Bunched-Beam Envelope Simulation with Space Charge within the SAD Environment

Christopher K. Allen
Los Alamos National Laboratory

Abstract

The capability for simulating the envelopes of three-dimensional (bunched) beams has been implemented in the SAD accelerator modeling environment. The simulation technique itself is similar to that of other common envelope codes, such as Trace3D, TRANSPORT, and the XAL online model. Specifically, we follow the second-order statistics of the beam distribution rather than tracking individual particles. If we assume that the beam maintains ellipsoidal symmetry in phase space, we can include the first order effects due to space charge using a semi-analytic model. This is the attractive characteristic of envelope codes, since it greatly reduces computational time. This new feature of SAD is implemented primarily in the SADScript interpreted language, with only a small portion appearing as compiled code. As such, the simulation does run slower than other compiled envelope codes such as TRACE3D or XAL, however, as interpreted code it does have the benefit of being easily modified. We demonstrate use of the new feature and present example simulations of the J-PARC linear accelerator section.

Outline

1. Overview

Motivation

To have envelope simulation capability for three-dimensional (bunched) beams, including space-charge, within the SAD environment.

Such an engine is useful for\

- Model reference (Fast)
- Low energy electron simulation
- Proton simulation
- Longitudinal effects

1. Overview (cont.)

RMS Envelope – Approach Used Within SAD Environment

- The simulation principle is that same as that used by Trace3D and TRANSPORT. Specifically, it is an extension of linear beam optics to the second-order moment dynamics.
- For a beam optics model we require a matrix Φ_{sc} to account for the linear part of the space-charge force, it is accurate only over short distances Δs.
- 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.
- The space charge matrix Φ_{sc} depends upon the second moments, however, by the dynamics equations the second moments σ depend upon Φ_{sc}. Thus, we have self-consistency issues and must employ a propagation algorithm that maintains a certain level of consistency.

2. Envelope Dynamics Review

RMS envelope simulation is based on the following:

- Phase space coordinates $\mathbf{z} = (x\ x'\ y\ y'\ z\ dp)^{\mathrm{T}}$
- 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 $\boldsymbol{\sigma} = \langle \mathbf{z}\mathbf{z}^T\rangle$
- Propagation of moment matrix $\boldsymbol{\sigma}_{n+1} = \Phi_n\boldsymbol{\sigma}_n\Phi_n{}^T$

2. Envelope Dynamics Review (cont.)

- The moment matrices $\{\sigma_n\}$ propagate down the beamline according to

$$\sigma_{n+1} = \Phi_n \sigma_n \Phi_n^T$$

where the $\{\Phi_n\}$ are transfer matrices for each **lattice elements**

- To include space charge effects we must determine the self forces (from σ) then augment the dynamics $\sigma_{n+1} = \Phi_n \sigma_n \Phi_n^T$ accordingly.

- The quantity $\sigma \in \Re^{6\times 6}$ is the matrix of second-order moments of the beam distribution given by

$$\boldsymbol{\sigma} = \langle \mathbf{z}\mathbf{z}^T \rangle = \begin{pmatrix} \langle x^2 \rangle & \langle xx' \rangle & \langle xy \rangle & \langle xy' \rangle & \langle xz \rangle & \langle xdp \rangle \\ \langle xx' \rangle & \langle x'^2 \rangle & \langle x'y \rangle & \langle x'y' \rangle & \langle x'z \rangle & \langle x'dp \rangle \\ \langle xy \rangle & \langle x'y \rangle & \langle y^2 \rangle & \langle yy' \rangle & \langle yz \rangle & \langle ydp \rangle \\ \langle xy' \rangle & \langle x'y' \rangle & \langle yy' \rangle & \langle y'^2 \rangle & \langle y'z \rangle & \langle y'dp \rangle \\ \langle xz \rangle & \langle x'z \rangle & \langle yz \rangle & \langle y'z \rangle & \langle z^2 \rangle & \langle zdp \rangle \\ \langle xdp \rangle & \langle x'dp \rangle & \langle ydp \rangle & \langle y'dp \rangle & \langle zdp \rangle & \langle dp^2 \rangle \end{pmatrix}$$

3. SADScript Implementation

Overview

- Register the beamline with the SAD environment using GetMAIN[*latticeFile*] where *latticeFile* is the input deck describing the machine
- Acquire the set of transfer matrices $\{\Phi_n\}$ and lengths $\{L_n\}$ for all beamline elements from the SAD environment
- Take the N^{th} root of each transfer matrix where N is the number of space charge “kicks” to be applied within the element. This is done using the matrix logarithm function.
- Propagate moment matrix σ through each element using above transfer matrix and the space charge matrix Φ_{sc} computed for each step Δs

3. SADScript Implementation (cont.)

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 $\boldsymbol{\sigma}_0$ is built from the initial Twiss parameters

  $\boldsymbol{\sigma}_0$ = CorrelationMatrix6D[$\{\alpha,\beta,\gamma\}_x$, $\{\alpha,\beta,\gamma\}_y$, $\{\alpha,\beta,\gamma\}_z$]

$$
\boldsymbol{\sigma}_0 = \begin{pmatrix}
\beta_x\tilde{\varepsilon}_x & -\alpha_x\tilde{\varepsilon}_x & 0 & 0 & 0 & 0 \\
-\alpha_x\tilde{\varepsilon}_x & \gamma_x\tilde{\varepsilon}_x & 0 & 0 & 0 & 0 \\
0 & 0 & \beta_y\tilde{\varepsilon}_y & -\alpha_y\tilde{\varepsilon}_y & 0 & 0 \\
0 & 0 & -\alpha_y\tilde{\varepsilon}_y & \gamma_y\tilde{\varepsilon}_y & 0 & 0 \\
0 & 0 & 0 & 0 & \beta_z\tilde{\varepsilon}_z & -\alpha_z\tilde{\varepsilon}_z \\
0 & 0 & 0 & 0 & -\alpha_z\tilde{\varepsilon}_z & \gamma_z\tilde{\varepsilon}_z
\end{pmatrix}
$$

3. SADScript Implementation (cont.)

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

- The transfer matrix Φ_n for an element n has the form

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

where L_n is the length of the element and $\mathbf{F}_n$ is the *generator matrix* which represents the external forces of element n.

- To sub-divide element n, we require the matrix $\mathbf{F}_n$, given by

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

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

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

3. SADScript Implementation (cont.)

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

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

3. SADScript Implementation (cont.)

Numerical Efficiency

- For a step size of Δs, rather than computing the "exact" transfer matrix

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

we compute a transfer matrix which is second-order accurate in Δs.

- Note that

$$\Phi_n(\Delta s) = \Phi_{sc}(\Delta s/2)\ \Phi_n(\Delta s)\ \Phi_{sc}(\Delta s/2) + O(\Delta s^3)$$

where

$$\Phi_n(\Delta s) = \exp(\Delta s\mathbf{F}_n) \qquad \text{(computed once per element)}$$

$$\Phi_{sc}(\Delta s/2) = \exp(\Delta s/2\mathbf{F}_{sc}) = \mathbf{I} + \Delta s/2\mathbf{F}_{sc} \quad \text{(by idempotency, } \mathbf{F}_{sc}{}^2 = \mathbf{0}\text{)}$$

We have reduced a matrix exponentiation at each step Δs to one matrix addition and two matrix multiplications

3. SADScript Implementation

SADScript Modules and Some Notable Functions

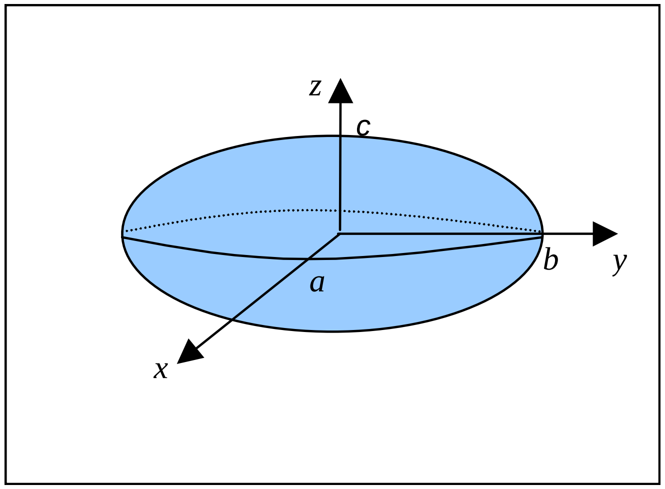
- oldsad/Packages/Scheff.n
  - $\{\{s_n\},\{\gamma_n\},\{\sigma_n\}\}$ = ScheffSimulate[K_0, σ_0, Δs:0.01]
  - $\{\{s_n\},\{\gamma_n\},\{\Phi_n\}\}$ = GetBeamLineElementData[]
  - SaveBeamMatrixData[*file*, $\{s_n\}$, $\{\gamma_n\}$, $\{\sigma_n\}$]

- oldsad/Packages/Trace3dToSad.n
  - K = ComputePerveance[f, E_r, W, Q]
  - $\{\alpha,\beta,\gamma\}_{\mathrm{SAD}}$ = TraceToSadTransTwiss[$\{\alpha,\beta,\gamma\}_{\mathrm{T3D}}$]
  - $\{\alpha,\beta,\gamma\}_{\mathrm{SAD}}$ = TraceToSadLongTwiss[f, E_r, W, $\{\alpha,\beta,\gamma\}_{\mathrm{T3D}}$]

- oldsad/Packages/TwissUtility.n
  - $\boldsymbol{\sigma}$ = CorrelationMatrix6D[$\{\alpha,\beta,\gamma\}_x$, $\{\alpha,\beta,\gamma\}_{\mathrm{y}}$, $\{\alpha,\beta,\gamma\}_{\mathrm{z}}$]

- oldsad/Packages/MatrixFunctions.n
  - $\mathbf{F}$ = MatrixLog[$\boldsymbol{\Phi}$]
  - $\boldsymbol{\Phi}$ = MatrixExp[$\mathbf{F}$]

4. Field Calculations



- Space charge effects are included by assuming the beam has **ellipsoidal symmetry** with dimensions corresponding to the statistics in $\boldsymbol{\sigma}$.

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

- Analytic field expressions for such a bunch distributions are available

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

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

4. Field Calculations (cont.)

Coordinate Transformations

- To apply the previous formula for ϕ we must rotate to the coordinates of the beam ellipsoid semi-axes using a transformation

$$\mathbf{R} \in SO(3) \subset SO(6).$$

- Moreover, we require a transformation $\mathbf{G} = \mathrm{diag}(1,1,1,1,\gamma,1/\gamma)$ to convert longitudinal coordinates from (z,dp) to (z,z') (momentum to primed)

- The complete transformation is

$$\Lambda = \mathbf{R}^T\mathbf{G}^T\sigma\mathbf{G}\mathbf{R}$$

where the $\langle xy\rangle$, $\langle xz\rangle$, $\langle yz\rangle$ elements of Λ are zero

It is important that this transform is numerically accurate!

4. Field Calculations (cont.)

Take the Linear Part of the Electric Fields

To each electric self-field component E_x, E_y, E_z is expanded in the form

$$E_x = a_1 x + a_2 y + a_3 z \qquad \text{(e.g., for } x \text{ plane)}$$

- Multiplying the above equation by the functions $\{x,y,z\}$ then taking moments

$$\begin{pmatrix} \langle x^2\rangle & \langle xy\rangle & \langle xz\rangle \\ \langle xy\rangle & \langle y^2\rangle & \langle yz\rangle \\ \langle xz\rangle & \langle yz\rangle & \langle z^2\rangle \end{pmatrix} \begin{pmatrix} a_1 \\ a_2 \\ a_3 \end{pmatrix} = \begin{pmatrix} \langle xE_x\rangle \\ \langle yE_x\rangle \\ \langle zE_x\rangle \end{pmatrix}$$

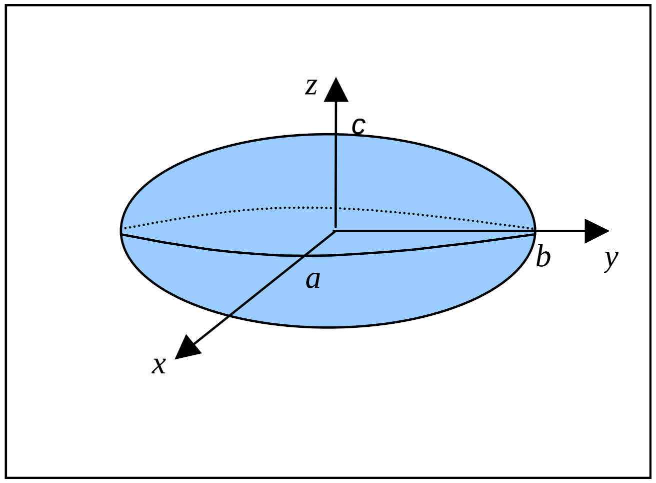
if $\langle xy\rangle = \langle xz\rangle = \langle yz\rangle$ (transform **GR**) we have $E_x(x,y,z) = \dfrac{\langle xE_x\rangle}{\langle x^2\rangle} x$

This is the *weighted, least-squares, linear approximation* for the self fields

4. Field Calculations (cont.)

Field Moments

For ellipsoidal beams having a density distribution *f*, the self-field moments can be calculated from ϕ and are given by the following:



$$\langle xE_x \rangle = \frac{\Lambda(f)}{\sqrt{3}} \frac{Q}{24\pi\varepsilon_0} < x^2 > R_D\left[< y^2 >, < z^2 >, < x^2 >\right] ,$$

$$\langle yE_y \rangle = \frac{\Lambda(f)}{\sqrt{3}} \frac{Q}{24\pi\varepsilon_0} < y^2 > R_D\left[< z^2 >, < x^2 >, < y^2 >\right] ,$$

$$\langle zE_z \rangle = \frac{\Lambda(f)}{\sqrt{3}} \frac{Q}{24\pi\varepsilon_0} < z^2 > R_D\left[< x^2 >, < y^2 >, < z^2 >\right] ,$$

where $\Gamma(f)$ is almost constant (F. Sacherer) and R_D is the elliptic integral

$$R_D(x, y, z) \equiv \frac{3}{2} \int_0^\infty \frac{dt}{(t+x)^{1/2}(t+y)^{1/2}(t+z)^{3/2}}$$

4. Field Calculations (cont.)

Space Charge Generator Matrix

Thus, the full space charge generator matrix $\mathbf{F}_{sc}(\sigma)$ is given by

$$\mathbf{F}_{sc}(\boldsymbol{\sigma}) = \mathbf{G}^{-1}\mathbf{R}\begin{pmatrix} 0 & 0 & 0 & 0 & 0 & 0 \\ 1/f_x & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 1/f_y & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 1/f_z & 0 \end{pmatrix}\mathbf{R}^T\mathbf{G}^{-1}$$

where

$$\frac{1}{f_x} = \frac{K}{2\cdot 5^{3/2}} R_D\left[<y^2>,<z^2>,<x^2>\right]$$

$$\frac{1}{f_y} = \frac{K}{2\cdot 5^{3/2}} R_D\left[<z^2>,<x^2>,<y^2>\right]$$

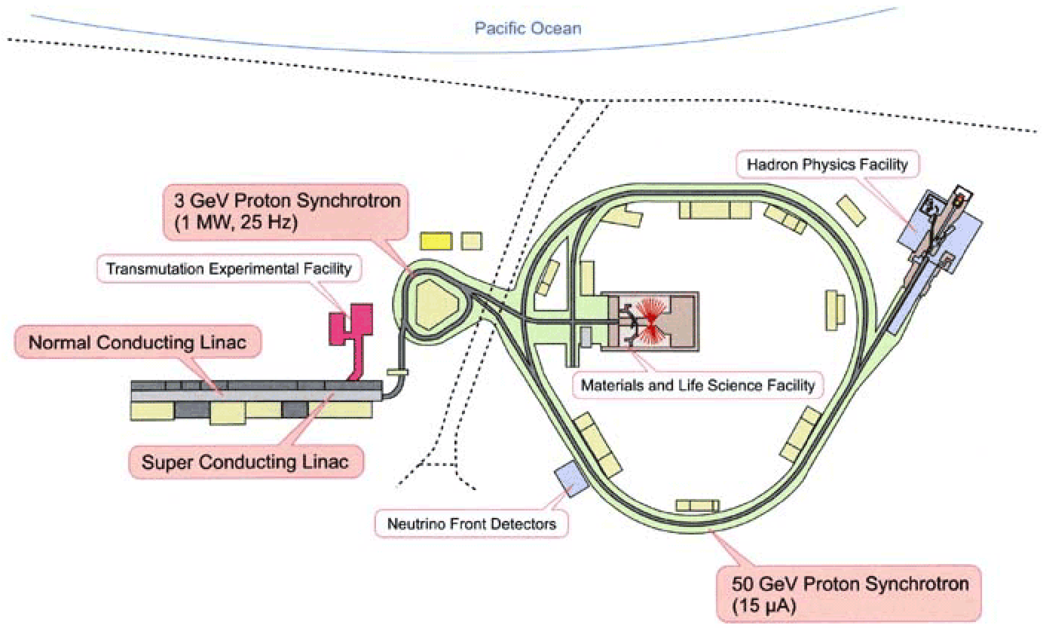
$$\frac{1}{f_z} = \frac{\gamma^2 K}{2\cdot 5^{3/2}} R_D\left[<x^2>,<y^2>,<z^2>\right]$$

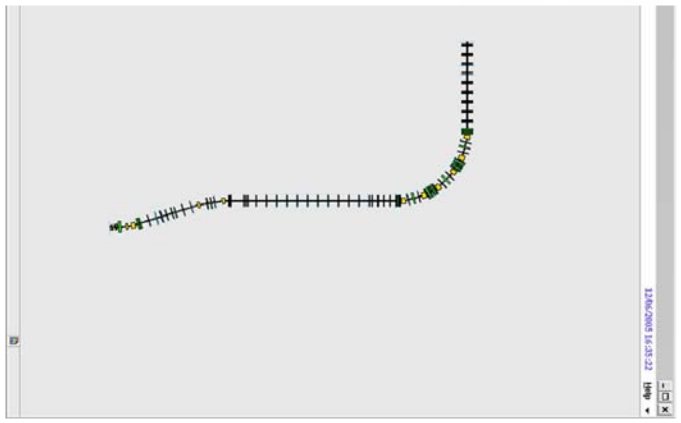
5. Simulation Results

J-PARC BT Line Simulation

- Show SADScript envelope simulation of the J-PARC BT line for several cases
  - Zero current
  - 30 mA
  - 130 mA

- Compare the SADScript envelope simulation to simulations provided by Trace3D
  - Notable differences
    - Trace3D does not impose symplectic condition
    - Trace3D can simulate emittance growth thru RF Gaps (removed)

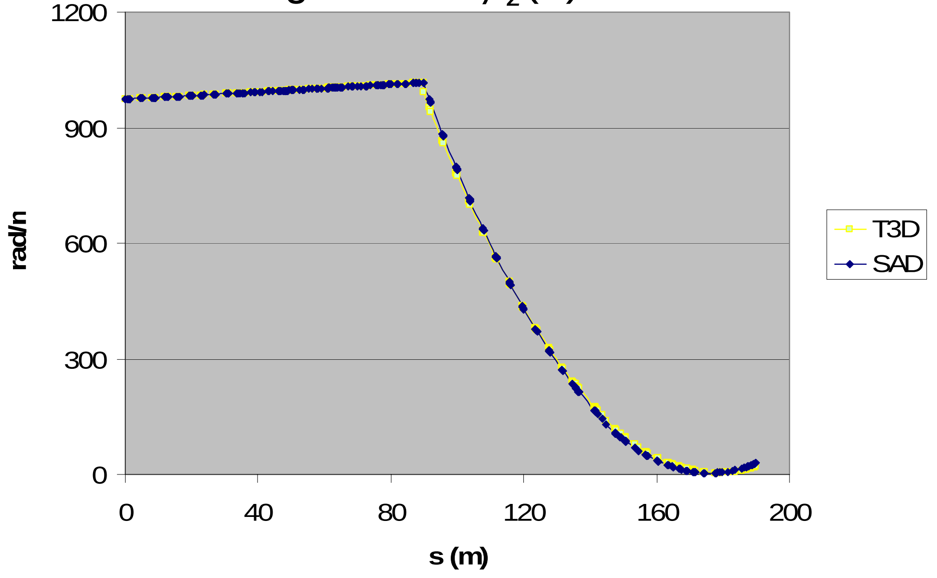

Pacific Ocean
Hadron Physics Facility
3 GeV Proton Synchrotron (1 MW, 25 Hz)
Transmutation Experimental Facility
Normal Conducting Linac
Materials and Life Science Facility
Super Conducting Linac
Neutrino Front Detectors
50 GeV Proton Synchrotron (15 μA)



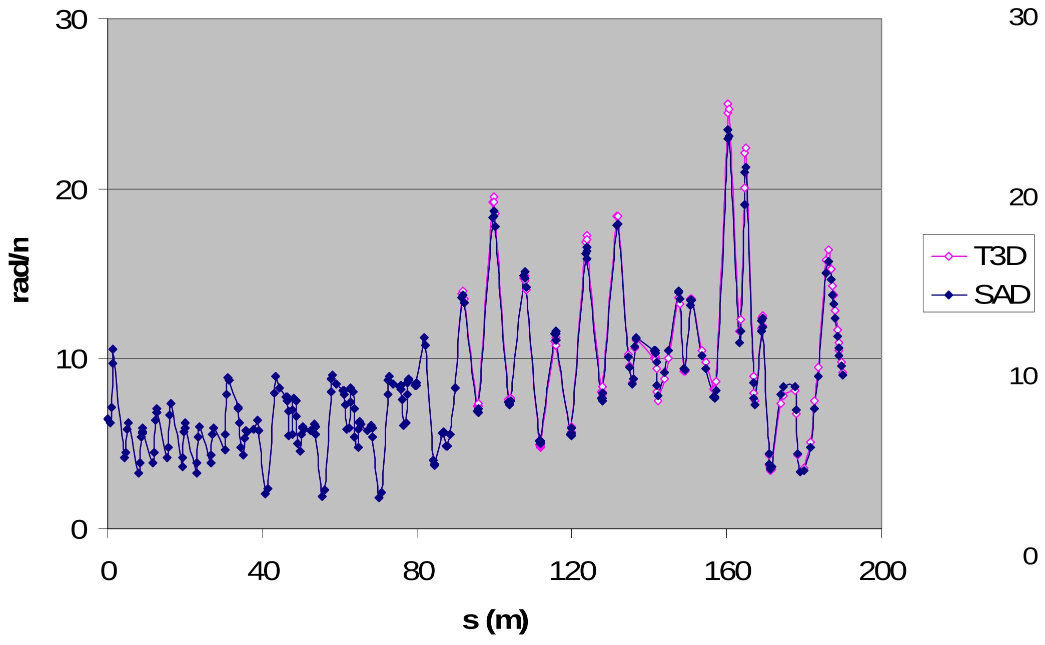
5. J-PARC Simulation

0 mA

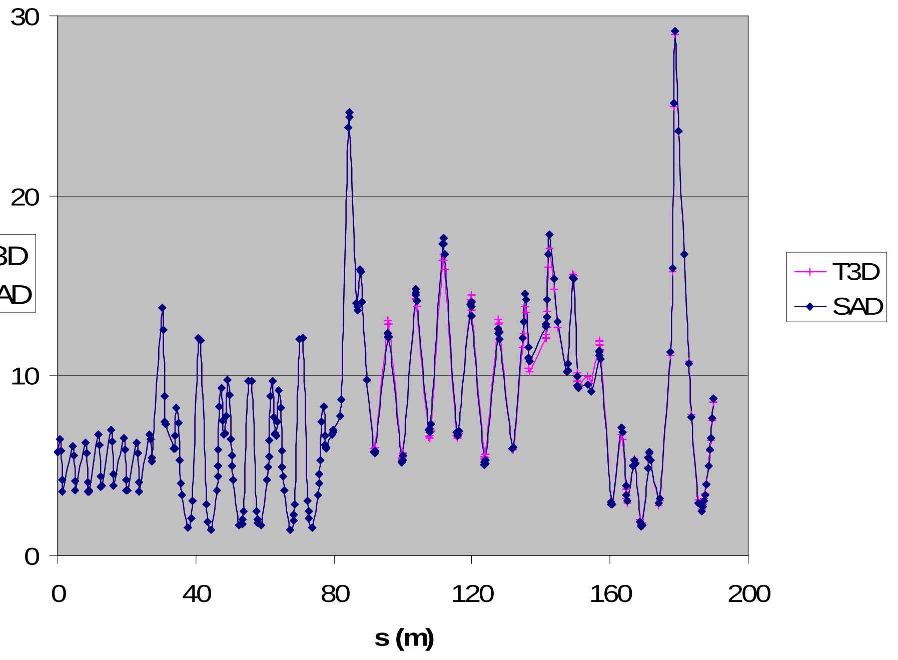
Longitudinal $\beta_z(s)$



Horizontal $\beta_x(s)$



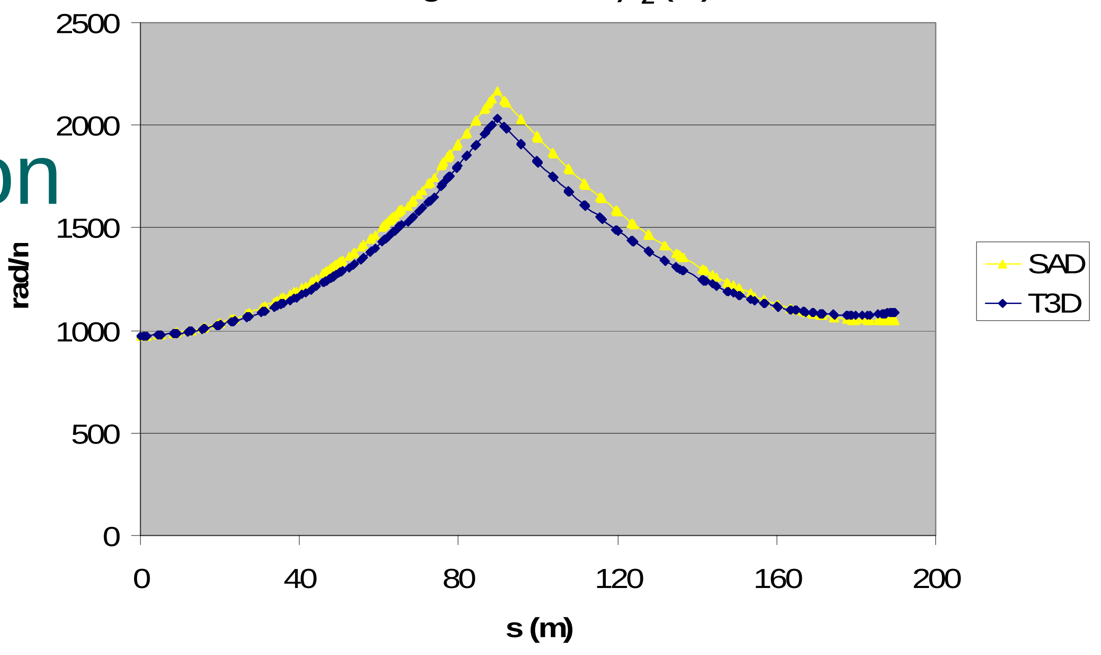
Vertical $\beta_y(s)$



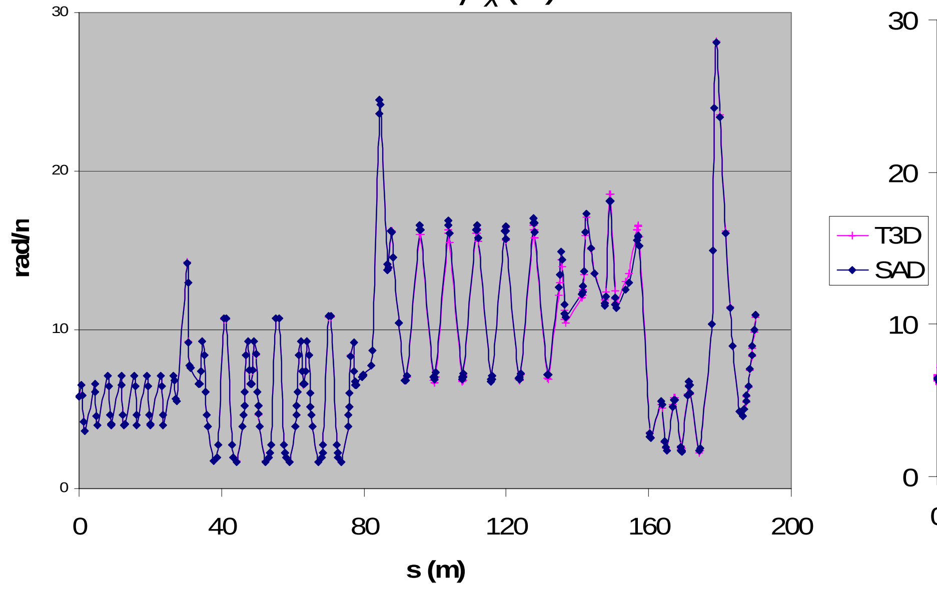
5. J-PARC Simulation

30 mA

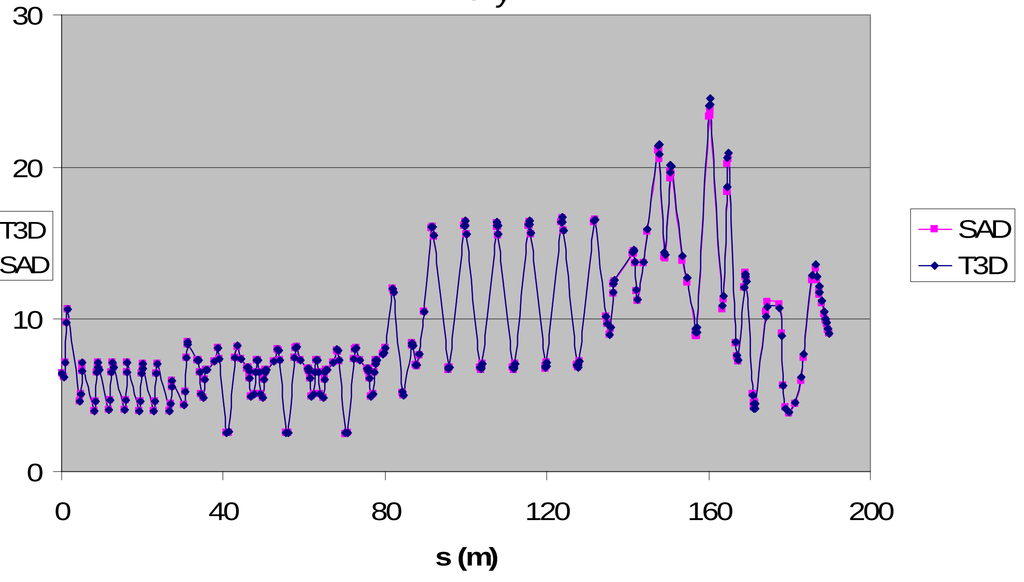
Longitudinal $\beta_z(s)$



Horizontal $\beta_x(s)$



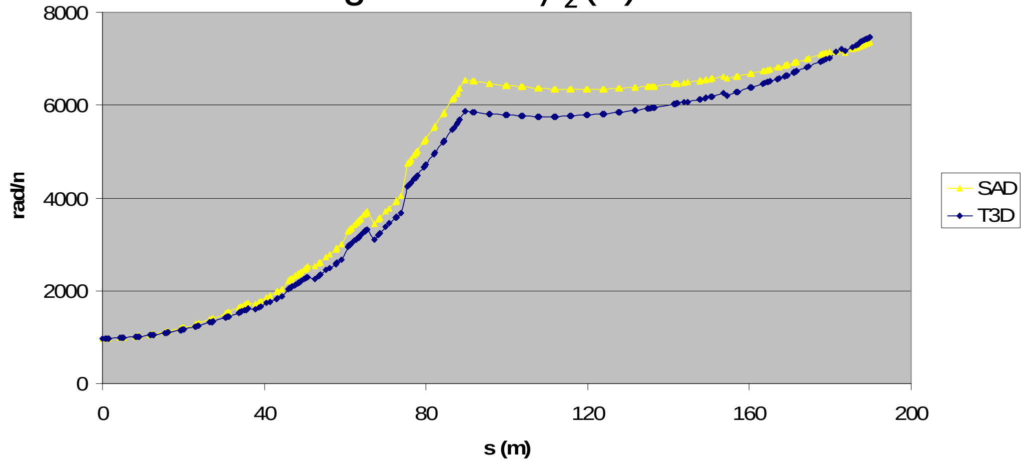
Vertical $\beta_y(s)$



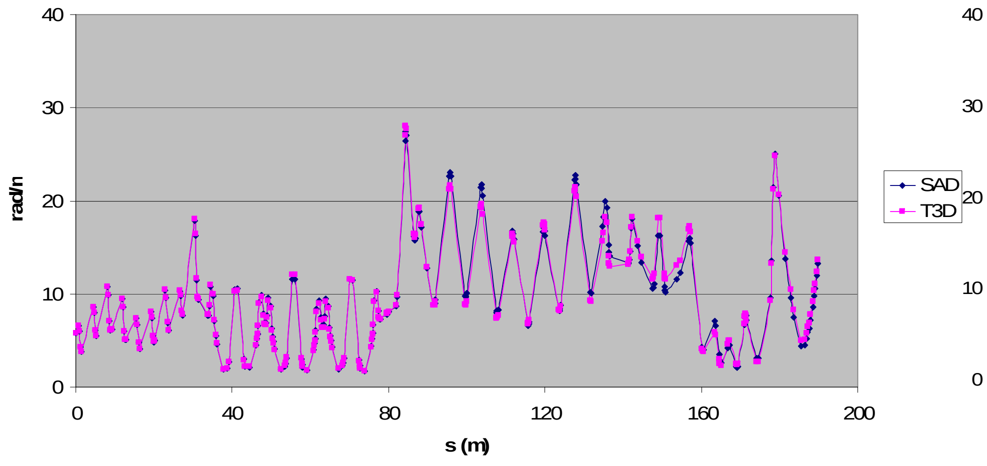
5. J-PARC Simulation

130 mA

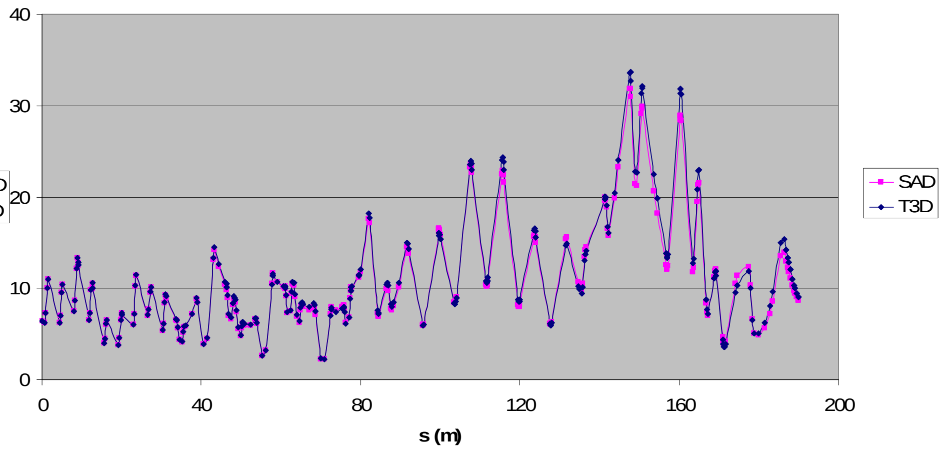
Longitudinal $\beta_z(s)$



Horizontal $\beta_x(s)$



Vertical $\beta_y(s)$



6. Issues

- Computation of the matrix logarithm log(Φ_n) is expensive.
  - Current procedure uses an iterative technique which computes a matrix exponential exp($\mathbf{F}_n$) at each step
  - The procedure works for Φ_n close to the identity matrix $\mathbf{I}$
    - It is not robust, but it should suffice for symplectic matrices

- Computation of the matrix exponential exp($\mathbf{F}_n$) is also non-trivial
  - Use a Taylor expansion with scaling and squaring.

6. Issues (cont.)

- Currently only a simple stepping procedure is employed.
  - The step size Δs remains constant throughout simulation
  - By implementing an adaptive stepping algorithm we can obtain significant speedup and maintain a specified level of accuracy in the solution (see “Bunched Beam Envelope Simulation with Space Charge”, KEK, Jan 20, 2005.)
- The longitudinal space charge force seems to be slightly stronger in all SAD simulations as compared to Trace3D.
  - I have debugged the SAD code extensively and have not found any errors in the theory or implementation.
  - I believe this condition is simply a result of the difference in simulation architecture, but I may be wrong.

6. Conclusions

- A major issue is computational speed
  For the J-PARC simulation
  - Trace3D runs on the order of 0.5 seconds
  - SAD ScheffSimulate runs on the order of 0.5 minutes

  By implementing the adaptive stepping and/or implementing the computationally expensive functions as compiled code we should see significant speedup.
  - Only the elliptic integral function $R_D(x,y,z)$ in implemented in compiled code.
  - Implementing MatrixExp() as compile code would be the most cost-effective

- The small difference in longitudinal dynamics between SAD and Trace3D may be an artifact of the different approaches. However, I am not sure.

3. SADScript Implementation

- Initialization
  - Load the beamline information from the target "input deck"
  - We acquire all the transfer matrices $\{\Phi_n\}$ and all the element lengths $\{L_n\}$

- Space Charge
  - Compute the partial transfer matrix $\Phi_n(\Delta s)$ where $\Delta s = L_n/N$
  - Compute space charge transfer matrix Φ_{sc} for a distance Δs.
  - Combine $\Phi_n\,(\Delta s)$ and Φ_{sc} for full transfer matrix $\Phi_{n,sc}(\Delta s)$

- Propagation
  - Propagate σ_n through **element** using $\sigma_{i+1} = \Phi_{n,sc}(\Delta s)\sigma_i\Phi_{n,sc}(\Delta s)^T$ recomputing $\Phi_{n,sc}(\Delta s)$ as necessary
  - Propagate $\{\sigma_n\}$ through beamline using above procedure for each element n.

3. SADScript Implementation (cont.)

Beam Dynamics with Space Charge

~~To include space charge effects we must determine the self forces then augment~~ the dynamics $\sigma_{n+1} = \Phi_n \sigma_n \Phi_n^T$ to include them.

To propagate the moment matrix σ_n through an element we must compute a transfer matrix $\Phi_{n,sc}$ that includes space charge.

- Such a transfer matrix can only be computed accurately for a short distance $\Delta s <$ Ln
- We must divide each beamline element into sub-elements of length Δs having the appropriate transfer matrix $\sqrt[N]{\Phi_n}$
- We apply the dynamics $\sigma_{n+1} = \sqrt[N]{\Phi_n} \sigma_n \sqrt[N]{\Phi_n}^T$ many times, recomputing

We use a “kick-like” approach - correcting the beam state at regular intervals through each element

- We divide each beamline element into sub-elements of length Δs
- We represent the self force through Δs as a transfer matrix Φ_{sc}
- Because the self forces depend upon the beam shape, the matrix self force transfer matrix Φ_{sc} depends upon the moment matrix σ