Beam Envelope Simulation with Space Charge in SAD

Kazuro Furukawa (KEK)
Christopher K. Allen (LANL)

Sep. 6, 2006

Workshop SAD2006

Contents

- **Background**
  - KEKB, J-PARC, SNS, Space Charge, Beam loss, Commissioning-tool
- **Envelope Simulation with Space Charge**
  - Linear Calculation, Assumption
- **Adaptive Stepping**
  - Find Appropriate Step for Space Charge Matrix
- **For SAD**
- **Example, Comparison**

Background

◆ Success of KEKB with SAD

- Fast Commissioning Tool was a Primary Concern
  - to Compete with SLAC/PEPII
- Pre-SAD
  - Data Collection - Data Manipulation - Compare/Fit to Simulation - Feedback to Machine
  - In Several Programs, by Several persons, may take a week
- SAD
  - In one Panel, by one person, in a minute
  - All-in-one (All but Kitchen-sink)
    - Accelerator Modeling, Machine Controls, Data Archives, Data Manipulations, GUI
  - Anyone can Write
    - List-oriented (Mathematica-like) Scripting Language
  - Was Quicker to Achieve Higher Luminosity

Background

J-PARC

- Fast Commissioning Tool Again
  - Determine/Calibrate Accelerator Equipment
  - Optimize Parameters one-by-one
  - Quicker Turn-around
- Space Charge Calculation is Expensive
  - Linear Optics vs. Space Charge
  - Envelope Simulation vs. Tracking Simulation
  - At least Linac need Space Charge Handling from the Beginning
    - Peak Current cannot be Reduced, only Pulse Width can be reduced
    - RCS/MR may start with Linear Optics (?)
- SNS
  - Adaptive Envelope Simulation under XAL/Java Environment
- Possible J-PARC Strategy(?), with Online and Offline Models
  - Envelope Online Tools for Commissioning
  - Tracking Offline Tools for Detailed Beam-loss Estimation

Background

- **Chance to Invite Christopher K. Allen**
  - **Experience to Develop Envelope Simulation**
  - **XAL/Java Environment**
  - **Same Method under SAD(?)**
  - **Possible Application to Electron Machines(?)**

Beam Simulation Overview

◆Extension of Linear Beam Optics

❖ In a straightforward manner, the linear beam optics model for single particle dynamics can be extended to the dynamics for the second moments of the beam.

❖ For intense beams, space charge effects are significant and must be included. For a beam optics model, this means a matrix Φ_{sc} that accounts for space charge (linear force!). It is accurate only over short distance.

❖ For ellipsoidally symmetric beams, we can produce such a Φ_{sc} that is almost independent of the actual beam profile.

Beam Simulation Overview

- **In the SAD environment we are given the full transfer matrix Φ_n for each element n. We must take the N^{th} root of each Φ_n where $N = L_n/\Delta s$ is the number of space charge "kicks" to be applied within the element.**

- **Propagate moment matrix σ through each element using above transfer matrix and the space charge matrix Φ_{sc} computed for each step Δs**

- **Since the second moments depend upon Φ_{sc} and Φ_{sc} depends upon the second moments, we have self-consistency issues. We employ an adaptive propagation algorithm that maintains certain level of consistency.**

Envelope Simulation

- **RMS envelope simulation is based on the following:**
  - **Phase space coordinates** $\mathbf{z} = (x\ x'\ y\ y'\ z\ dp)^T$
  - **Linear beam optics - transfer matrices** $\mathbf{z}_{n+1} = \Phi_n\, \mathbf{z}_n$
  - **Moment operator** $\langle\cdot\rangle$, $\langle g\rangle \equiv \int g(\mathbf{z}) f(\mathbf{z})\, d^6\mathbf{z}$
  - **Moment matrix** $\sigma = \langle \mathbf{z}\mathbf{z}^T \rangle$
  - **Propagation of moment matrix** $\sigma_{n+1} = \Phi_n \sigma_n \Phi_n^T$

Implementation under SAD

◆ Initialization

❖ We can obtain $\{\Phi_n\}$ and $\{L_n\}$, the lengths of the elements, from calls to the SAD environment

$\{\Phi_n\}$ = TransferMatrices/.Emittance[Matrix->True];
$\{L_n\}$ = LINE["LENGTH"];

❖ The initial moment matrix σ_0 is built from the initial Twiss parameters
σ_0 = CorrelationMatrix6D[$\{\alpha,\beta,\gamma\}_x$, $\{\alpha,\beta,\gamma\}_y$, $\{\alpha,\beta,\gamma\}_z$]

$$\sigma_0 = \begin{pmatrix} \beta_x\tilde{\varepsilon}_x & -\alpha_x\tilde{\varepsilon}_x & 0 & 0 & 0 & 0 \\ -\alpha_x\tilde{\varepsilon}_x & \gamma_x\tilde{\varepsilon}_x & 0 & 0 & 0 & 0 \\ 0 & 0 & \beta_y\tilde{\varepsilon}_y & -\alpha_y\tilde{\varepsilon}_y & 0 & 0 \\ 0 & 0 & -\alpha_y\tilde{\varepsilon}_y & \gamma_y\tilde{\varepsilon}_y & 0 & 0 \\ 0 & 0 & 0 & 0 & \beta_z\tilde{\varepsilon}_z & -\alpha_z\tilde{\varepsilon}_z \\ 0 & 0 & 0 & 0 & -\alpha_z\tilde{\varepsilon}_z & \gamma_z\tilde{\varepsilon}_z \end{pmatrix}$$

Implementations

◆Sub-Dividing Beamline Elements (the N^{th} root of Φ_n)

❖ **The transfer matrix Φ_n for an element n has the form**

$$\Phi_n = \exp(L_n \mathrm{F}_n)$$

where L_n is the length of the element and F_n is the *generator matrix* which represents the external forces of element n.

❖ **To sub-divide element n, we require the matrix F_n, given by**

$$\mathrm{F}_n = \log(\Phi_n)/L_n$$

❖ **The "sub-transfer matrix" $\Phi_n(\Delta s)$ for element n can then be computed as**

$$\Phi_n(\Delta s) = \exp(\Delta s \mathrm{F}_n)$$

Implementations

◆Transfer Matrices with Space Charge

❖Whether using the equations of motion or Hamiltonian formalism, within a section Δs of a element n we can write the first-order continuous dynamics as

$$\mathbf{z}'(s) = \mathbf{F}_n \mathbf{z}(s) + \mathbf{F}_{sc}(\sigma)\mathbf{z}(s)$$

❖ where the matrix $\mathbf{F}_n$ represents the external force of element n and $\mathbf{F}_{sc}(\sigma)$ is the matrix of space charge forces.

❖For $\mathbf{F}_{sc}(\sigma)$ constant, the solution is $\mathbf{z}(s) = \exp[s(\mathbf{F}_n + \mathbf{F}_{sc})]\mathbf{z}_0$.

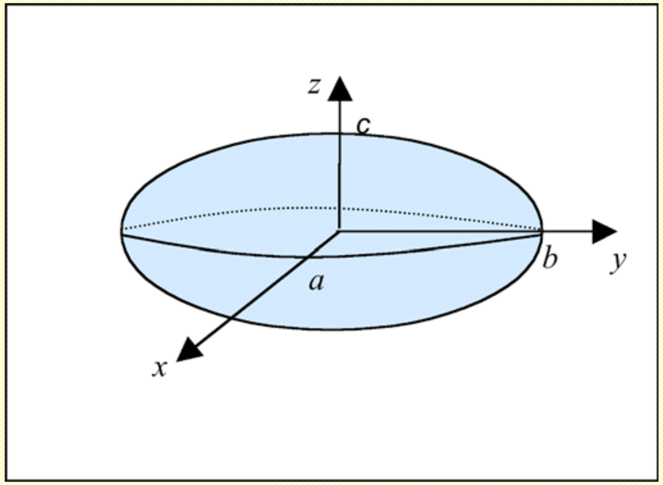
❖Thus, the full transfer matrix **including space charge** should be

$$\Phi_n = \exp[\Delta s(\mathbf{F}_n+\mathbf{F}_{sc})]$$

Field Calculations

- **Space charge effects are included by assuming the beam has ellipsoidal symmetry with dimensions corresponding to the statistics in σ.**

$$f(\mathbf{z}) = f(\mathbf{z}^T \sigma^{-1} \mathbf{z})$$



- **Analytic field expressions for such a bunch distributions are available**

$$\phi(x,y,z) = \frac{qabc}{4\varepsilon_0} \int_0^\infty \int_{\frac{x^2}{t+a^2}+\frac{y^2}{t+b^2}+\frac{z^2}{t+c^2}}^\infty \frac{f(s)}{(t+a^2)^{1/2}(t+b^2)^{1/2}(t+c^2)^{1/2}} \, ds\,dt$$

where *a, b, c*, are the semi-axes of the ellipsoid (depends upon σ) and (*x,y,z*) are the coordinates along the semi-axes

- **Second-order Accurate Transfer-Matrix can be generated**

Stepping

◆Approach

❖ Form Form a transfer matrix $\Phi(s;s_0)$ that includes space effects to second order (2nd order accurate)

❖Choose error tolerance ε in the solution (~ 10^{-5} to 10^{-7})

❖Use $\Phi(s;s_0)$ to propagate τ in steps h whose length is determined adaptively to maintain ε

Codeing Example

```
!---------------------------------
!
!  MODULE ScheffTest
!
!---------------------------------
!
!
! Module for testing the envelope space charge routines in
!   SADScript.  Specifically for testing  the functions found
!   in the packages
!
!      Scheff.n
!      Trace3dToSad.n
!      TwissUtility.n
!
! Currently the file is set up to simulation the J-PARC transport
!   line at 181 MeV.
!
!    Author    : Christopher K. Allen
!    Created   : November, 2005
!

!!========================================
!!
!!  Initialize SAD
!!
!!========================================

FFS;                    ! Begin SADScript

!
!  GLOBAL CONSTANTS
!

strBeamline = "L3BT01all";        ! beamline
strFileOut = "ScheffTestOut.txt"   ! output file name

!
!                      Load Beamline
!

GetMAIN["~ckallen/J-Parc/linac/simdb-LI_L3BT01-nopmq0000.sad"];
!GetMAIN["~ckallen/J-Parc/linac/simdb-NoBends.sad"];
L3BT01 = ExtractBeamLine["L3BT01all"];
!
!                      Initialize SAD Environment
!

USE L3BT01;
TRPT;
INS;
CAL;

NOCOD;
RFSW;

$DisplayFunction = CanvasDrawer;



!
!                      Define the Initial Beam Particle
!

!MASS   = 0.939294 GEV;
!CHARGE = -1;
!MOMENTUM = 0.610624 GEV;

!InitialOrbits =  {{ 0.0,0.0,0.0,0.0,0.0,0.0 }};
```

```
!---------------------------------------
!
! FUNCTION  SaveMatrix
!
!---------------------------------------
!
! Function saving an arbitrary matrix to persistent storage.
!
! Parameters
!   strFile   file name to store matrix
!   mat       matrix to be stored
!
! Returned Value
!   None
!
!
!   Author  :  Christopher K. Allen
!   Created :  November, 2005
!

SaveMatrix[strFile_, mat_] := Module[
  {
  dims,         ! matrix dimensions vector
  M,            ! number of rows
  N,            ! number of columns
  m,            ! loop control - rows
  n,            ! loop control - columns
  fos           ! file output stream
  },

  dim = Dimensions[mat];
  M = dim[[1]];
  N = dim[[2]];

  fos = OpenWrite[strFile];
  Write[fos, "Matrix Dimensions ", M, "x", N];

  For[m=1, m<=M, m++,
    For[n=1, n<=N, n++,

      WriteString[fos," : ",  mat[[m,n]]]
    ];
    Write[fos, " : "];
  ];
  Close[fos];
];
```

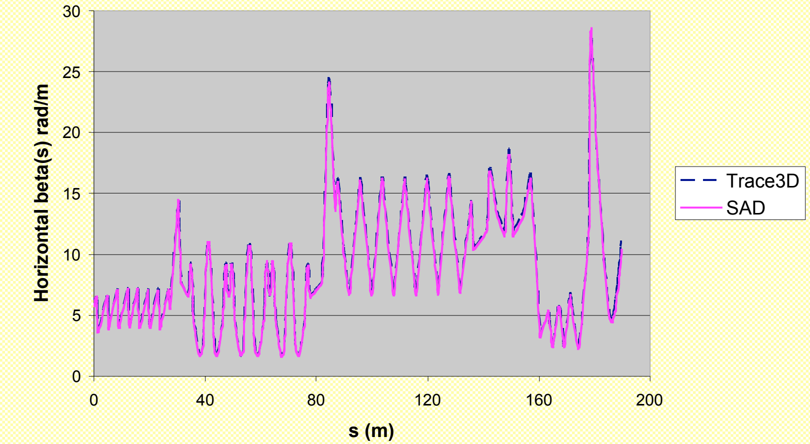
```
!
! INITIALIZE SIMULATION
!


!
!   TRACE3D Parameters
!

f  = 324.0e6;        ! RF frequency (Hz)

Er = 939.29432e6;    ! particle rest energy (eV)
W  = 181.0338e6;     ! beam kinetic energy (eV)
XI = 30.0e-3         ! beam current (A)

vecTwissXt3d = {-0.44117, 5.774, 1.889};
vecTwissYt3d = {0.21808, 6.4229, 1.706};
vecTwissZt3d = {0.3095, 2.0888, 466.99};


!
!                    Numerical Parameters
!

h0 = 0.01;                        ! initial step length
errSoln = 1.0e-5;    ! solution error tolerance
hmax = 0.0;                       ! maximum step length (=0 turned off)
hslack = 0.05;                    ! adaptive step backlash tolerance


!
!   Convert to SAD Parameters
!

Q  = XI/f;                ! beam bunch charge (C)


vecTwissX = TraceToSadTransTwiss[vecTwissXt3d];
vecTwissY = TraceToSadTransTwiss[vecTwissYt3d];
vecTwissZ = TraceToSadLongTwiss[f, Er, W, vecTwissZt3d];
```

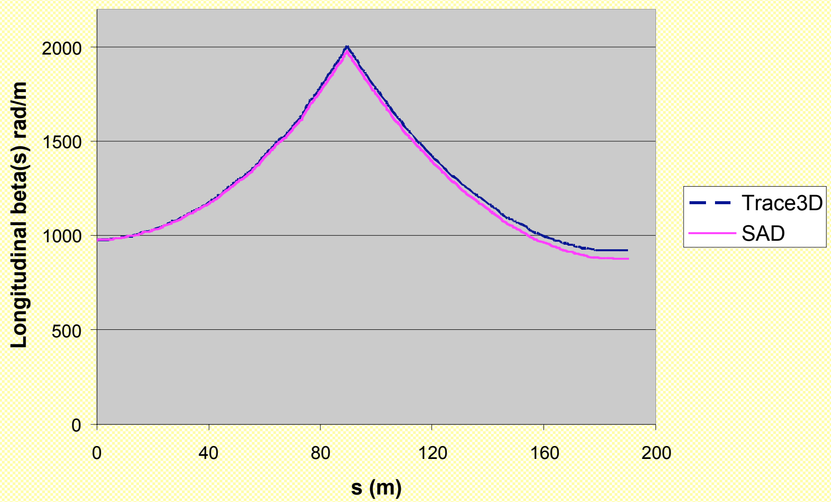
```
!
!  RUN SIMULATION
!

! Compute generalized perveance and initial moment matrix

K0   = ComputePerveance[f, Er, W, Q];
sig0 = CorrelationMatrix6D[vecTwissX, vecTwissY, vecTwissZ];


! Run simulation
!{lstPos, lstGamma, lstSig} = ScheffSimulate[K0, sig0];
{lstPos, lstGamma, lstSig} = ScheffSimulate[K0, sig0, h0, errSoln, hslack,
hmax];

! Store results
SaveBeamMatrixData[strFileOut, lstPos, lstGamma, lstSig];

!  Look at the Results
PlotBeamBeta[lstPos, lstSig];

!{lstPos, lstGamma, lstTm} = GetBeamlineElementData[];
!TmRf = lstTm[[161]];
!posRf = lstPos[[161]];
!SaveMatrix["SadRfGapMatrix.txt",TmRf];

Exit[];
```

Comparison to Trace3D

◆Simulation Test

- J-PARC Beam Transport Line from Linac to RCS
  - 181MeV, 30mA
- Good Agreement
  - Small Discrepancy
    - Symplectic transfer matrix
    - Adaptive Stepping





Summary

- **Envelope Simulation with Space Charge was Implemented in SAD Environment**

- **There are Several Other Efforts**

- **Application to Electron is Rather Difficult with Envelope**
- **Oide-san's Poisson Solver is Possible**