14 = CS3*KK6*FF4
TT15 = CS7*KK6*FF4
TT16 = KK6*FF3
TT17 = CS2*KK6*FF3
TT18 = CS8*KK6*FF4
TT19 = CS14*KK6
TT20 = CS13*KK6*FF4
TT21 = CS14*KK6*FF4
TT22 = CS16*FF2
TT23 = CS16*KK2*FF2
TT24 = CS3*KK2
TT25 = CS15
TT26 = CS15*KK2
TT27 = CS3*KK6
TT28 = CS7*KK6
TT29 = CS13*KK6
TT30 = CS3*KK2*FF2
TT31 = CS15*FF2
TT32 = CS15*KK2*FF2
TT33 = CS4*KK6*FF4
TT34 = CS9*KK6*FF4
TT35 = CS12*KK6
TT36 = CS10*KK6*FF4
TT37 = CS5*KK6*FF4
TT38 = CS11*KK6*FF4
TT39 = CS4*KK6*FF3
TT40 = CS9*KK6*FF3
TT41 = CS12*KK6*FF4
```

```
      TT42 = CS18*FF2
      TT43 = CS18*KK2*FF2
      TT44 = CS5*KK2
      TT45 = CS17
      TT46 = CS17*KK2
      TT47 = CS10*KK6
      TT48 = CS5*KK6
      TT49 = CS11*KK6
      TT50 = CS5*KK2*FF2
      TT51 = CS17*FF2
      TT52 = CS17*KK2*FF2
      TT53 = CS7*FF2
      TT54 = CS6*FF2
      TT55 = CS8*FF2
      TT56 = CS7
      TT57 = CS13*FF2
      TT58 = CS14*FF2
      TT59 = CS13
      TT60 = CS6*KK4
      TT61 = CS6*KK5
      L(7,1) = (+0.33333334327D+00*(+TT11-TT12-TT13))
      L(8,1) = (+0.66666668654D+00*(+TT14-TT15))
      L(13,1) = (+0.66666668654D+00*(-TT16+TT17)-0.33333333333D+00*TT
     *  18)
      L(18,1) = (+0.60000002384D+00*(-TT11+TT12)+0.2D+00*TT19)
      L(19,1) = (+0.40000000596D+00*(+TT14-TT20))
      L(22,1) = (+0.40000000596D+00*(-TT16+TT17)-0.2D+00*TT21)
      L(7,2) = (-0.33333333333D+00*TT27-0.66666666667D+00*TT28)
      L(8,2) = (+0.66666668654D+00*(-TT11+TT12)-0.13333333333D+01*TT13)
      L(13,2) = (+0.66666668654D+00*(+TT14-TT15))
      L(18,2) = (+0.6D+00*TT27+0.4D+00*TT29)
      L(19,2) = (+0.40000000596D+00*(-TT11+TT12)+0.8D+00*TT19)
      L(22,2) = (+0.40000000596D+00*(+TT14-TT20))
      L(9,3) = (+0.40000000596D+00*(-TT33+TT34)+0.8D+00*TT35)
      L(14,3) = (+0.4D+00*TT36-0.12D+01*TT37+0.8D+00*TT38)
      L(10,3) = (-0.8D+00*TT36+0.40000000596D+00*(+TT37+TT38))
      L(15,3) = (+0.80000001192D+00*(+TT39-TT40)+0.4D+00*TT41)
      L(9,4) = (+0.12D+01*TT47+0.40000000596D+00*(+TT48+TT49))
      L(14,4) = (+0.12000000477D+01*(-TT33+TT34)+0.4D+00*TT35)
      L(10,4) = (+0.40000000596D+00*(+TT33-TT34)+0.12D+01*TT35)
      L(15,4) = (-0.4D+00*TT36-0.8D+00*TT37+0.12D+01*TT38)
      L(7,7) = (+0.25D+00*(+TT53-TT54))
      L(8,7) = (+0.5D+00*TT55)
      L(13,7) = (+0.25D+00*(+TT56+TT8))
      L(18,7) = (+0.25D+00*(-TT57+TT54))
      L(19,7) = (-0.5D+00*TT58)
      L(22,7) = (+0.25D+00*(+TT59+TT8))
```

27,000 lines further down:

```
      L(449,7) = (-0.87890625D-01*TT59-0.244140625D-01*TT347
     *  +0.2197265625D-01*TT1723+0.14282226563D+00*TT6924+0.390625D-01*(
     *  +TT348-TT1724)-0.263671875D+00*TT6925-0.380859375D+00*TT8
     *  -0.1162109375D+00*TT9+0.9521484375D-01*TT60+0.26733398438D+00*TT
     *  351+0.1484375D+00*(+TT10-TT61)-0.439453125D+00*TT352
     *  +0.29296875D+00*TT349+0.91796875D-01*TT350-0.732421875D-01*TT
     *  1725-0.12451171875D+00*TT6926+0.109375D+00*(-TT1726+TT1727)
     *  +0.17578125D+00*TT6927+0.140625D+00*TT1728+0.546875D-01*TT1729
     *  -0.3515625D-01*TT1730-0.29296875D-01*TT6928+0.3125D-01*(-TT
     *  6929+TT6930)+0.234375D-01*TT6931+0.78125D-02*(+TT6892-TT6933)
     *  +0.390625D-02*(+TT6893+TT6934)+0.1953125D-02*(-TT6894+TT6935)
     *  -0.9765625D-03*TT6895-0.15625D-01*TT6932)
      L(458,7) = (+0.234375D-01*TT8-0.390625D-02*TT9-0.8203125D-01*TT
     *  60+0.380859375D+00*TT351-0.32470703125D+00*TT1731
     *  +0.76904296875D-01*TT6936+0.15625D-01*TT10+0.21875D+00*TT61
     *  -0.9140625D+00*TT352+0.7421875D+00*TT1732-0.1708984375D+00*TT
     *  6937)
      L(459,7) = (+0.390625D-01*TT8+0.390625D-02*TT9-0.1171875D-01*TT
     *  60-0.68359375D-01*TT351+0.18798828125D+00*TT1731
     *  -0.76904296875D-01*TT6936-0.15625D-01*TT10+0.3125D-01*TT61
     *  +0.1640625D+00*TT352-0.4296875D+00*TT1732+0.1708984375D+00*TT
     *  6937)
      L(460,7) = (+0.13671875D+00*TT8+0.29296875D-01*TT9+0.5859375D-02
     *  *TT60-0.48828125D-02*TT351-0.25634765625D-01*TT1731
     *  +0.38452148438D-01*TT6936-0.1171875D+00*TT10-0.15625D-01*TT61
     *  +0.1171875D-01*TT352+0.5859375D-01*TT1732-0.8544921875D-01*TT
     *  6937)
*
  500 IF(NORDER.EQ.5) GOTO 1000
*
 1000 CONTINUE
*
      RETURN
      END
```

History of Higher Order Optics

	Light Optics (Round Lenses)	**Electron Optics** (Round Lenses)	**Particle Optics** (Non-Round Lenses)
1	Gauss 1841		?
2	(Gauss 1841)		Brown 1959
3	Petzval 1840 Seidel 1856	Scherzer 1936	Matsuda, Wollnik 1965
4			
5	Kohlschütter, Schwarzschild 1905 Rabinovich 1946		M.B. 1985

Maps as Taylor Series

"The determination of terms of order higher than fourth is very laborious in all but the simplest cases. For this reason, algebraic calculations are usually restricted to the domain of the Seidel theory, supplemented where necessary by ray tracing".

Born-Wolf, *Principles of Optics, Pergamon 1989*

Some Power Series Particle Optics Codes:

- TRANSPORT (2nd order, thick elements, early 60s)
- GIOS (3rd order, thick elements, fringe fields, late 60s)
- MaryLie (3rd order, thick elements, fringe fields, late 70s)
- COSY 5.0 (5th order, thick elements, fringe fields, 1985) □

Computational DA – What's That?

- The Differential Algebra for analytic aberrations:
- Polynomials in six phase space variables and t, sin(ωt), cos(ωt), sinh(ωt), cosh(ωt)
- This space is closed under addition, multiplication (i.e. it forms an algebra)
- The space is also closed under diff and integ (i.e. it forms a differential algebra)
- The Program HAMILTON for the Analytic Solution of the Equations of Motion in Particle Optical Systems through Fifth Order
  M. Berz, H. Wollnik, *Nuclear Instruments and Methods* **A258** (1987) 364-373
- This space had floating point coefficients, so it was always "numerical"
- That being said, why not evaluate all the integrals not in closed form, but using a numerical integrator?
- This is the conceptually simplest method to get high order maps simply, called TPSA method
- The Method of Power Series Tracking for the Mathematical Description of Beam Dynamics
  M. Berz, *Nuclear Instruments and Methods* **A258** (1987) 431-436

Maps as Taylor Series

"The determination of terms of order higher than fourth is very laborious in all but the simplest cases. For this reason, algebraic calculations are usually restricted to the domain of the Seidel theory, supplemented where necessary by ray tracing".

Born-Wolf, *Principles of Optics, Pergamon 1989*

Some Power Series Particle Optics Codes:

- TRANSPORT (2nd order, thick elements, early 60s)
- GIOS (3rd order, thick elements, fringe fields, late 60s)
- MaryLie (3rd order, thick elements, fringe fields, late 70s)
- COSY 5.0 (5th order, thick elements, fringe fields, 1985)
- COSY INFINITY (arbitrary order, thick elements, fringe fields...)□

Codes Using DA methods

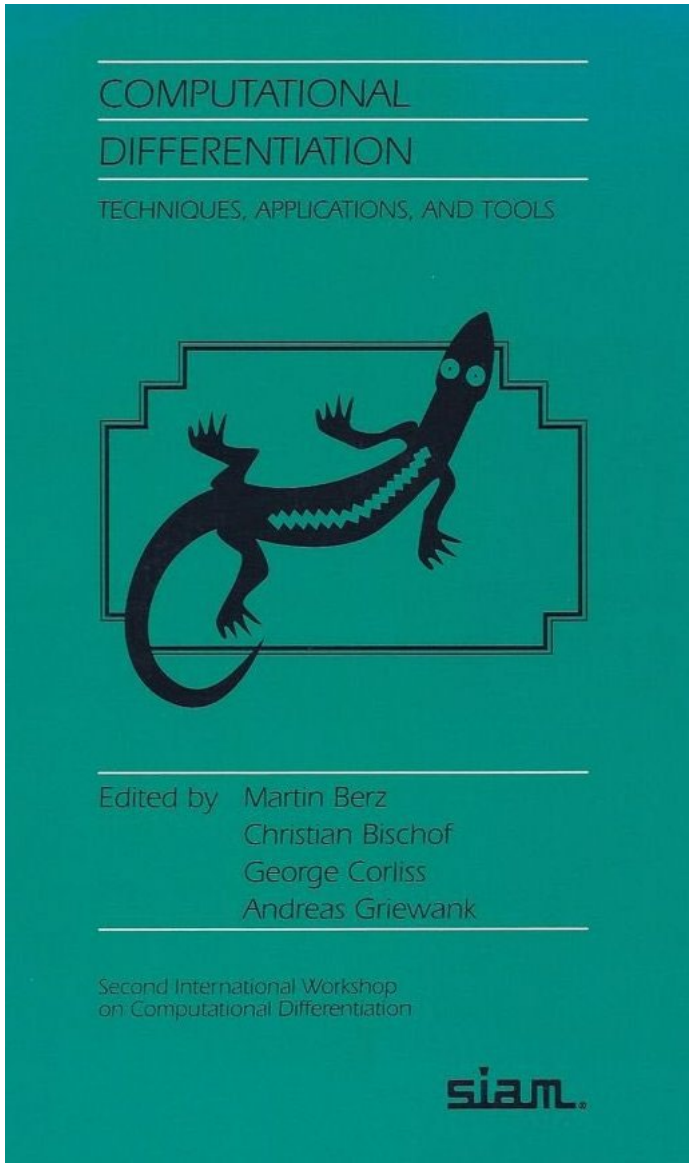
Since we introduced it, many/most modern codes use DA:

- PowerTrack (Berz) *)
- TPot (Talman) *)
- TLie (van Zeijts)
- ZLib (Yan)
- MXYZTPLK (Michelotti, ...)
- DACYC (Davies, ...)
- Classic (Iselin, ...)
- PTC (Forest, ...) *)
- MAD-X, SixTrack (Schmidt, ...) *)
- TPSALib (Yang)
- COSY Infinity *)

*) Using modern or earlier versions of our DA package

Automatic Differentiation – What It Is and Isn't

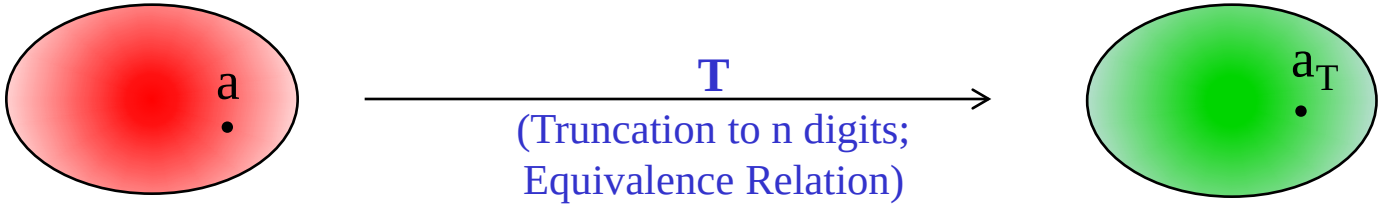
- Transform existing code to compute not only values, but derivatives.
- Experts often shy away from saying "Automatic", since neither the forward nor the reverse mode usually work as black box. It usually requires clever "checkpointing", i.e. doing things in handpicked pieces.
- Having derivatives more often than not ends up in re-doing the code for increased efficiency.
- First Conference in the field: Andreas Griewank, who sadly passed in 2021. We organized the second Conference
- Amazon: This volume goes beyond the first volume published in 1991 (SIAM) in that it encompasses both the automatic transformation of computer programs as well as the methodologies for the efficient exploitation of mathematical underpinnings or program structure.



Computational DA – Better Ways

- Except of special cases, stick to the simplest possible DA: Polynomials in phase space variables, parameters, and t
- This space is closed under addition, multiplication, it forms an algebra. It is also closed under diff and integ, so it forms a differential algebra.
- Do not try to just use a tracking code and DA-ify it. Rather think from scratch to build algorithms fully exploiting the DA spirit and possibilities

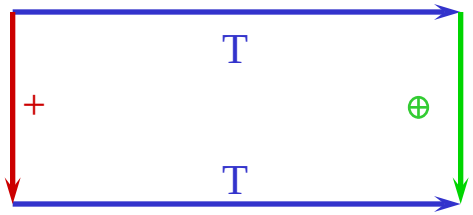
Number Fields and Floating Point Numbers



Real Numbers

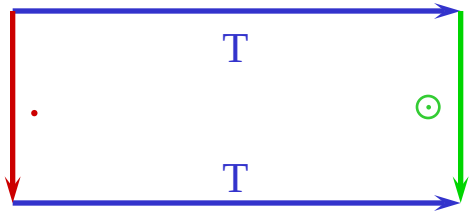
Floating Point Numbers

$c = a + b$



$c_T = a_T \oplus b_T$

$c = a \cdot b$



$c_T = a_T \odot b_T$

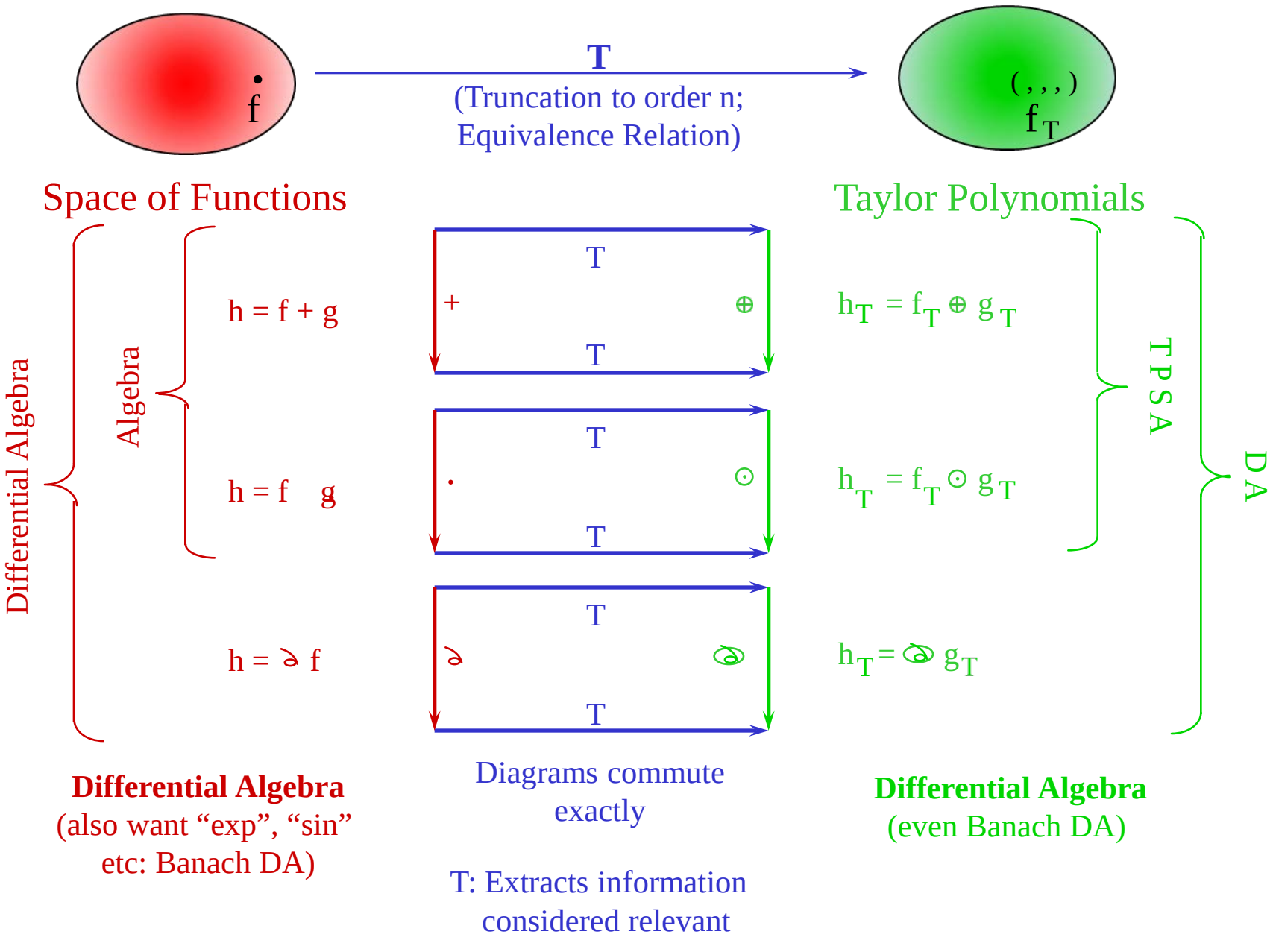
Field
(Also want “exp”, “sin” etc: Banach Field)

Diagrams commute “approximately”

Field
(“approximately”)

T: Extracts information considered relevant

Function Algebras



Computational DA – Better Ways

- Except of special cases, stick to the simplest possible DA: Polynomials in phase space variables, parameters, and t
- This space is closed under addition, multiplication, it forms an algebra. It is also closed under diff and integ, so it forms a differential algebra.
- Do not try to just use a tracking code and DA-ify it. Rather think from scratch to build algorithms fully exploiting the DA spirit and possibilities
- Solve ODEs (motion) and PDEs (fields) directly using differential algebraic tools:
- Picard Iteration et al. for ODEs an enhanced version of this won the Moore Prize for rigorous computing for Makino and Berz "for the development of novel high performance rigorous self-verified integrators for flows of ODEs based on Taylor model and differential algebraic methods"
- Fixed Point Iteration for PDEs (allows to self-consistently solve for Maxwellian fields)
- Minimal Impact Symplectification of map tracking
- Fast Multipole Methods for some space charge simulations

COSY INFINITY

- Arbitrary order
- Maps depending on parameters (mass dependence!)
- No approximations in motion or field description
- Large library of elements
- Arbitrary Elements (you specify fields)
- Very flexible input language
- Powerful interactive graphics
- Errors: position, tilt, rotation
- Tracking through maps
- Normal Form Methods
- Spin dynamics
- Fast fringe field models using SYSCA approach
- Reference manual (80 pages) and Programming manual (90 pages)
- As of December 2004, more than 1000 registered users

Elements in COSY

- Magnetic and electric multipoles
- Superimposed multipoles
- Combined function bending magnets with curved edges
- Electrostatic deflectors
- Wien filters
- Wigglers
- Solenoids, various field configurations
- 3 tube electrostatic round lens, various configurations
- Exact fringe fields to all of the above
- Fast fringe fields (SYSCA)
- General electromagnetic element (measured data)
- Glass lenses, mirrors, prisms with arbitrary surfaces
- Misalignments: position, angle, rotation

All can be computed to arbitrary order, and the dependence on any of their parameters can be computed.

The Operator ∂^{-1}on Taylor Models

Let (P_n, I_n) be an n-th order Taylor model of f. From this we can obtain a Taylor model for the indefinite integral $\partial_i^{-1} f = \int f \; dx_i'$ with respect to variable x_i.

Taylor polynomial part: $\int_0^{x_i} P_{n-1} dx_i'$,

Remainder Bound: $(B(P_n - P_{n-1}) + I_n) \cdot B(x_i)$, where $B(P)$ is a polynomial bound.

So define the operator ∂_i^{-1}on space of Taylor models as

$$\begin{aligned} &\partial_i^{-1}(P_n, I_n) \\ &= \left(\int_0^{x_i} P_{n-1} dx_i' \;\; , \;\; (B(P_n - P_{n-1}) + I_n) \cdot B(x_i) \right) \end{aligned}$$

Taylor Models for the Flow

Goal: Determine a Taylor model, consisting of a Taylor Polynomial and an interval bound for the remainder, for the flow of the differential equation

$$\frac{d}{dt}\vec{r}(t) = \vec{F}(\vec{r}(t), t)$$

where $\vec{F}$ is sufficiently differentiable. The Remainder Bound should be fully rigorous for all initial conditions $\vec{r}_0$ and times t that satisfy

$$\vec{r}_0 \in [\vec{r}_{01}, \vec{r}_{02}] = \vec{B}$$
$$t \in [t_0, t_1].$$

In particular, $\vec{r}_0$ itself may be a Taylor model, as long as its range is known to lie in $\vec{B}$.

The Use of Schauder's Theorem

Re-write differential equation as integral equation

$$\vec{r}(t) = \vec{r}_0 + \int_{t_0}^{t} \vec{F}(\vec{r}(t'), t')\, dt'.$$

Now introduce the operator

$$A : \vec{C}^0[t_0, t_1] \to \vec{C}^0[t_0, t_1]$$

on space of continuous functions via

$$A\left(\vec{f}\right)(t) = \vec{r}_0 + \int_{t_0}^{t} \vec{F}\left(\vec{f}(t'), t'\right)\, dt'.$$

Then the solution of ODE is transformed to a fixed-point problem on space of continuous functions

$$\vec{r} = A(\vec{r}).$$

Theorem (Schauder): *Let A be a continuous operator on the Banach Space X. Let $M \subset X$ be compact and convex, and let $A(M) \subset M$. Then A has a fixed point in M, i.e. there is an $\vec{r} \in M$ such that $A(\vec{r}) = \vec{r}$.*

The Polynomial of the Self-Including Set

Attempt sets M^* of the form

$$M^* = M_{\vec{P}^*+\vec{I}^*} \text{ where}$$
$$\vec{P}^* = \mathcal{M}_n(\vec{r}_0, t),$$

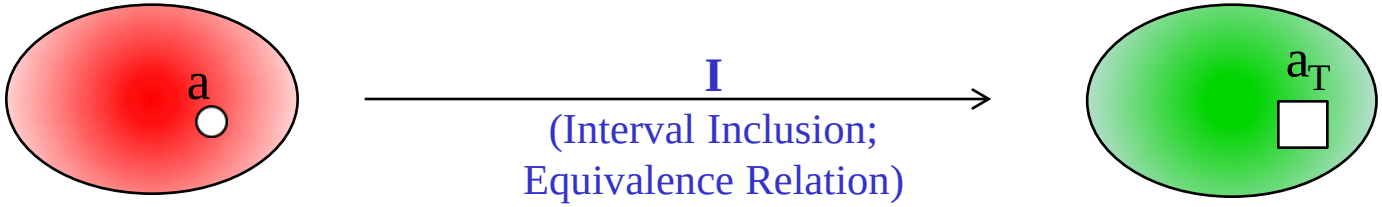
the n-th order Taylor expansion of the flow of the ODE. It is to be expected that $\vec{I}^*$ can be chosen smaller and smaller as order n of $\vec{P}^*$ increases.

This requires knowledge of nth order flow $\mathcal{M}_n(\vec{r}_0, t)$, including time dependence. It can be obtained by iterating in polynomial arithmetic, or Taylor models without treatment of a remainder. To this end, one chooses an initial function $\mathcal{M}_n^{(0)}(\vec{r}, t) = \mathcal{I}$, where $\mathcal{I}$ is the identity function, and then iteratively determines

$$\mathcal{M}_n^{(k+1)} =_n A(\mathcal{M}_n^{(k)}).$$

This process converges to the exact result $\mathcal{M}_n$ in exactly n steps.

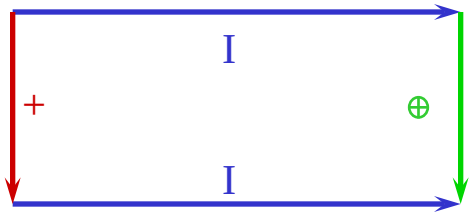
Set Inclusions (Intervals)



Real Numbers

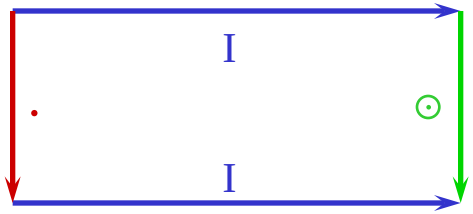
Floating Point Intervals

$c = a + b$



$c_I = a_I \oplus b_I$

$c = a \cdot b$



$c_I = a_I \odot b_I$

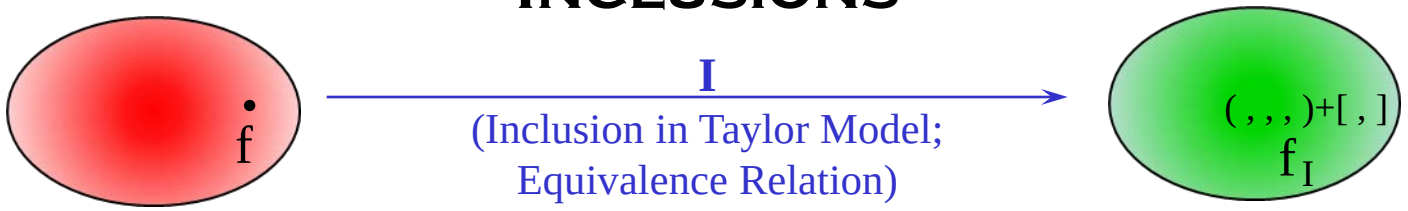
Field
(Also want “exp”, “sin” etc: Banach Field)

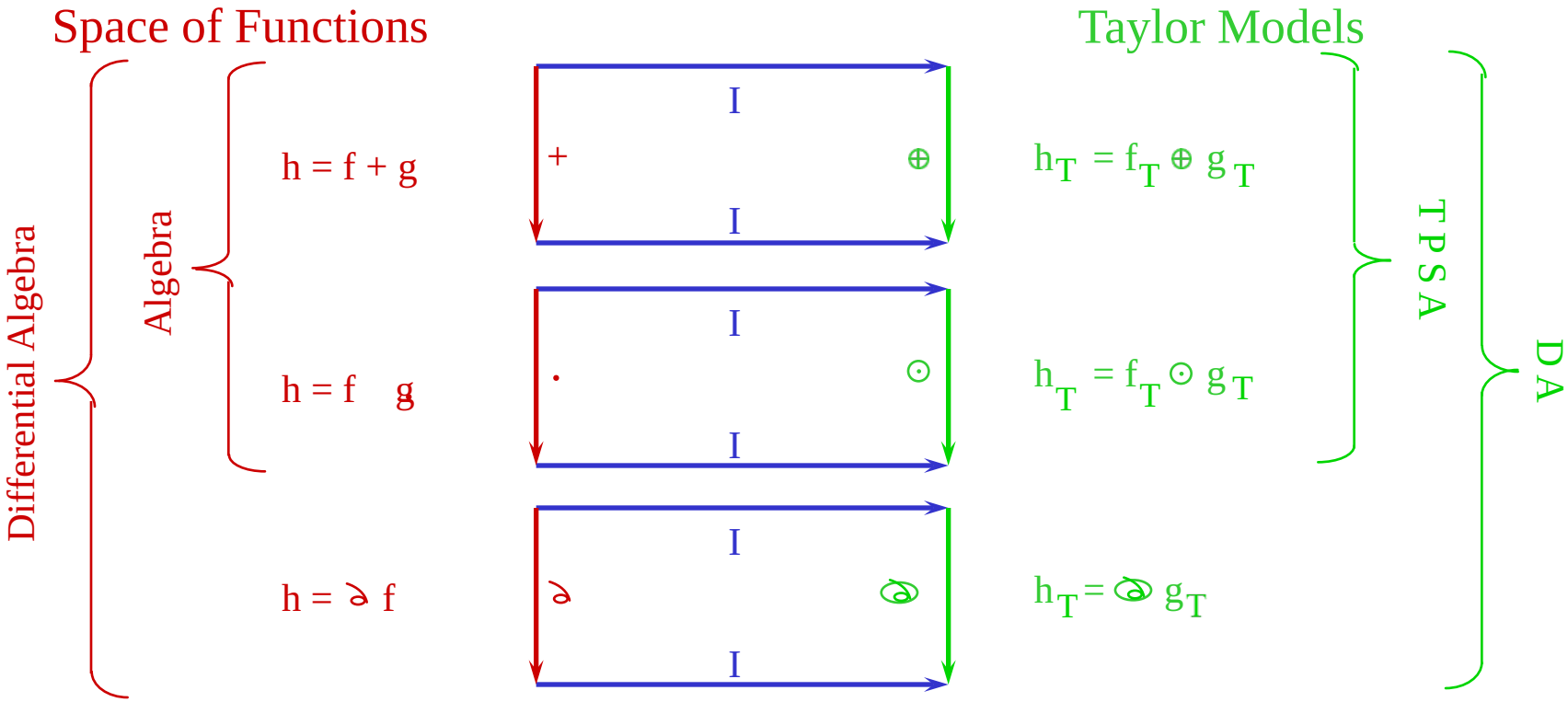
Diagrams commute exactly!

Little Algebraic Structure

I: Extracts information considered relevant

Function Algebra Inclusions





Differential Algebra
(also want “exp”, “sin” etc: Banach DA)

Diagrams commute exactly

T: Extracts information considered relevant

Differential Algebra
(even Banach DA)

The Remainder of the Self-Including Set

Now try to find $\vec{I}^*$ such that

$$A(\mathcal{M}_n + \vec{I}^*) \subset \mathcal{M}_n + \vec{I}^*,$$

the Schauder inclusion requirement. Suitable choice for $\vec{I}^*$ requires experimenting, but is greatly simplified by the observation

$$\vec{I}^* \supset \vec{I}^{(0)} = A(\mathcal{M}_n(\vec{r}, t) + [\vec{0}, \vec{0}]) - \mathcal{M}_n(\vec{r}, t).$$

Evaluating the right hand side in RDA yields a lower bound for $\vec{I}^*$, and a benchmark for the size to be expected. Now iteratively try

$$\vec{I}^{(k)} = 2^k \cdot \vec{I}^{(0)},$$

until computational inclusion is found, i.e.

$$A(\mathcal{M}_n(\vec{r}, t) + \vec{I}^{(k)}) \subset \mathcal{M}_n(\vec{r}, t) + \vec{I}^{(k)}.$$

Field Description in Differential Algebra

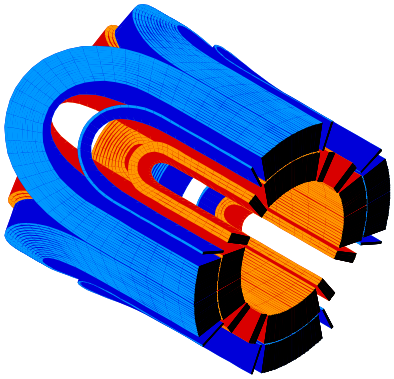
There are various DA algorithms to treat the fields of beam optics efficiently. For example, **DA PDE Solver**

- requires to supply only
  - the midplane field for a midplane symmetric element.
  - the on-axis potential for straight elements like solenoids, quadrupoles, and higher multipoles.
- treats arbitrary fields straightforwardly.
  - Magnet (or, Electrostatic) fringe fields:
    The Enge function fall-off model

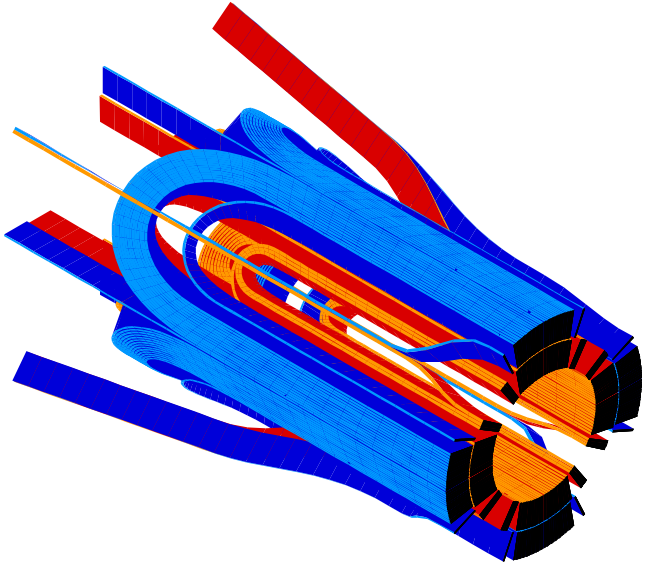
$$F(s) = \frac{1}{1 + \exp(a_1 + a_2 \cdot (s/D) + ... + a_6 \cdot (s/D)^5)}$$

    where D is the full aperture.
    Or, any arbitrary model including the measured data representation.
  - Solenoid fields including the fringe fields.
  - Measured fields: E.g. Use Gaussian wavelet representation.–

  Etc. etc.

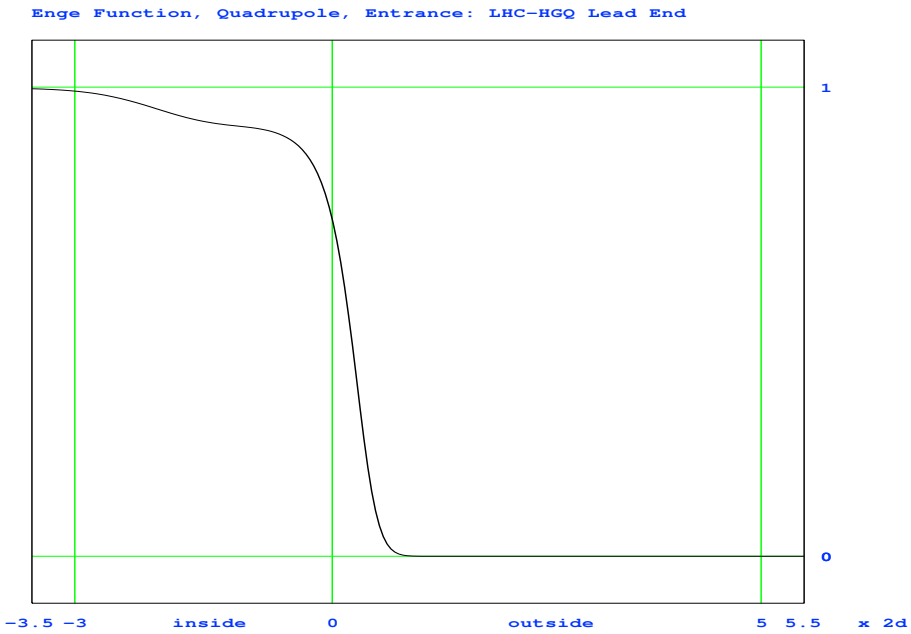


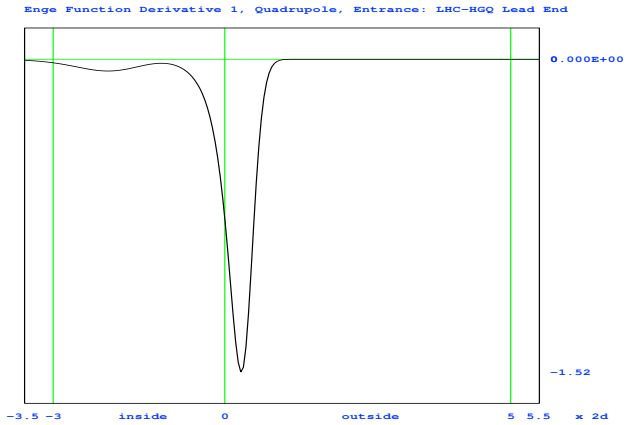
The HGQ return end

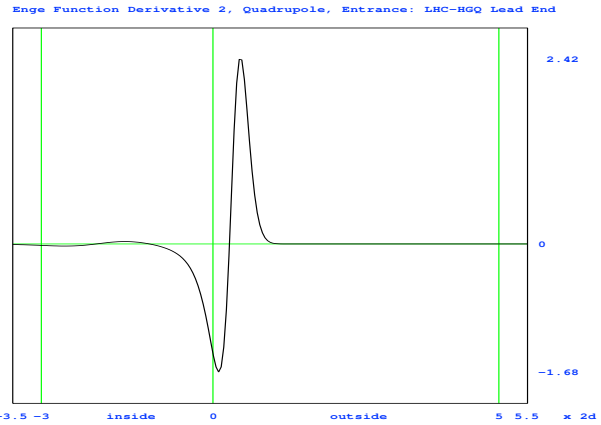


The HGQ lead end

LHC-HGQ Lead End Enge Function







DA Fixed Point PDE Solvers

The **DA fixed point theorem** allows to **solve PDEs iteratively** in **finitely many steps** by rephrasing them in terms of a fixed point problem.

Consider the rather general PDE

$$a_1\frac{\partial}{\partial x}\left(a_2\frac{\partial}{\partial x}V\right)+b_1\frac{\partial}{\partial y}\left(b_2\frac{\partial}{\partial y}V\right)+c_1\frac{\partial}{\partial z}\left(c_2\frac{\partial}{\partial z}V\right)=0,$$

where a_i, b_i, c_i are functions of x, y, z.

The PDE is re-written in **fixed point form** as

$$V=V|_{y=0}+\int_0^y\frac{1}{b_2}\left(b_2\frac{\partial V}{\partial y}\right)\Bigg|_{y=0}$$
$$-\int_0^y\frac{1}{b_2}\int_0^y\left(\frac{a_1}{b_1}\frac{\partial}{\partial x}\left(a_2\frac{\partial V}{\partial x}\right)+\frac{c_1}{b_1}\frac{\partial}{\partial z}\left(c_2\frac{\partial V}{\partial z}\right)\right)dydy.$$

Assume the derivatives of V and $\partial V/\partial y$ with respect to x and z are **known in the plane** $y=0$. Then the right hand side is **contracting** with respect to y (which is necessary for the DA fixed point theorem), and the various orders in y can be **iteratively** calculated by mere iteration.

The Crux of Symplectic Tracking

- Symplecticity governs all Hamiltonian systems
- Symplecticity is rather hard to enforce; thus:
- Either try hard to track the **right** system, end up being non-symplectic
- Or track the **wrong** system with symplectic models

The Crux of Symplectic Tracking

- Symplecticity governs all Hamiltonian systems
- Symplecticity is rather hard to enforce; thus:
- Either try hard to track the **right** system, end up being non-symplectic
- Or track the **wrong** system with symplectic models

Right System, Non-Symplectic

- Best possible fields, potentials
- Exact Hamiltonian
- Good integrators
- Examples: numerical integrators,
  Map codes

The Crux of Symplectic Tracking

- Symplecticity governs all Hamiltonian systems
- Symplecticity is rather hard to enforce; thus:
- Either try hard to track the **right** system, end up being non-symplectic
- Or track the **wrong** system with symplectic models

Right System, Non-Symplectic

- Best possible fields, potentials
- Exact Hamiltonian
- Good integrators
- Examples: numerical integrators, Map codes

Wrong System, Symplectic

- Approximate Hamiltonian
- Approximate Fields
- Symplectic Integrators
- Examples: Kick codes

The Crux of Symplectic Tracking

- Symplecticity governs all Hamiltonian systems
- Symplecticity is rather hard to enforce; thus:
- Either try hard to track the **right** system, end up being non-symplectic
- Or track the **wrong** system with symplectic models

Right System, Non-Symplectic

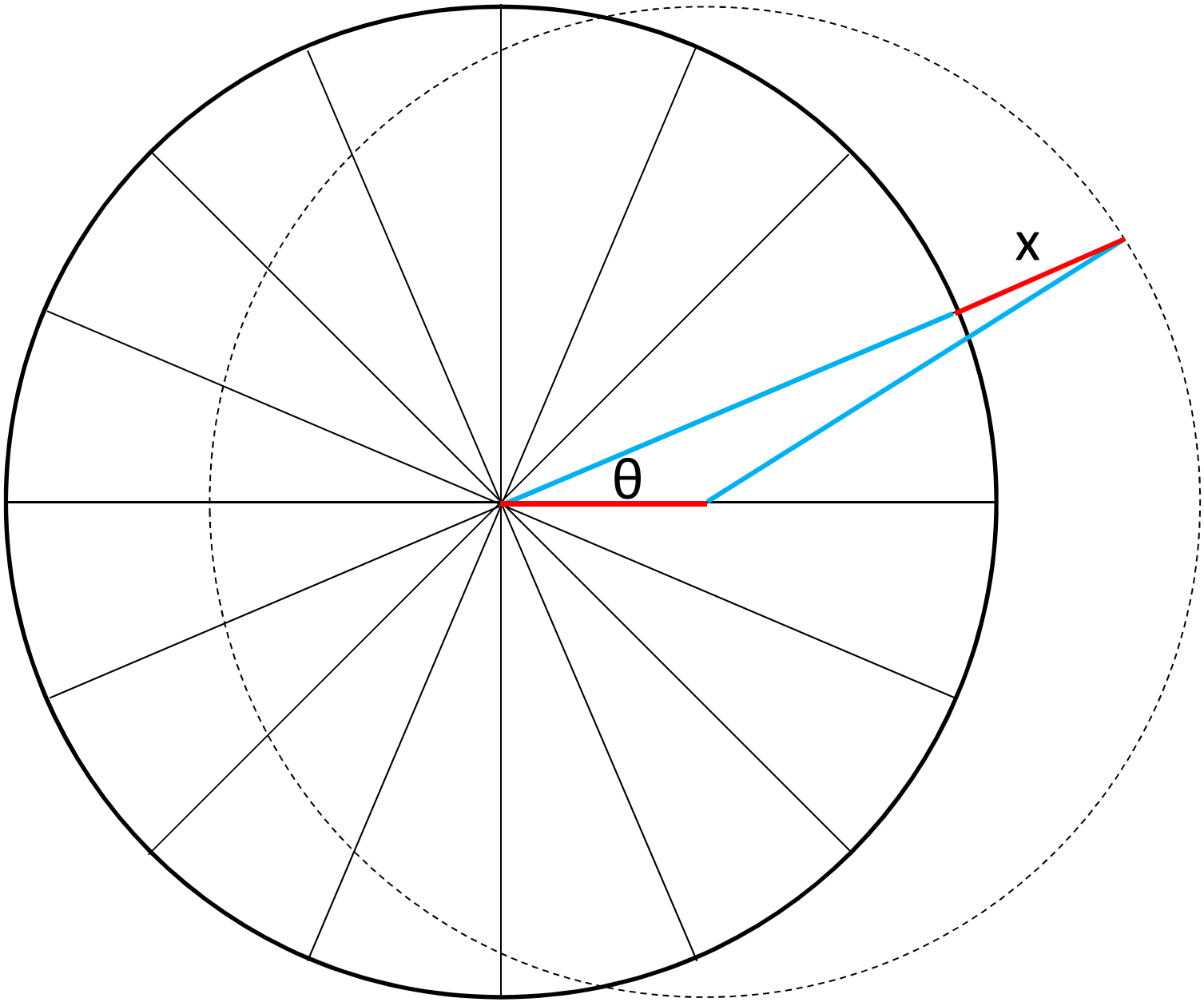
- Best possible fields, potentials
- Exact Hamiltonian
- Good integrators
- Examples: numerical integrators, Map codes

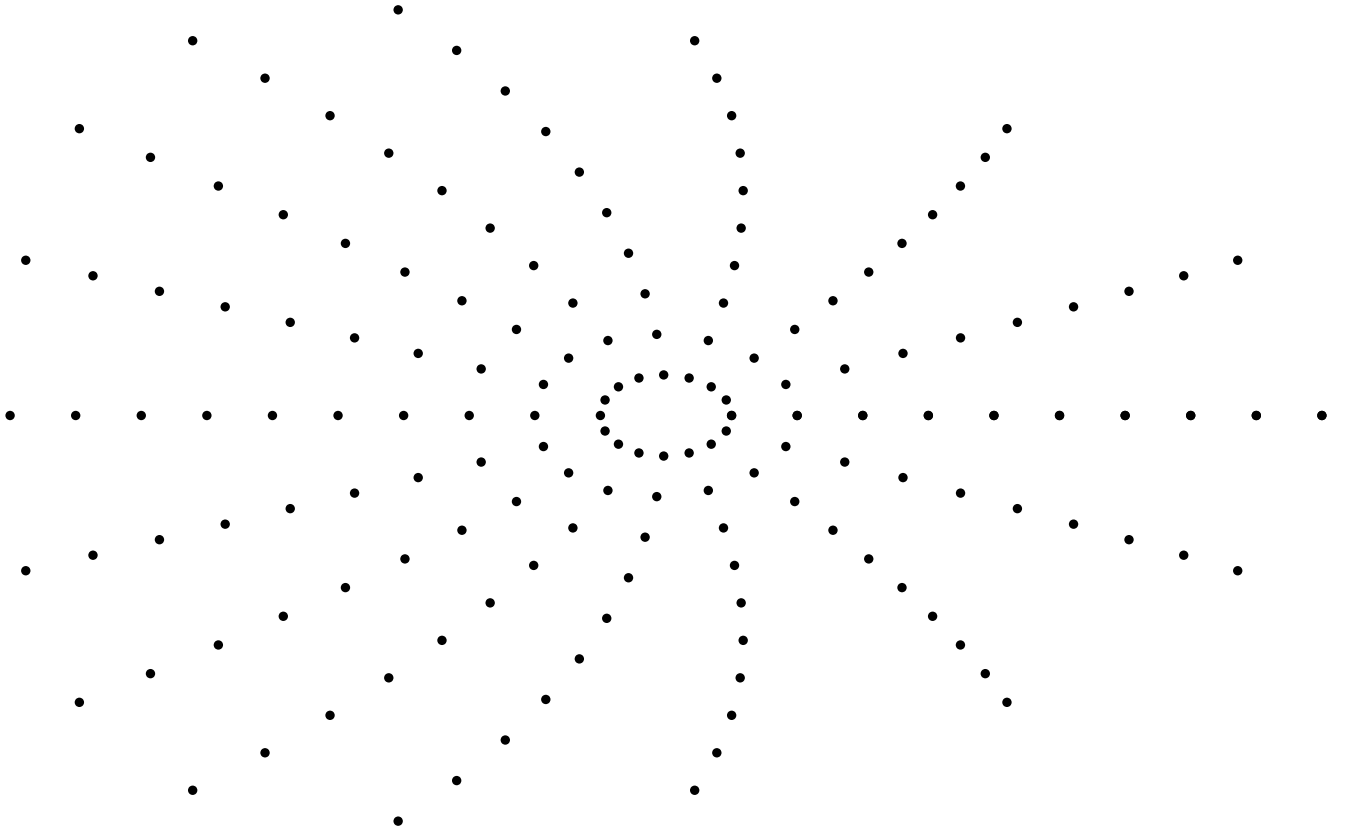
Wrong System, Symplectic

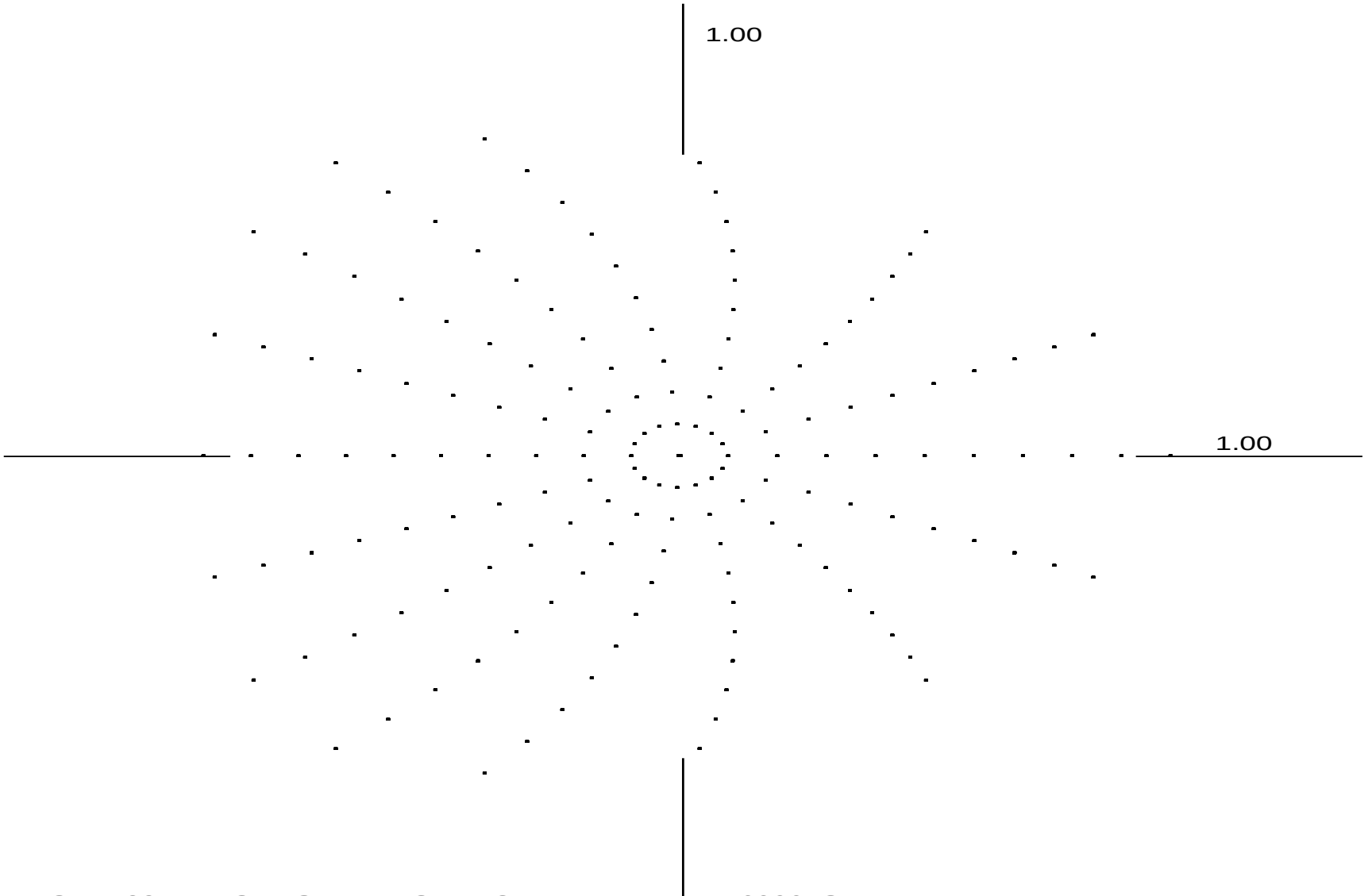
- Approximate Hamiltonian
- Approximate Fields
- Symplectic Integrators
- Examples: Kick codes

Goal: Search **wrong** system nearest to **righ**t system

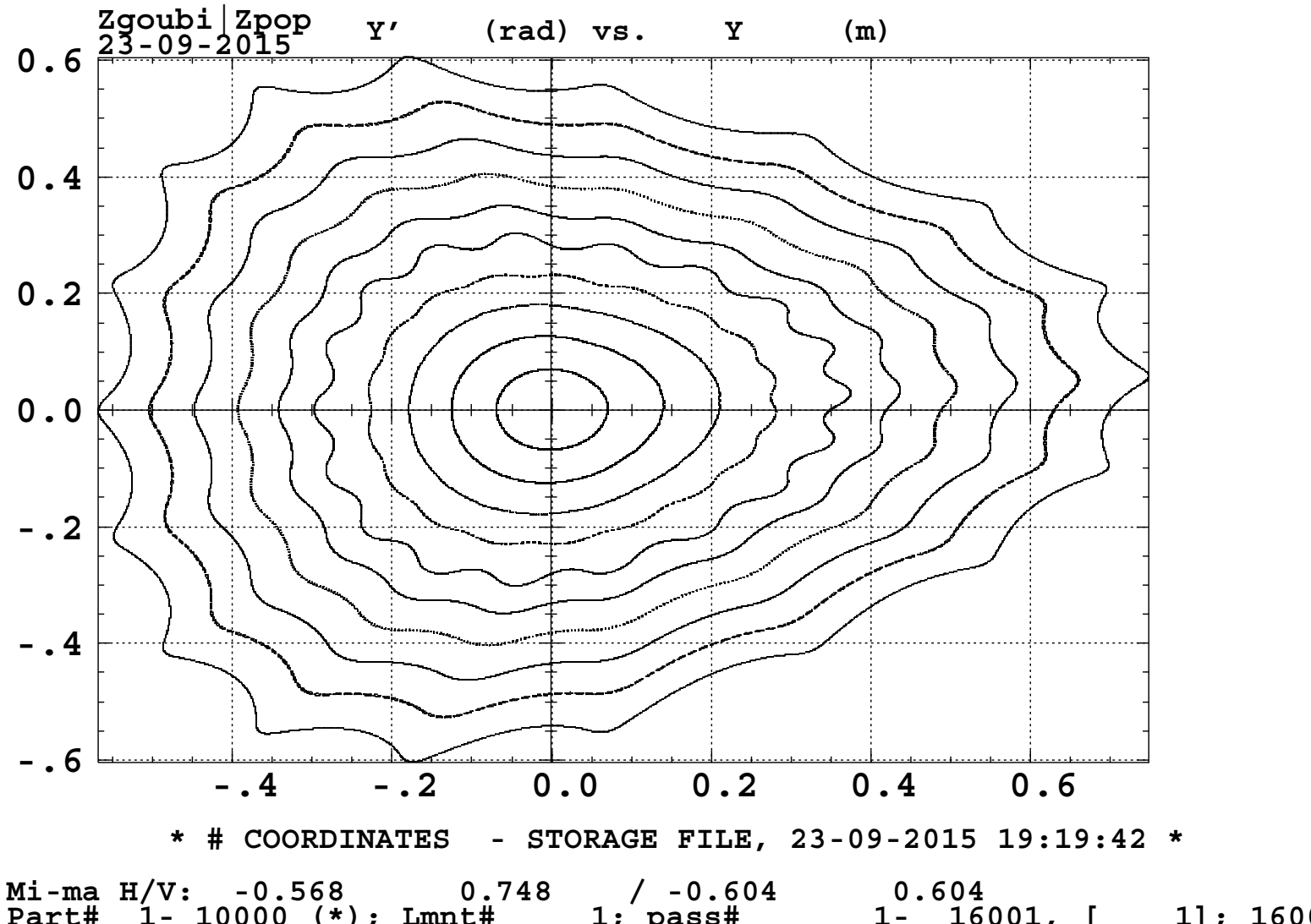
- Start with best possible **right** system
- High-order transfer map using “best” fields
- This makes it **wrong** - finite order, numerical error
- Symplectify using “nearest” via Hofer’s metric

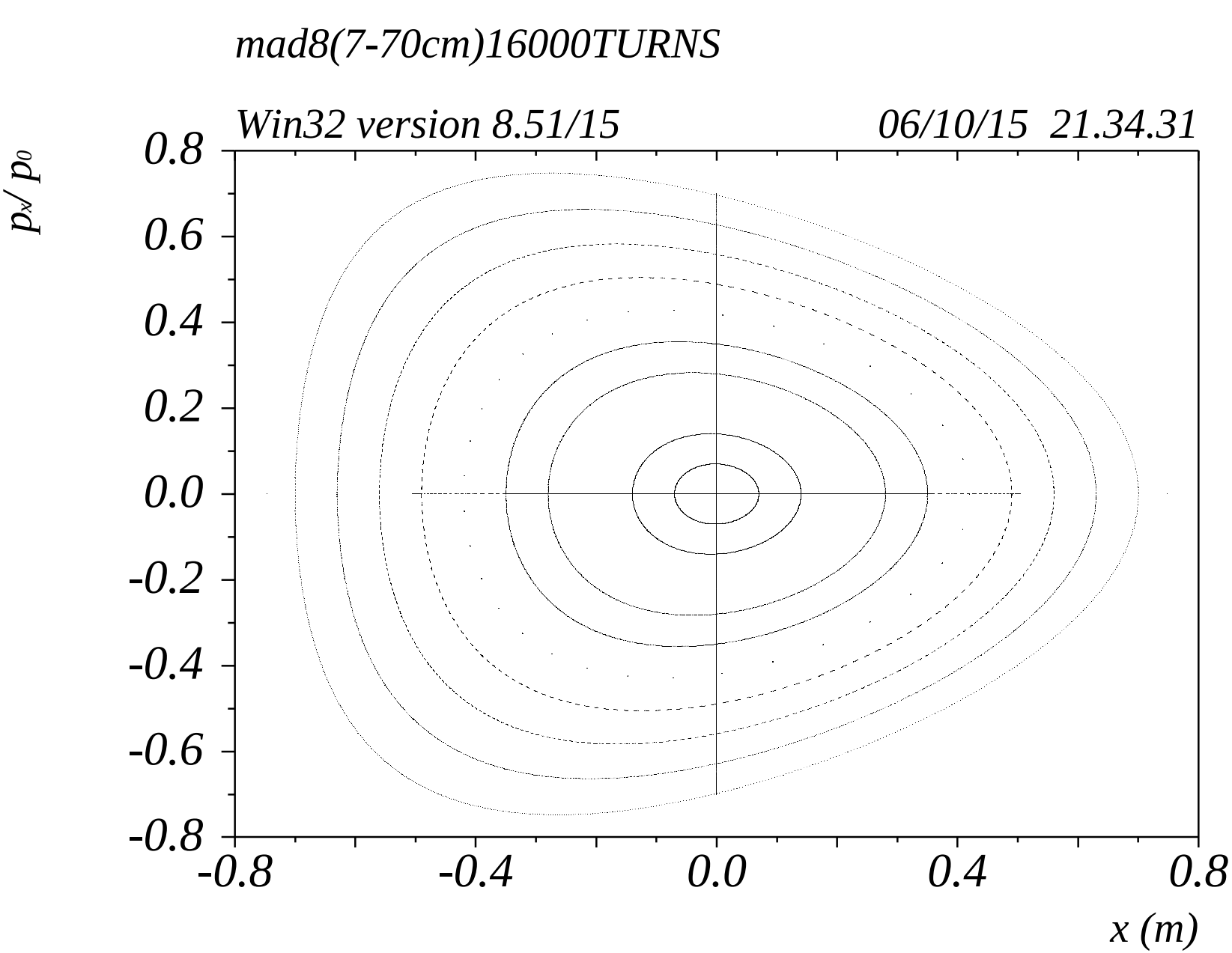

x
θ

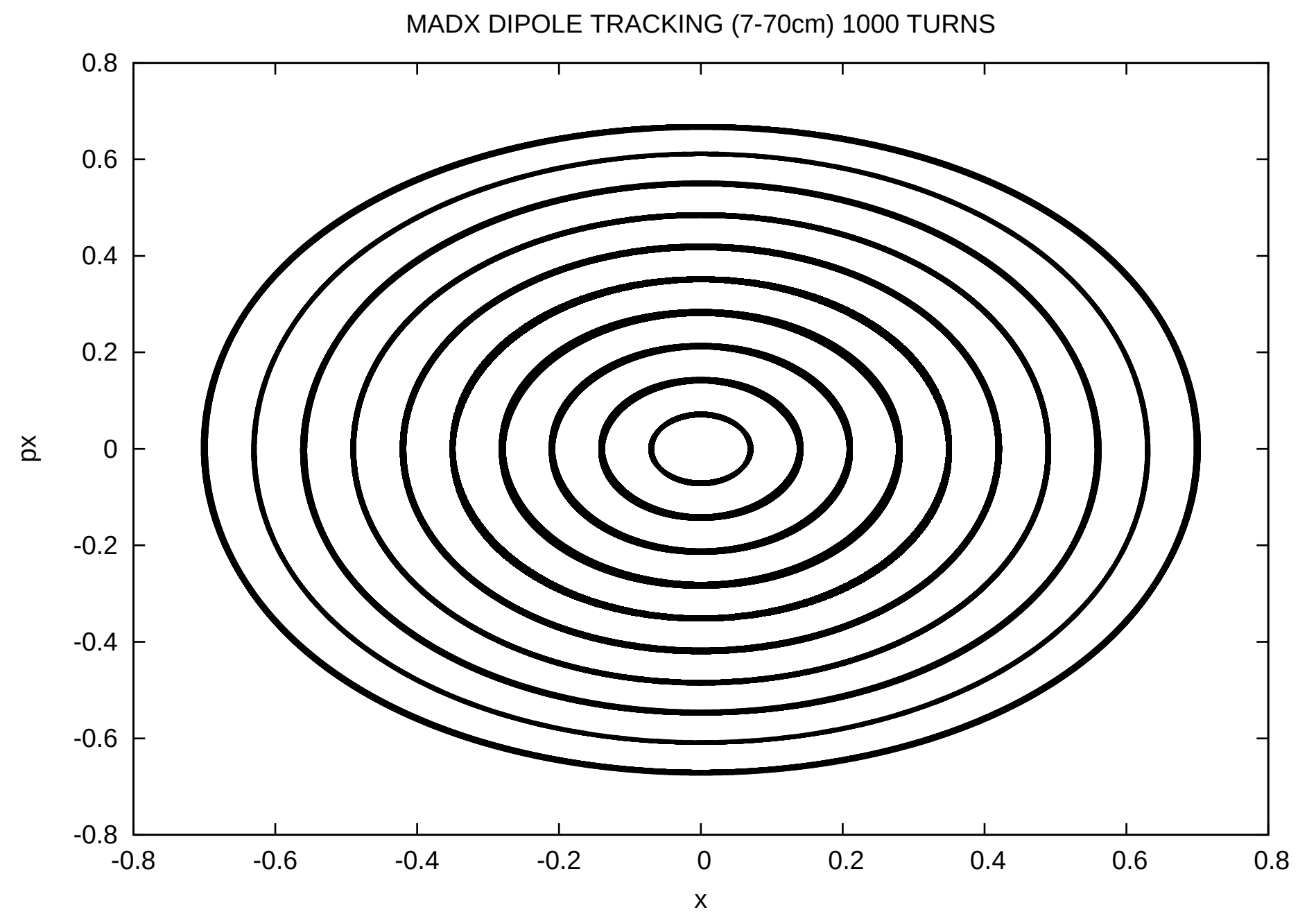


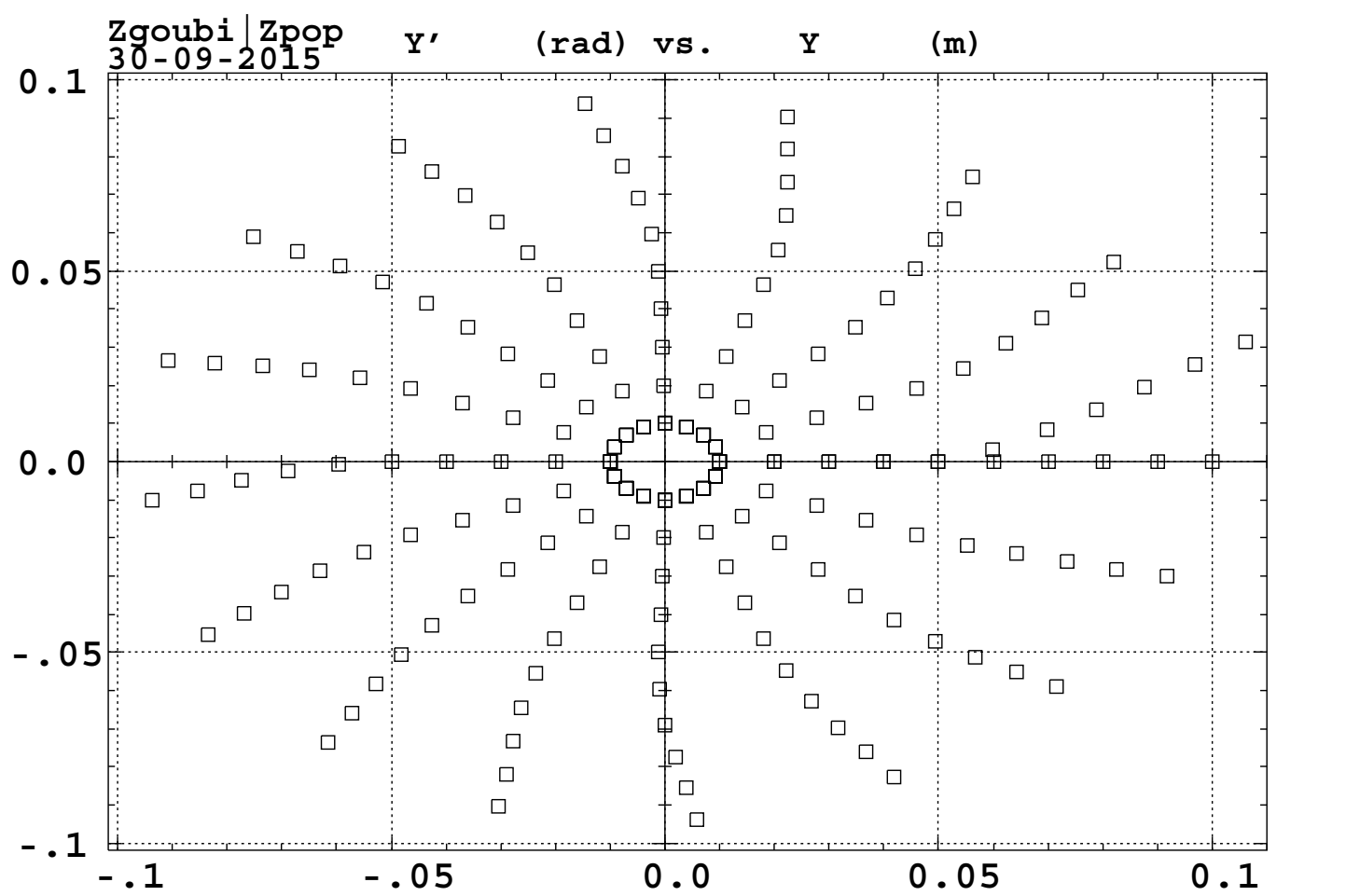


MAG DI. 22.5 DEG. MSU INTEGRATOR, REL.=1, NT=16000, STEP=11


Zgoubi|Zpop
23-09-2015
Y' (rad) vs. Y (m)
0.6
0.4
0.2
0.0
-.2
-.4
-.6
-.4
-.2
0.0
0.2
0.4
0.6
* # COORDINATES - STORAGE FILE, 23-09-2015 19:19:42 *
Mi-ma H/V: -0.568 0.748 / -0.604 0.604
Part# 1- 10000 (*); Lmnt# 1; pass# 1- 16001, [1]; 160

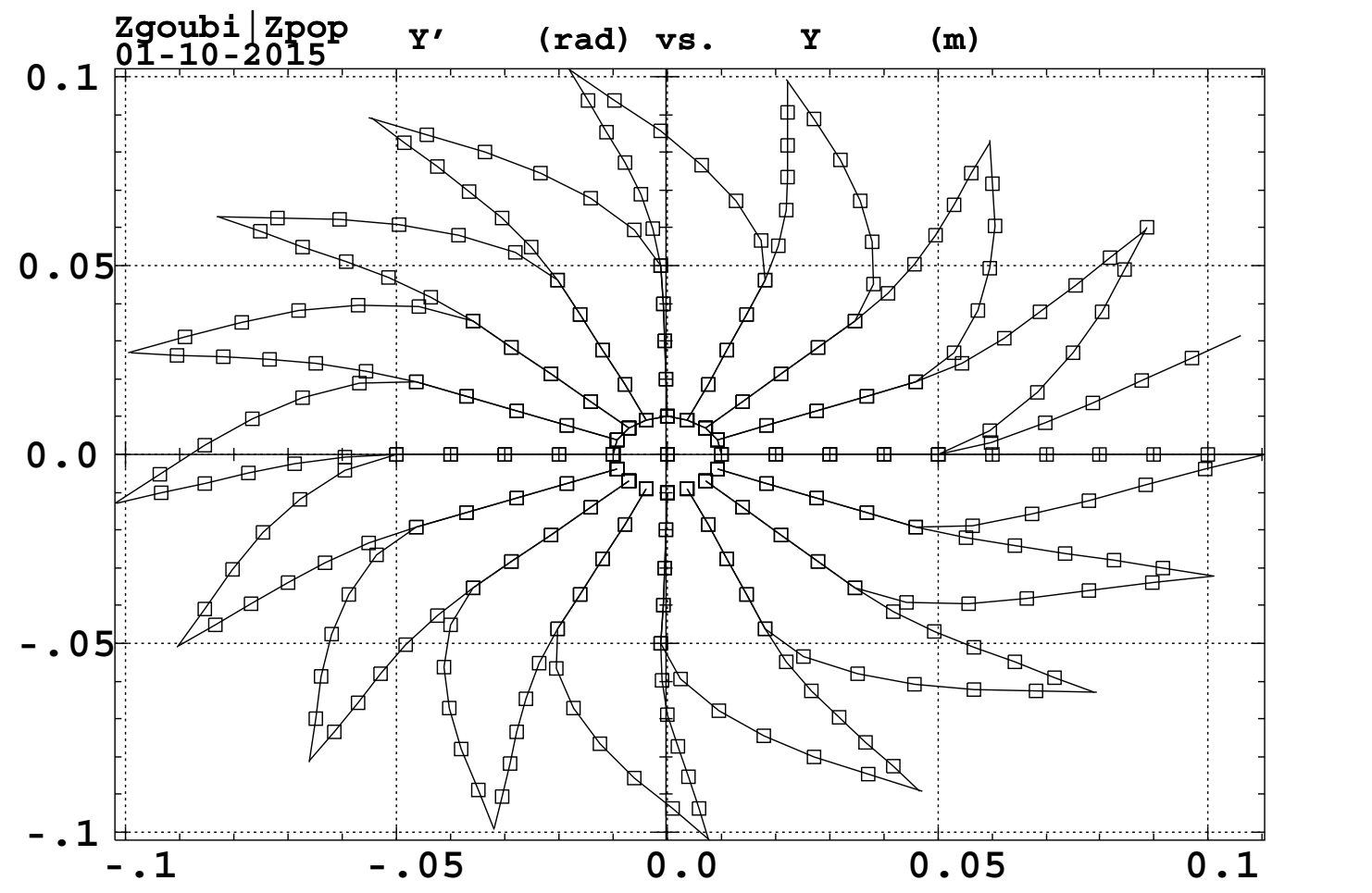

mad8(7-70cm)16000TURNS
Win32 version 8.51/15
06/10/15 21.34.31
p_x / p_0
0.8
0.6
0.4
0.2
0.0
-0.2
-0.4
-0.6
-0.8
-0.8
-0.4
0.0
0.4
0.8
x (m)


MADX DIPOLE TRACKING (7-70cm) 1000 TURNS
0.8
0.6
0.4
0.2
0
-0.2
-0.4
-0.6
-0.8
px
-0.8
-0.6
-0.4
-0.2
0
0.2
0.4
0.6
0.8
x



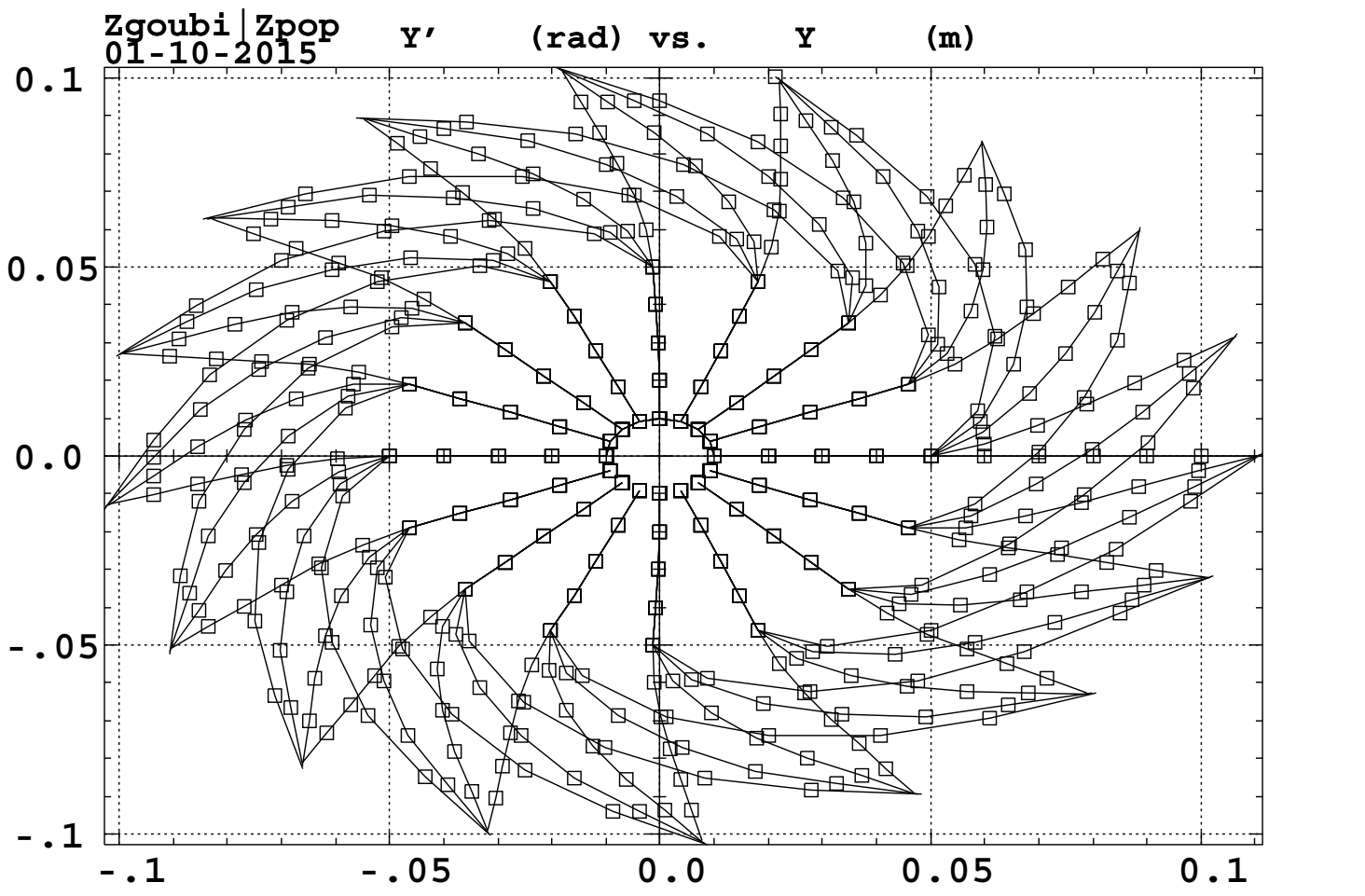
* # COORDINATES - STORAGE FILE, 30-09-2015 19:01:37 *

Mi-ma H/V: -0.102 0.110 / -0.102 0.102
Part# 1- 10000 (*); Lmnt# 1; pass# 1- 17, [1];



* # COORDINATES - STORAGE FILE, 01-10-2015 07:05:27 *

Mi-ma H/V: -0.102 0.110 / -0.102 0.102
Part# 1- 10000 (*); Lmnt# 1; pass# 1- 33, [1];



* # COORDINATES - STORAGE FILE, 01-10-2015 07:10:24 *

Mi-ma H/V: -0.103 0.111 / -0.103 0.103
Part# 1- 10000 (*); Lmnt# 1; pass# 1- 65, [1];

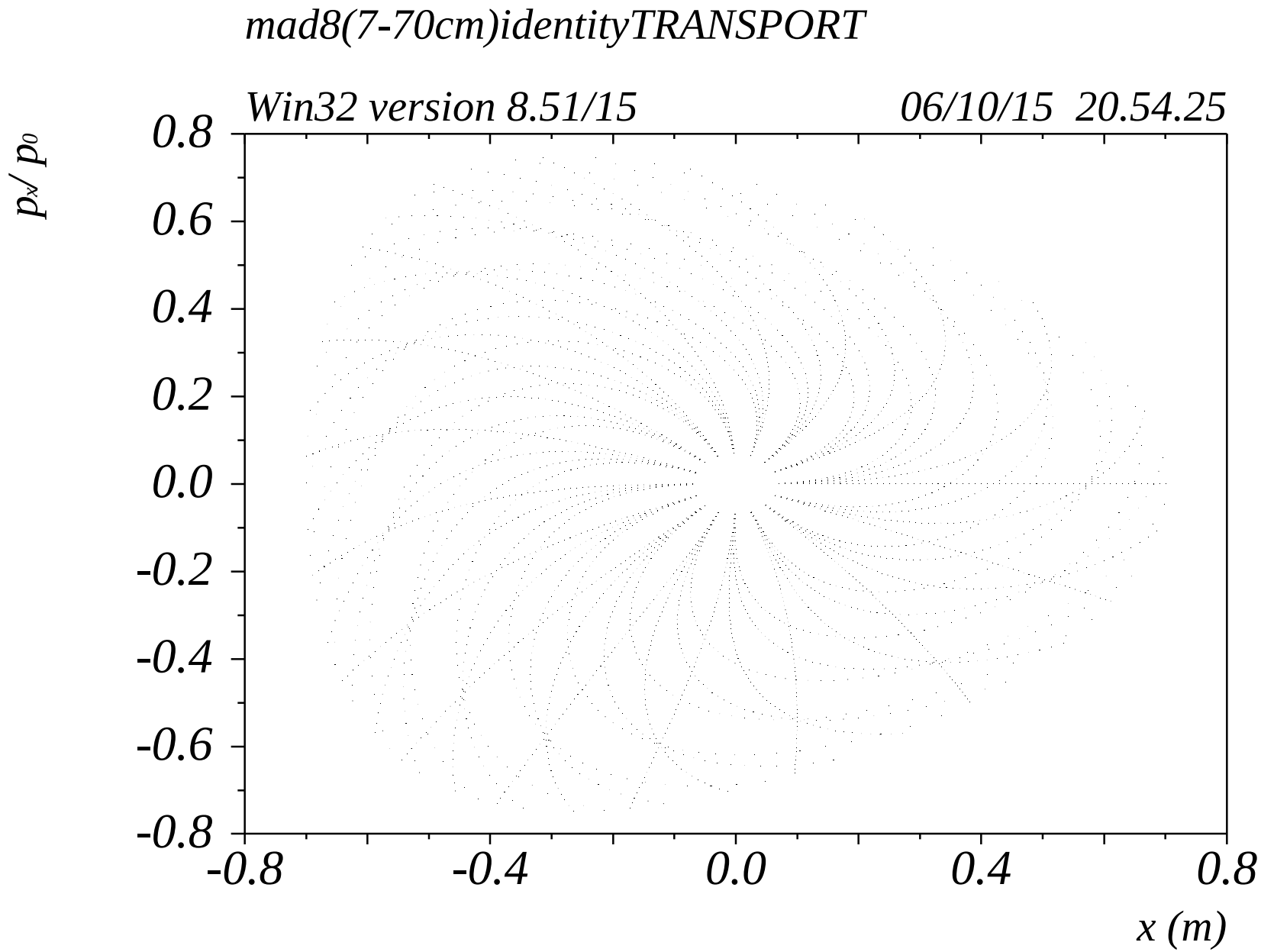
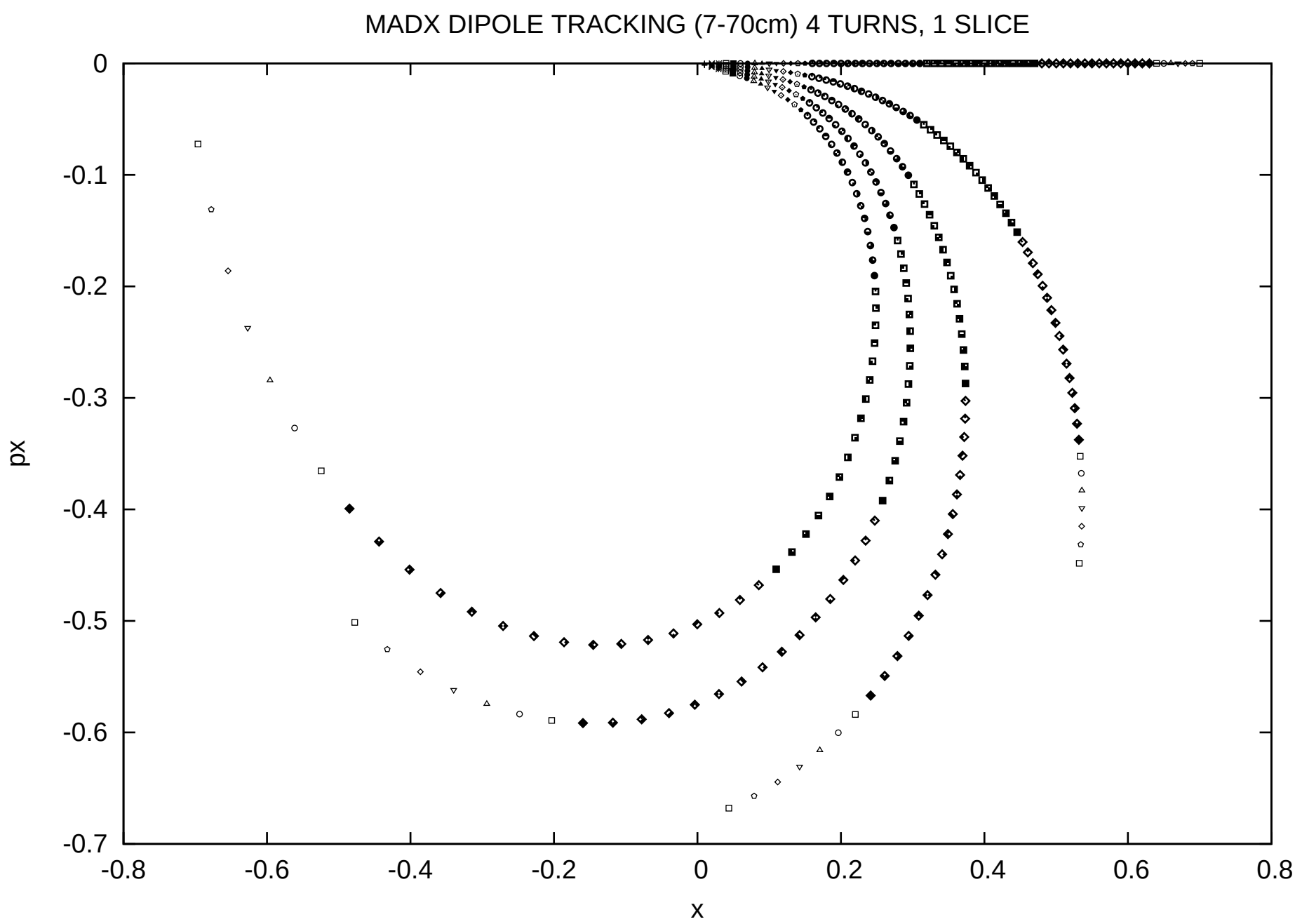
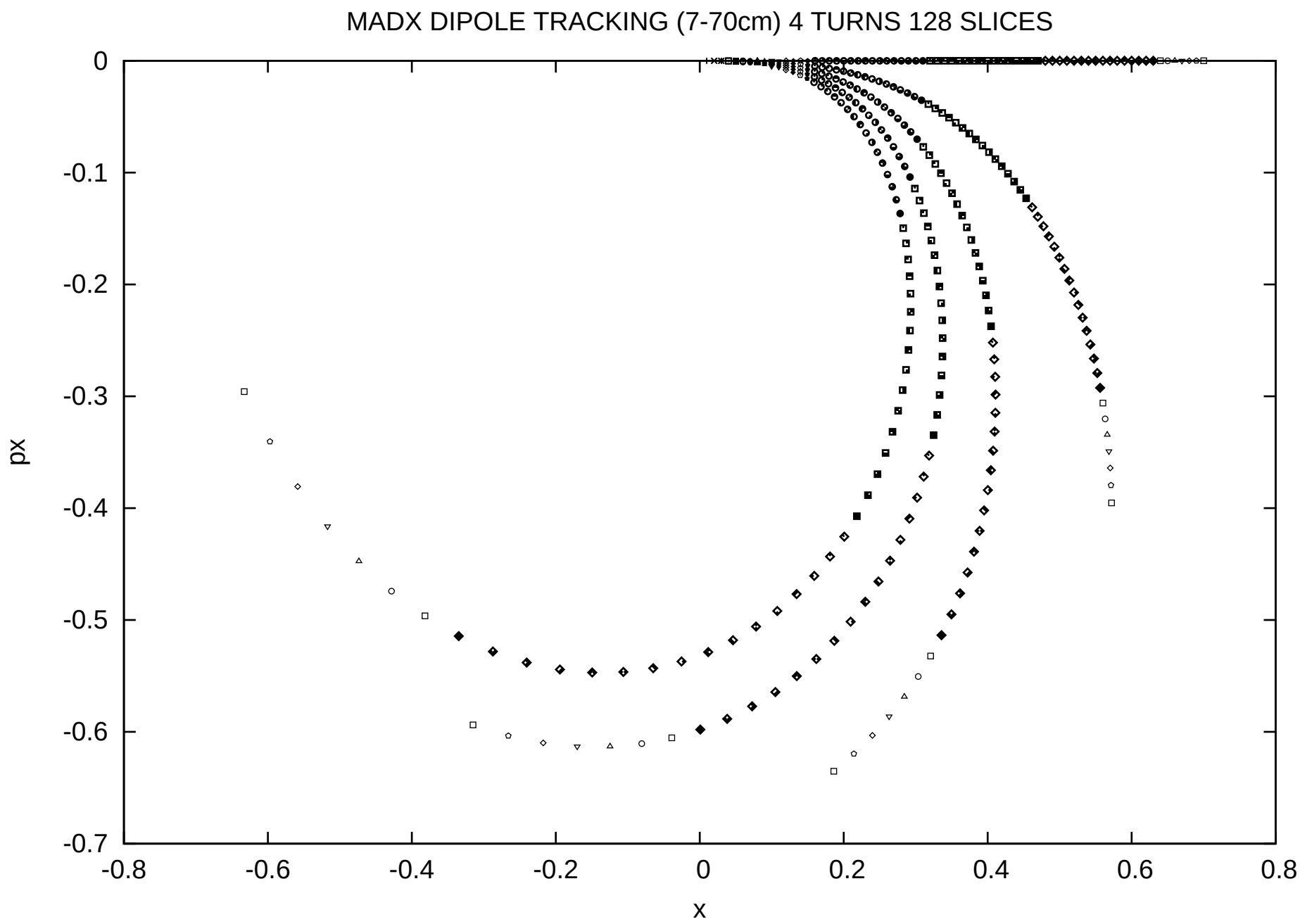
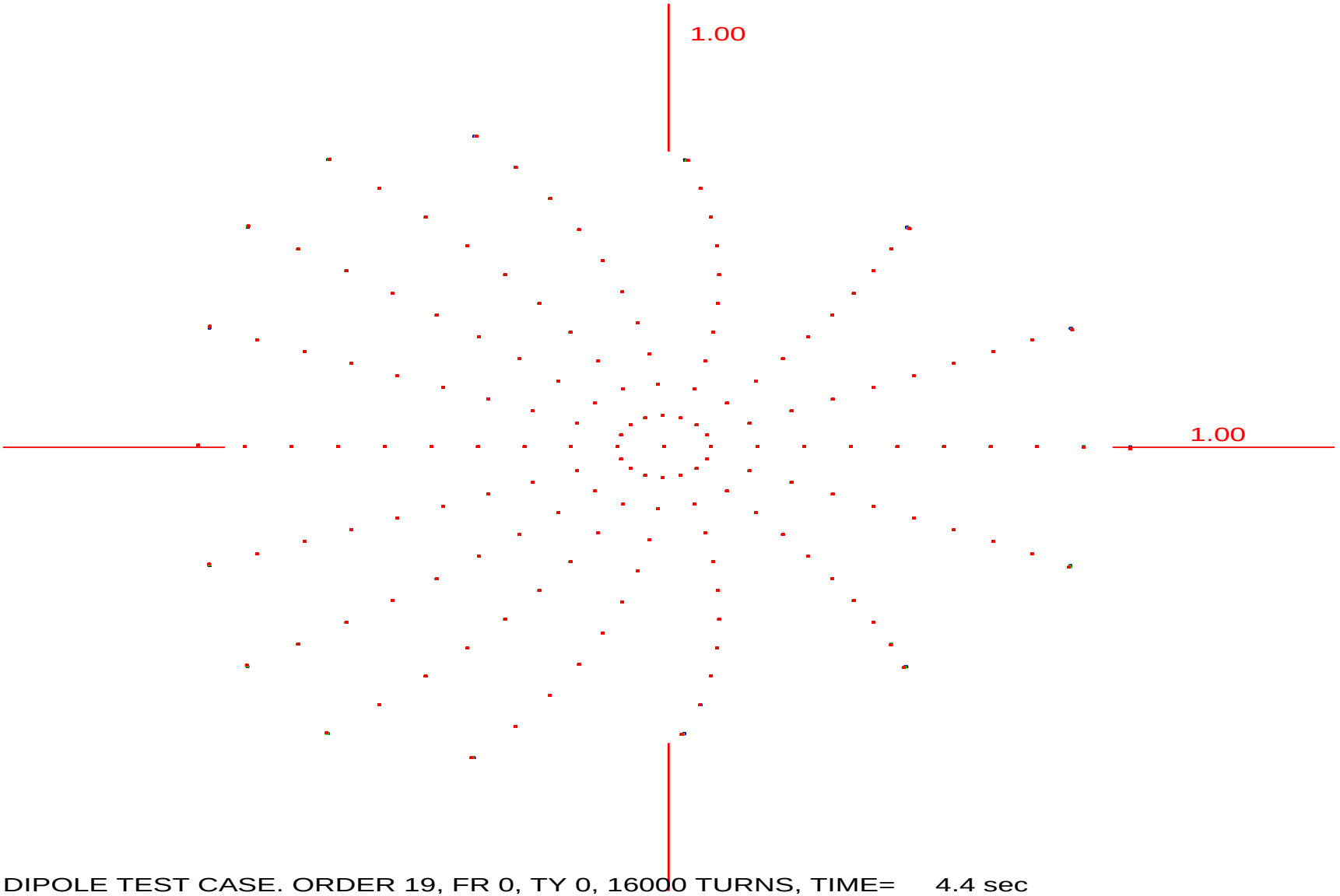


Table name = TRACKTABLE


MADX DIPOLE TRACKING (7-70cm) 4 TURNS, 1 SLICE
0
-0.1
-0.2
-0.3
-0.4
-0.5
-0.6
-0.7
px
-0.8
-0.6
-0.4
-0.2
0
0.2
0.4
0.6
0.8
x


MADX DIPOLE TRACKING (7-70cm) 4 TURNS 128 SLICES
0
-0.1
-0.2
-0.3
-0.4
-0.5
-0.6
-0.7
px
-0.8
-0.6
-0.4
-0.2
0
0.2
0.4
0.6
0.8
x



DIPOLE TEST CASE. ORDER 19, FR 0, TY 0, 16000 TURNS, TIME= 4.4 sec

The Fast Multipole Method [Greengard et al.]

- According to our particle physics colleagues, there is perfect democracy: all electrons are equal

The Fast Multipole Method (FMM)

- According to our particle physics colleagues, there is perfect democracy: all electrons are equal
- But as it turns out, some are more equal than others:

The Fast Multipole Method (FMM)

- According to our particle physics colleagues, there is perfect democracy: all electrons are equal
- But as it turns out, some are more equal than others:
- Coulomb Law Discriminates - Far away charges have less influence ☺

The Fast Multipole Method (FMM)

- According to our particle physics colleagues, there is perfect democracy, all electrons are equal
- But as it turns out, some are more equal than others:
- Coulomb Law Discriminates - Far away charges are less important!

FMM Method: To determine the field, lump together far away charges and replaces them with their multipole expansion.

The Fast Multipole Method (FMM)

- According to our particle physics colleagues, there is perfect democracy, all electrons are equal
- But as it turns out, some are more equal than others:
- Coulomb Law Discriminates - Far away charges are less important!

FMM Method: To determine the field, lump together far away charges and replaces them with their multipole expansion.

Playing this trick to the end leads to computational expense that scales linear with the number of particles:

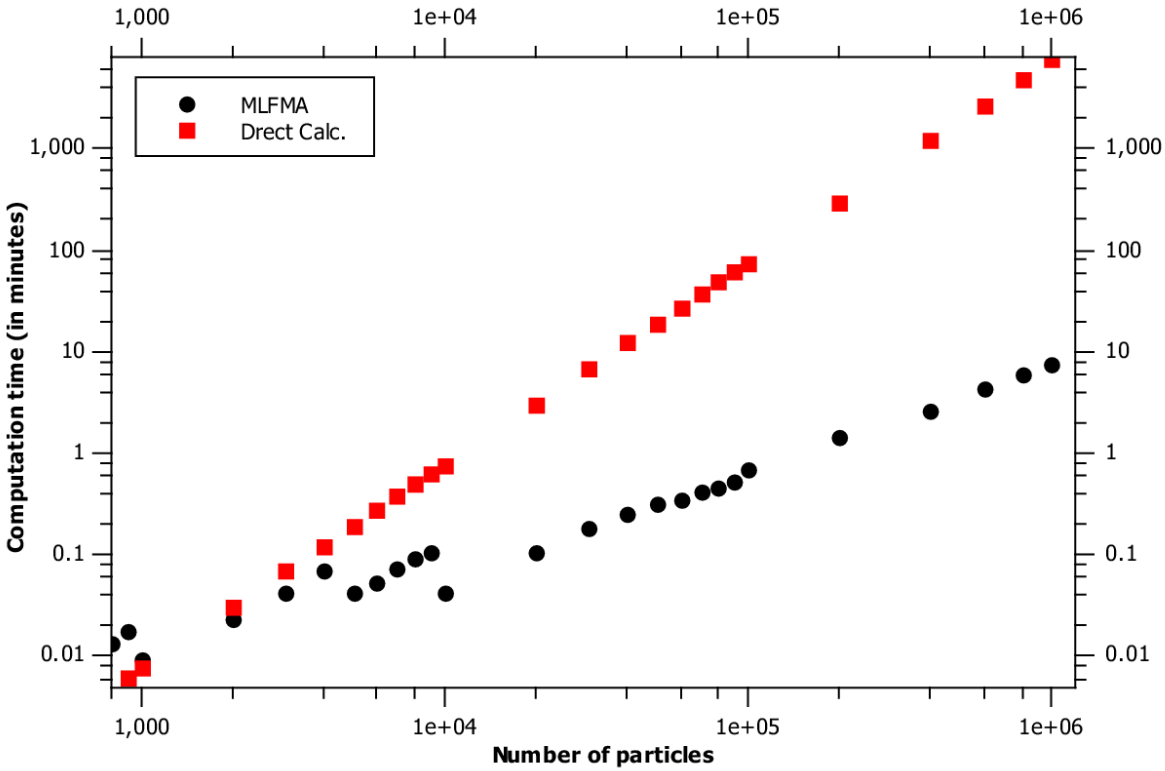


Figure Courtesy He Zhang

FMM – Basic Ideas 1

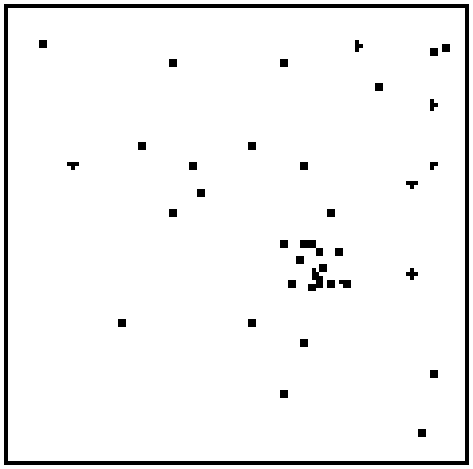
First step - Particle Scattering: Distribute the particles into a grid of variable-sized boxes.

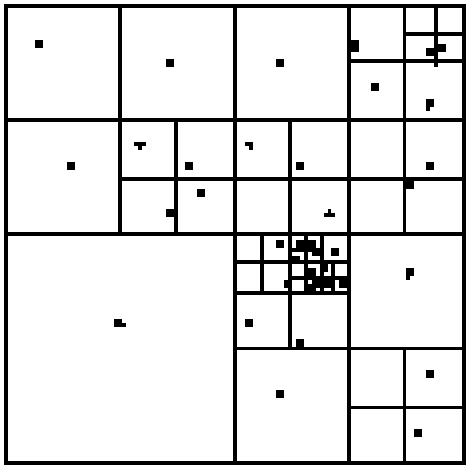
- Determine a global box that contains all N particles (Linear in N)
- While distributing particles, if number of particles in a box exceeds pre-specified maximum M, split that box into two (Linear in N)

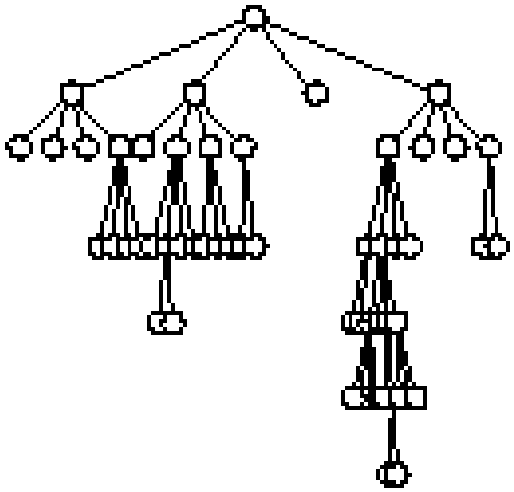
FMM – Basic Ideas 1

First step - Particle Scattering: Distribute the particles into a grid of variable-sized boxes.

- Determine a global box that contains all N particles (Linear in N)
- While distributing particles, if number of particles in a box exceeds pre-specified maximum M, split that box into two (Linear in N)
- **End result:** an arrangement of boxes of unequal size that are related to each other through a sparse tree describing split hierarchy.







FMM – Basic Ideas 2

Second Step – Compute "Far" Multipoles (J.D. Jackson-type) of each box:

- Expand potential of all particles around center of their box. Order n of expansion is an accuracy control parameter.
- In practice, expand in powers of 1/r and trig functions (conventional spherical harmonics), or in powers of 1/r, 1/x, 1/y, 1/z (more convenient)

FMM – Basic Ideas 2

Second Step – Compute "Far" Multipoles (J.D. Jackson-type) of each box:

- Expand potential of all particles around center of their box. Order n of expansion is an accuracy control parameter.
- In practice, expand in powers of 1/r and trig functions (conventional spherical harmonics), or in powers of 1/r, 1/x, 1/y, 1/z (more convenient)
- Particles can be points, Gaussians, wavelets, asymmetric, etc etc – no fundamental difference, possibly big practical difference in complexity
- Process is again linear in N, since for each particle, there is a fixed effort of determining its multipole contribution to the box.

FMM – Basic Ideas 2

Second Step – Compute "Far" Multipoles (J.D. Jackson-type) of each box:

- Expand potential of all particles around center of their box. Order n of expansion is an accuracy control parameter.
- In practice, expand in powers of 1/r and trig functions (conventional spherical harmonics), or in powers of 1/r, 1/x, 1/y, 1/z (more convenient)
- Particles can be points, Gaussians, wavelets, asymmetric, etc etc – no fundamental difference, possibly big practical difference in complexity
- Process is again linear in N, since for each particle, there is a fixed effort of determining its multipole contribution to the box.

Third Step – Compute "Near" Multipoles (Optics-type) for each box:

- Classify all other boxes as "nearby" or "distant", depending on ratio r of their diameter to distance of center points.
- For all "distant" boxes, determine resulting local expansion in x, y, z. If a parent box is distant, ignore its child boxes.
- **If potential fall-off is fast enough**, number of contributing distant boxes has fixed upper bound, resulting in process linear in N

FMM – Basic Ideas 3

In the case of **point particles**, force on each particle has two contributions:

1. Local multipoles from the box in which it resides
2. Direct Coulomb force from nearby particles

FMM – Basic Ideas 3

In the case of **point particles**, force on each particle has two contributions:

1. Local multipoles from the box in which it resides
2. Direct Coulomb force from nearby particles

In the case of **soft particles**, add forces of nearby particles into local expansion

FMM – Basic Ideas 3

In the case of **point particles**, force on each particle has two contributions:

1. Local multipoles from the box in which it resides
2. Direct Coulomb force from nearby particles

In the case of **soft particles**, add forces of nearby particles into local expansion

Properties of the FMM Method:

- No need for Particle-In-Cell, PDE solvers, etc
- Works for any particle distribution – smooth, lumpy, disk-shaped, etc etc
- No artificial smoothing
- All approximations can be quantitatively understood and estimated

FMM – Basic Ideas 3

In the case of **point particles**, force on each particle has two contributions:

1. Local multipoles from the box in which it resides
2. Direct Coulomb force from nearby particles

In the case of **soft particles**, add forces of nearby particles into local expansion

Properties of the FMM Method:

- No need for Particle-In-Cell
- Works for any particle distribution – smooth, lumpy, disk-shaped, etc etc
- No artificial smoothing
- All approximations can be quantitatively understood and estimated

Accuracy is controlled by

1. Order n of local expansion
2. Ratio r deciding “nearness/farness”

DA-FMM 1 – Help with Expansions

The use of **DA methods** can help in the FMM framework in two different ways:

First, FMM is all about expansions. But, this is what **DA does automatically** and very efficiently.

1. In the case of point particles, it simplifies the treatment and results in code not much more difficult than the one for direct particle-to-particle summation. In particular, no graduate students will be abused computing high-order expansions and coding the results.

DA-FMM 1 – Help with Expansions

The use of **DA methods** can help in the FMM framework in two different ways:

First, FMM is all about expansions. But, this is what **DA does automatically** and very efficiently.

1. In the case of point particles, it simplifies the treatment and results in code not much more difficult than the one for direct particle-to-particle summation. In particular, no graduate students will be abused computing high-order expansions and coding the results.
2. In the case of non-point particles, computing multipole expansions can be rather difficult – any volunteers for non-symmetric wavelets?
3. Using computer algebra systems does not help much since the resulting derivative code intimidates by sheer size, and is consequently also slow, especially when compared to DA code

DA-FMM 2 – The Real Virtue

In conventional FMM, a local expansion of the potentials/fields is computed for each box.

For integration of particle ensembles in time, this is then inserted in an ODE solver, possibly with step size control.

DA-FMM 2 – The Real Virtue

In conventional FMM, a local expansion of the potentials/fields is computed for each box.

For integration of particle ensembles in time, this is then inserted in an ODE solver, possibly with step size control.

But: thinking further along the FMM philosophy of local expansion, why stop at expansion in position for each time step?

DA-FMM 2 – The Real Virtue

In conventional FMM, a local expansion of the potentials/fields is computed for each box.

For integration of particle ensembles in time, this is then inserted in an ODE solver, possibly with step size control.

But: thinking further along the FMM philosophy of local expansion, why stop at expansion in position for each time step?

Why not also expand each local box forward in time?

DA-FMM 2 – The Real Virtue

In conventional FMM, a local expansion of the potentials/fields is computed for each box.

For integration of particle ensembles in time, this is then inserted in an ODE solver, possibly with step size control.

But: thinking further along the FMM philosophy of local expansion, why stop at expansion in position for each time step?

Why not also expand each local box forward in time?

Then, instead of inserting particles in local **field polynomials**, …

One inserts particles into the local high-order **transfer map polynomials** and is done with the entire time step at once.

DA-FMM 2 – The Real Virtue

In conventional FMM, a local expansion of the potentials/fields is computed for each box.

For integration of particle ensembles in time, this is then inserted in an ODE solver, possibly with step size control.

But: thinking further along the FMM philosophy of local expansion, why stop at expansion in position for each time step?

Why not also expand each local box forward in time?

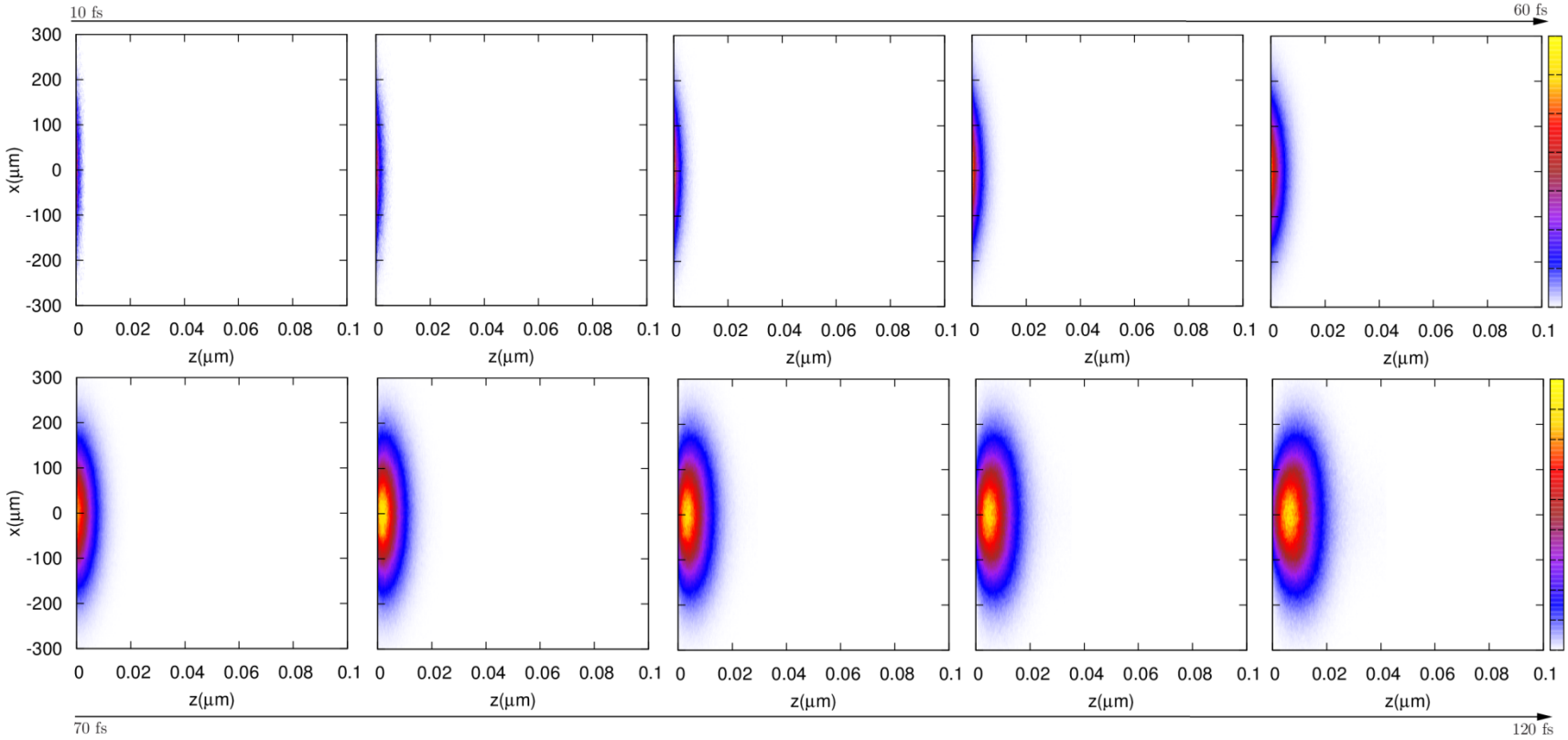
Then, instead of inserting particles in local **field polynomials**, …

One inserts particles into the local high-order **transfer map polynomials** and is done with the entire time step at once.

The DA flow operator does not mind where its right hand side comes from. As long as it is a code list, it can compute the high-order flow automatically.

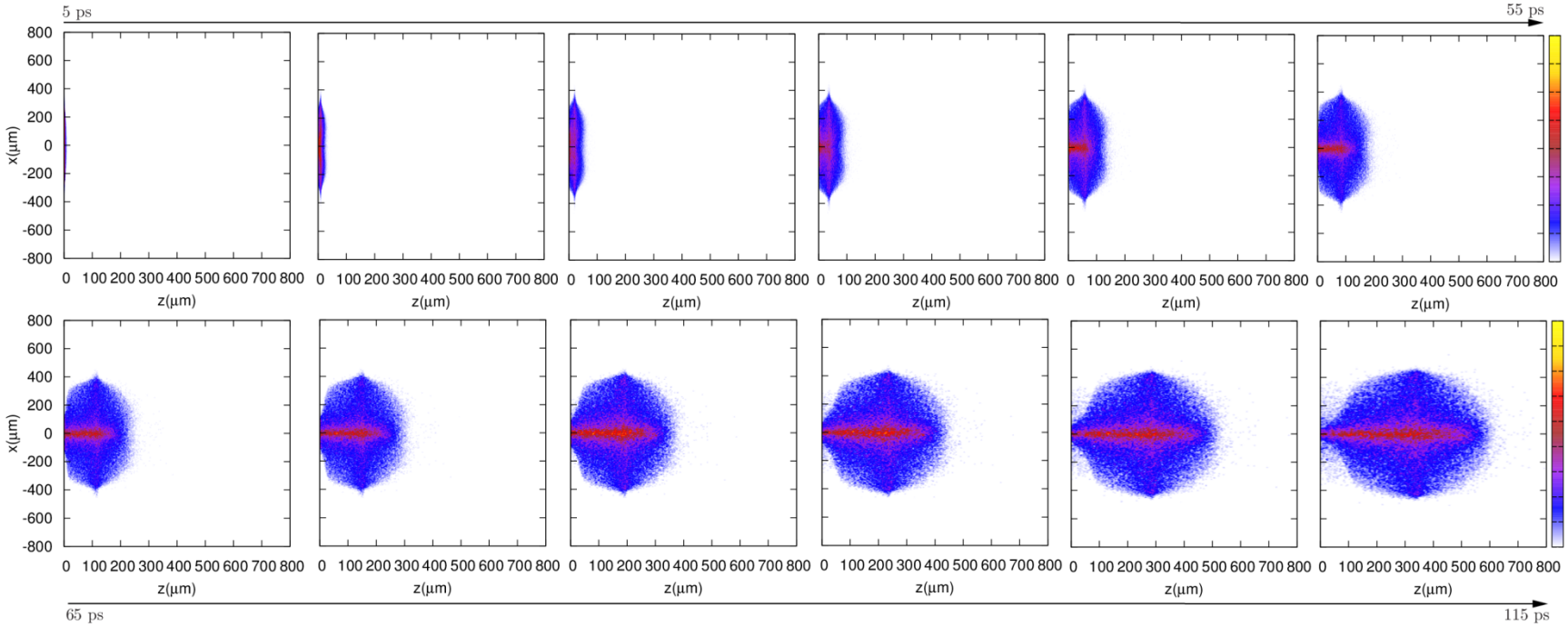
Beam Emission Simulation 1 - 3D DA-FMM

- Laser pulse is applied, Gaussian in position and time
- Extraction field of 2kV/m removes electrons from surface
- A bunch is formed and starts to move away from surface
- [All simulations from He Zhang, dissertation MSU 2012]



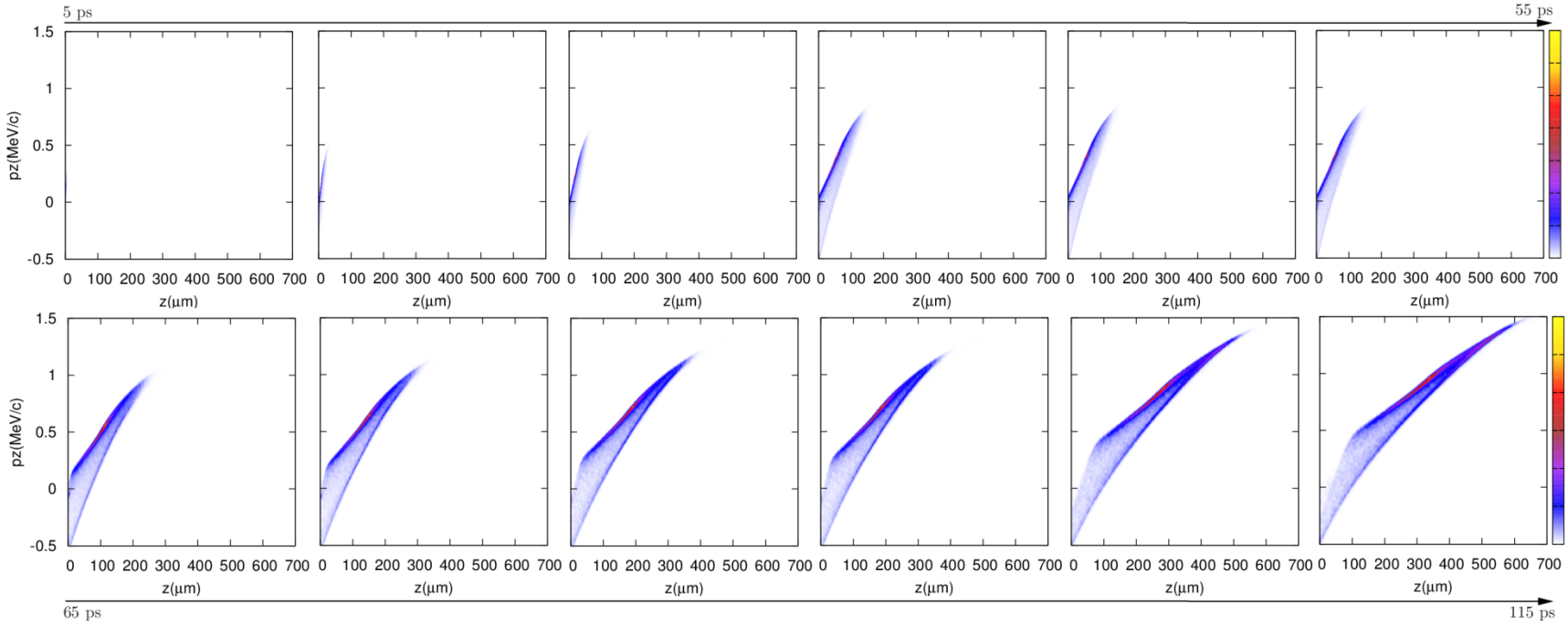
Beam Emission Simulation 2 - 3D DA-FMM

- Electron bunch develops further; here we show 115 ps total
- Some electrons are pulled back by evolving counter charge
- Eventually the beam nearly fully detaches
- Complexity of the extraction process entails irregular beam shape
- Irregular shape entails non-linear space charge field, and thus aberrations



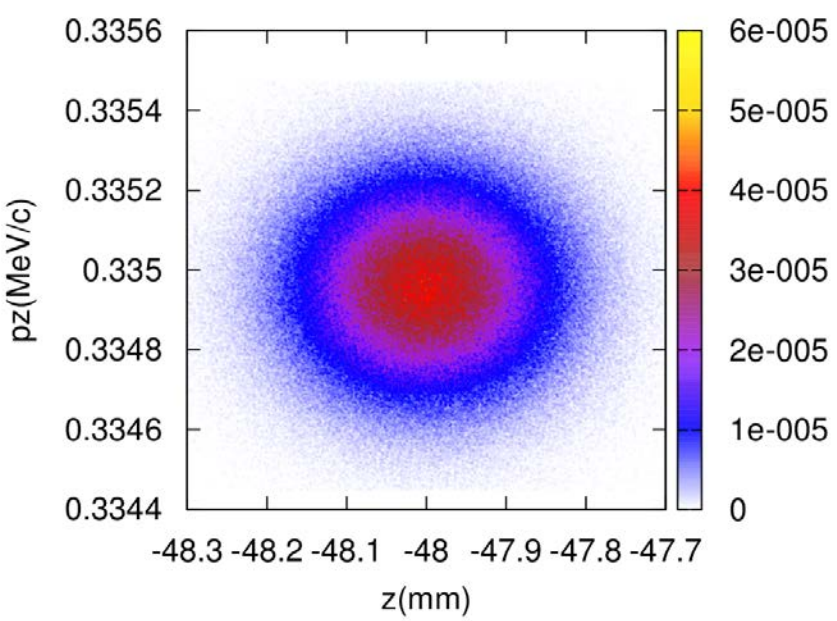
Beam Emission Simulation 3 – z-pz Projection

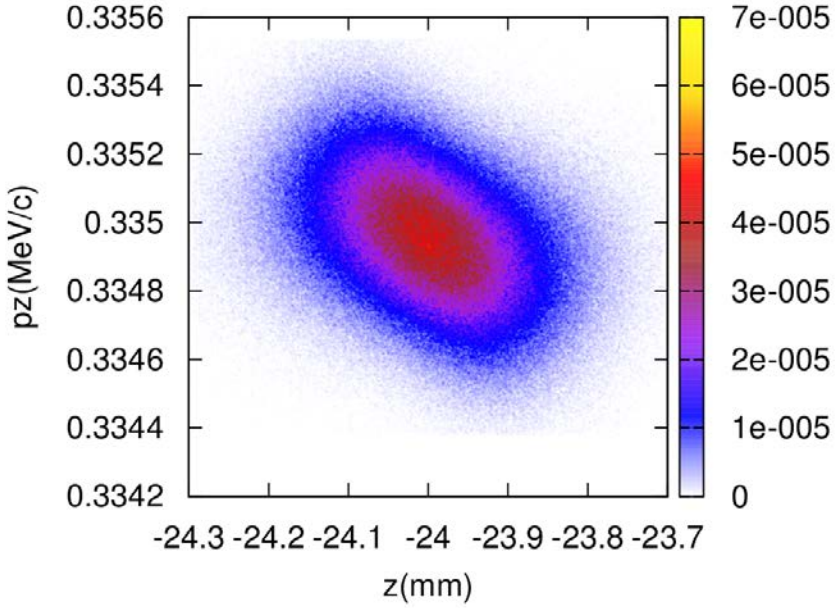
- Show projection of dynamics in z-pz space, up to 155 ps
- z increases, and correlates with pz as expected
- Linear dynamics would appear as elliptical shape
- Actual distribution is convex, and heavily clustered at the top
- Further evidence of significant nonlinear effects



Beam Dynamics Simulation – Effect of Round Lens

- During further simulation, beam bunch encounters magnetic lens
- Display z-pz motion
- Originally round shape gets severely distorted





Beam Dynamics Simulation – Effect of Round Lens