Modelling with Space-Charge

Chris Prior
UK Accelerator Science & Technology Centre
and
University of Oxford

Historical

• 1980’s - inertial confinement fusion, behaviour of beams under intense space charge
  – Lawson, Reiser, Keefe, Wangler..
• 1990’s - spallation neutron sources: EHF, JHF, ESS, SNS
  – Machida, Holmes ...
• 2000’s - proton drivers, neutrino factory, Fermilab upgrades
• 2010’s - ...

Monday, 15 April 2013
Historical

- 1980’s - inertial confinement fusion, behaviour of beams under intense space charge
  - Lawson, Reiser, Keefe, Wangler..
- 1990’s - spallation neutron sources: EHF, JHF, ESS, SNS
  - Machida, Holmes ...
- 2000’s - proton drivers, neutrino factory, Fermilab upgrades
- 2010’s - ...

Message: Don’t run codes blindly and assume the answers are always right. Question the results. It pays to know what is in the codes, what they calculate and how. In accelerator modelling, all-purpose “black boxes” don’t always work.
Overview

- Principles of simulation with space-charge
- Calculation of space-charge forces
- Analytical models
- Formulation of results
- Benchmarking and validation
- Use of codes
- Illustrations
- List of codes
Particle coordinates \((x, y, z)\) with respect to frame whose motion is given by \(s\)

Factor \(1/\gamma^2\) from electrostatic-magnetostatic effects

Other factor \(\gamma\) from relativistic mass \(m = m_0 \gamma\)

Space-charge field \(\mathbf{E}\) from Maxwell’s equation:

\[
\nabla \cdot \mathbf{E} = \frac{q}{\epsilon_0} n(x, y, z, s) \tag{4}
\]

where \(n(x, y, z, s)\) is the number density of the beam distribution.
The total number of particles in the beam is

$$N = \iiint n(x, y, z, s) \, dx \, dy \, dz. \quad (7)$$

This is a complete set of seven coupled equations in which the distribution determines the forces, which determine the motion, which determines the distribution, and so on.
Recall: For a 2D uniform beam with elliptical cross section \( \frac{x^2}{a^2} + \frac{y^2}{b^2} \leq 1 \), space-charge forces are linear and given by

\[
E = \frac{Nq}{\pi \varepsilon_0 (a + b)} \left( \frac{x}{a}, \frac{y}{b} \right),
\]

where \( N \) is the number of particles per unit length.

Equations of particle motion and envelope equations are then:

\[
\begin{align*}
x'' + k_x(s)x - \frac{2K}{a + b} \frac{x}{a} &= 0 \\
y'' + k_y(s)y - \frac{2K}{a + b} \frac{y}{b} &= 0 \\
a'' + k_x(s)a - \frac{\varepsilon_x^2}{a^3} - \frac{2K}{a + b} &= 0 \\
b'' + k_b(s)b - \frac{\varepsilon_y^2}{b^3} - \frac{2K}{a + b} &= 0
\end{align*}
\]

\[
K = \frac{I}{I_0} \frac{2}{(\beta \gamma)^3} \quad \text{is the Perveance} \quad \text{and} \quad I_0 = \frac{4\pi \varepsilon_0 m_0 c^3}{q}
\]
For non-linear beams, rms beam size is $\tilde{a} = \sqrt{\langle x^2 \rangle}$ and **rms evolution equations** are

$$\frac{d^2 \tilde{a}}{ds^2} + k_x(s) \tilde{a} - \frac{\tilde{\epsilon}^2}{\tilde{a}^3} - \frac{q}{m_0 \gamma^3 \beta^2 c^2} \frac{\langle x E_x \rangle}{\tilde{a}} = 0$$  \hspace{1cm} (1)

$$\frac{d^2 \tilde{\epsilon}^2}{ds^2} = \frac{2q}{m_0 \gamma^3 \beta^2 c^2} \left[ \langle x^2 \rangle \langle x' E_x \rangle - \langle xx' \rangle \langle x E_x \rangle \right]$$  \hspace{1cm} (2)

where $\tilde{\epsilon} = \sqrt{\langle x^2 \rangle \langle x'^2 \rangle - \langle xx' \rangle^2}$ is the rms emittance.  \hspace{1cm} (3)

Sacherer showed that for ellipsoidal particle densities of the form

$$n(x, y, z, s) = n \left( \frac{x^2}{a^2} + \frac{y^2}{b^2} + \frac{z^2}{c^2} \right)$$

the averages $\langle x E_x \rangle, \langle y E_y \rangle$ etc depend only very weakly on the exact charge distribution.

So the general rms envelope equation is the same as for an equivalent KV beam.
2D (transverse) codes use envelope equations for KV beam:

\[
a'' + k_x(s)a - \frac{\varepsilon_x^2}{a^3} - \frac{2K}{a + b} = 0
\]

\[
b'' + k_y(s)b - \frac{\varepsilon_y^2}{b^3} - \frac{2K}{a + b} = 0
\]

\[
K = \frac{I}{I_0} \frac{2}{(\beta \gamma)^3}
\]

is the Perveance

Define \[X = \begin{bmatrix} a \\ a' \\ b \\ b' \end{bmatrix}\]

\[
\frac{dX}{ds} = \frac{d}{ds} \begin{bmatrix} a \\ a' \\ b \\ b' \end{bmatrix} = \begin{bmatrix} a' \\ -k_x(s)a + \frac{2K}{a + b} + \frac{\varepsilon_x^2}{a^3} \\ b' \\ -k_y(s)b + \frac{2K}{a + b} + \frac{\varepsilon_y^2}{b^3} \end{bmatrix}
\]

Integrate using standard numerical packages based on, for example, Runge-Kutta techniques.

Output either as beam sizes or Twiss parameters defined by \[\beta_x = \frac{a^2}{\varepsilon_x},\]

\[\alpha_x = -aa'/\varepsilon_x\text{ etc}\]
**Examples of Envelope Codes**

**KVBL** (Prior) Includes matching, parameter optimisation, graphical output; switches to standard matrix methods in absence of space-charge

**WinAgile** (Bryant) Similar to KVBL but Windows based with GUI

**Trace2D** Old code from Los Alamos, based on matrices with space charge kicks in the middle of elements

**Trace3D** Development of Trace2d to include momentum effects; works from a keyboard implemented graphics system.

\[ \beta \text{-functions with space-charge from the ESS} \]
KVBL: ESS Funnel (1996)

- 57 mA linac current (H⁻)
- Achromatic in all planes
- Uniform optics to minimise non-linear space-charge effects

Enabled emittance growth to be reduced to <1% cf ~50% in early LANL experiments
Basic Tracking Procedures

1. Description of machine
2. Basic beam parameters
3. Input distribution, N macro-particles
4. Calculate external forces
5. Calculate space-charge forces
6. Push particles forward one step
7. Output as required

- e.g. MAD format, to include field maps
- graphical facilities

Monday, 15 April 2013
Input Beam Distribution

• How many macro-particles are needed to model a beam of $10^{10}$-$10^{14}$ real particles?
  – sufficiently many for good statistics
  – predicted effects should not be a consequence of reduced number of particles, statistical errors, rounding and interpolation errors.

• Most space-charge codes now use $\sim 10^5$-$10^6$ simulation particles; some runs have been made with $10^7$-$10^8$. 
Can be read from a given dataset, for example from a previous run, or can be based on physical data.

Can be a model distribution - KV, Waterbag, Gaussian, semi-Gaussian etc

Also include stationary distributions (self-consistent functions of the Hamiltonian $H$)

- generate a normalised distribution; then scale and rotate as appropriate
- may need to change coordinates, e.g Cartesians to 4D-polar system
- can be fitted to given beam sizes or created as an rms equivalent beam
Method based on $f(x) \, dx = dF(x)$ where $F'(x) = f(x)$; this may not always be possible.

**Method of Ratio of Uniform Deviates:**

The density function $f(x)$ can be generated through a uniform filling of the region

$$0 < u < \sqrt{f \left( \frac{v}{u} \right)}$$

of the two-dimensional $(u, v)$ plane with a random number generator. Then $x = \frac{v}{u}$ has the desired density function.

**Example: The Cauchy Distribution**

$f(x) = \frac{1}{\pi} \frac{1}{1 + x^2}$.

The sampling region is

$$\left\{ (u, v) : 0 \leq u \leq \left[ 1 + \left( \frac{v}{u} \right)^2 \right]^{-\frac{1}{2}} \right\} = \{(u, v) : u^2 + v^2 \leq 1, u \geq 0 \}.$$

Half circle, centred on origin, radius 1.
Example: Gaussian Distribution \[ f(x) = \frac{1}{\sqrt{2\pi}} e^{-\frac{1}{2}x^2}. \]

The sampling region is

\[ \{(u, v) : 0 \leq u \leq e^{-\frac{v^2}{2u^2}}\} = \{(u, v) : v \leq \sqrt{-2u^2 \ln u}, \, 0 \leq u \leq 1\}. \]

**Sampling method:** adjust tangents at \(P, Q\) to minimise area \(ABCD\).

\[\implies\] Probability that a point within \(ABCD\) is also within required region is 0.922. Method uses some simple initial pre-sampling to check that a point \((u, v)\) lies in region; if so, then a more accurate check.

Approach turns out to be faster than most other methods and also gives a very good model of the required distribution.
Solution of Equations of Motion

Equations of motion of the form \( x'' = F(s, x, x') \)

Euler (forward) difference method:

\[
\begin{bmatrix}
x_{n+1} \\
x'_{n+1}
\end{bmatrix} = \begin{bmatrix}
x_n + hx'_n \\
x'_n + hF(s_n, x_n, x'_n)
\end{bmatrix}
\]

\( h \) is step length

Accuracy is only \( O(h^2) \)

Could use Runge-Kutta method accurate to \( O(h^4) \), but requires extra storage and 4 calculations of \( F \) for each particle instead of one. Can be reduced by Blum’s method, but unlikely to give a viable method for modelling \( \gtrsim 10^5 \) particles in a realistic time.

Set \( X = \begin{bmatrix} x \\ x' \end{bmatrix} \), \( \frac{dX}{ds} = f(s, X) = \begin{bmatrix} x' \\ F(s, x, x') \end{bmatrix} \)

\[
\begin{align*}
k_1 &= hf(s_n, X_n) \\
k_2 &= hf(s_n + \frac{1}{2}h, X_n + \frac{1}{2}k_1) \\
k_3 &= hf(s_n + \frac{1}{2}h, X_n + \frac{1}{2}k_2) \\
k_4 &= hf(s_n + h, X_n + k_3)
\end{align*}
\]

\[
\begin{align*}
\implies X_{n+1} &= X_n + \frac{1}{6}(k_1 + 2k_2 + 2k_3 + k_4)
\end{align*}
\]
Symplectic Leap-frog Integration Scheme

Interleave position and velocity half a step out of phase, and leap-frog coordinates forward in distance or time.

\[
x_{n+1} = x_n + h x'_{n+\frac{1}{2}}
\]

\[
x'_{n+\frac{1}{2}} = x'_{n-\frac{1}{2}} + \frac{1}{2} h \left[ F(s_n, x_n, x'_{n-\frac{1}{2}}) + F(s_n, x_n, x'_{n+\frac{1}{2}}) \right]
\]

Error is \( O(h^3) \)

Step length must be chosen to allow plasma oscillations to be represented \( (\omega p h / \beta c \ll 2) \)

Method is extremely stable. It has a time-reversible property, which avoids long-term drift caused by systematic errors that could mask the true solution. Note: even 4th order Runge-Kutta suffers.
Split Operator Approach

\[ H = H_{\text{ext}} \]

Magnetic Optics

\[ \mathcal{M} = \mathcal{M}_{\text{ext}} \]

Split Operator Methods

\[ H = H_{\text{ext}} + H_{\text{sc}} \]

Multi-Particle Simulation

\[ \mathcal{M} = \mathcal{M}_{\text{sc}} \]

\[ \mathcal{M}(t) = \mathcal{M}_{\text{ext}}(t/2) \mathcal{M}_{\text{sc}}(t) \mathcal{M}_{\text{ext}}(t/2) + O(t^3) \]

Philosophy:
- Do not take tiny steps to push \( \sim 10^7 - 10^8 \) particles
- Do take tiny steps to compute maps, then push particles with maps.

Acknowledgement: Rob Ryne/Ji Qiang – IMPACT
Many different approaches, most based on approximations.

- Use Coulomb forces between pairs of particles to calculate forces at each step.

- Assume variation of space-charge with distance is small; impose space-charge kicks once or twice per element, but track using non-space-charge methods otherwise.

- Use KV linear space-charge formula, scaled by longitudinal line density.

- Calculate space-charge potential from Poisson’s equation in beam-frame, either in 3D or 2+1D.

Note: some methods ignore boundary effects
• Coulomb approach is $O(N^2)$ and very time consuming. Problems when particles move too close together $\implies$ cut-off distance needed (Debye length). Also open to rounding errors, especially on the axes where transverse forces should sum to zero. e.g. 2D circular uniform beam, $N = 50,000$ simulation particles: calculations take several minutes and only 60% within 10% error band.

• 2D+1D approach gives good results (beam split into 2D slices for transverse forces, then longitudinal force from line density).

• Finite difference method represents sophisticated approach but inclusion of boundaries (image effects) not easy.

• Finite elements provide most flexible approach and can call on huge range of engineering expertise.
Example: SIMPSONS

- Solves Poisson’s equation in cylindrical coordinates for perfectly conducting circular pipe:

\[
\frac{1}{r} \frac{\partial}{\partial r} \left( r \frac{\partial \Phi}{\partial r} \right) + \frac{1}{r^2} \frac{\partial^2 \Phi}{\partial \phi^2} = -\frac{\rho}{\varepsilon_0 \gamma^2}
\]

- Charge distribution is Fourier transformed in azimuthal direction

\[
\Phi = \sum_m \phi_m e^{im\phi} \quad \leftrightarrow \quad \begin{cases} 
\phi_m = \frac{1}{2\pi} \int_0^{2\pi} \Phi(r, z, \phi) e^{-im\phi} \, d\phi \\
n_m = \frac{1}{2\pi} \int_0^{2\pi} \frac{\rho(r, z, \phi)}{\varepsilon_0 \gamma^2} e^{-im\phi} \, d\phi
\end{cases}
\]

- Then

\[
n_m(r, z) = -\frac{1}{r} \frac{\partial}{\partial r} \left( r \frac{\partial \phi_m(r, z)}{\partial r} \right) + \frac{m^2}{r^2} \phi_m(r, z)
\]
Solution with $\phi_m = 0$ on $r = b$ is

$$\phi_m = W(r) - \left(\frac{r}{b}\right)^m W(b), \quad \phi_{-m} = \phi_m^*, \ m \geq 0$$

where

$$W(r, z) = \begin{cases} 
\int_0^r \ln\left(\frac{r}{r'}\right) n_m(r', z) r' \, dr' & \text{for } m = 0 \\
\frac{r^m}{2m} \int_0^r n_m(r', z) r'^{1-m} \, dr' - \frac{r^{-m}}{2m} \int_0^r n_m(r', z) r'^{1+m} \, dr' & \text{for } m \neq 0
\end{cases}$$

In practice, integral is replaced by summation over grid points.

Then $\vec{E} = \nabla \Phi = \sum_m \left( \frac{\partial \phi_m}{\partial r}, \frac{im}{r} \phi_m, \frac{\partial \phi_m}{\partial z} \right) e^{im\phi}$
Finite Element Approach

Based on the variational problem:

\[
\delta \pi(\phi) = 0, \text{ where } \pi(\phi) = \frac{1}{2} \int_V |\nabla \phi|^2 \, dV + \frac{1}{\epsilon_0} \int_V \rho \phi \, dV - \int_{\partial V} \bar{\phi}_n \phi \, dS
\]

Equivalent to \( \nabla^2 \phi = -\frac{\rho}{\epsilon_0} \) in \( V \), \( \phi = \bar{\phi} \), \( \frac{\partial \phi}{\partial n} = \bar{\phi}_n \) on \( \partial V \)

Cover region with mesh to fit boundaries:

Fit a polynomial to each mesh:

\[
\phi = a_0 + a_1 x + a_2 y + a_3 z + a_4 x^2 + a_5 y^2 + a_6 z^2 + a_7 xy + a_8 xz + a_9 yz + \ldots = \sum_i f_i(\xi_j) \phi_i
\]
\( f_i \) are shape functions, \( \xi_i \) are areal coordinates. Look for a complete set to chosen order. For example to order 2, 10 unknowns \((a_0, \ldots, a_9)\), so need 10 nodes:

Variational problem turns into sparse matrix equation for potential at nodes given by

\[
K_{ij} \phi_j = Q_i, \tag{1}
\]

where \( Q_i \) come from charge distribution

\[
\rho(x) = \sum_{\text{particles } i} q_i \delta(x - x_i) \quad \Rightarrow \quad \int \rho \phi \, dV = \sum_{\text{particles } i} q_i \phi(x_i).
\]

\((K_{ij})\) depends only on mesh. (1) is solved by standard methods (Gaussian elimination, triple factoring, conjugate gradient etc).
Space-charge forces calculated from

\[ \mathbf{F} = -\nabla \phi = - \sum_{i,j} \frac{\partial f_i}{\partial \xi_j} \phi_i \nabla \xi_j. \]

- Since *stiffness matrix* \( (K_{ij}) \) depends only on mesh, can be set up and pre-inverted, giving a fast, simple ”black-box” for space-charge calculations

\[ \phi_i = \sum_j K_{ij}^{-1} \rho_j \]

**Standard Test:** 2D uniform circular beam, \( N = 50,000 \) macro-particles

- 3rd Order, \( \sim 400 \) mesh elements, find \( > 95\% \) within 10\% error band
- 1st order, \( \sim 3000 \) mesh elements, find \( > 90\% \) within 10\% error band
• Simple conversion to different coordinate systems.
  
  – For example, 2D transverse \((x, y)\) Poisson solver to axisymmetric 3D \((r, z)\) code through \(x \rightarrow r, y \rightarrow z, \ dx\ dy \rightarrow r\ dr\ dz\)

• 2D code easily converted to 3D, triangular elements to tetrahedra

• Fits all boundaries likely in accelerators. For analytical purposes, “infinite” boundaries modelled using super-elements or matching to \(\ln r\) (2D) or \(1/r\) (3D) potentials at sufficiently large distances.

• Longitudinal boundary conditions generally periodic (bunch to bunch in linacs or rings)

• May require large amount of storage (not a problem nowadays) and CPU
  
  – can be parallelised using method of static condensation to split into substructures
Example: Multiturn Injection at FNAL

Injection into a 0.4-8 GeV RCS designed as replacement for Fermilab booster

- 400 MeV H⁻ injection
- 45 injection turns, 474 m ring
- Phase space painting
  - Horizontal orbit bumps
  - Vertical variation of beam angle

Basic injection parameters and painting optimisation using codes MISxxx (C.Prior)
Monday, 15 April 2013
Monday, 15 April 2013
Multiturn Injection modelled with TRACK2D

- 20 turns of 400 mA Bi$^{+1}$ beam.
- Space-charge tune depression $\sim 0.04$.
- Two-plane injection using tilted electrostatic septum
- Note distortion of individual turns in phase space
Study from ~1998 of 3 turn stacking process in PS-Booster using TRACK2D

Shows beam loss at septum and development of a 4th order space-charge resonance

C. Prior & P. Knaus
Special Case: 1D Longitudinal Codes

\[
\frac{d\Delta \phi}{dt} = \frac{\hbar \omega_0^2 \eta}{\beta^2 \mathcal{E}} \left( \frac{\Delta \mathcal{E}}{\omega_0} \right), \quad \frac{d}{dt} \left( \frac{\Delta \mathcal{E}}{\omega_0} \right) = \frac{q}{2\pi} \left[ V(\phi) - V(\phi_s) + U_s(\phi) \right]
\]

\[
U_s = -q \beta c R \left[ \frac{g_0}{2\beta} \frac{Z_0}{\gamma^2} - \omega_0 L \right] \frac{\partial \lambda}{\partial s}
\]

Tracking uses symplectic mapping with space charge calculated from derivative of the line density.

**CODES:** ESME (FNAL), LONG1D (TRIUMF), TRACK1D (RAL)

A major issue in a high intensity proton accelerator is building up the beam intensity through several turns of injection. For the SNS, \( N_{\text{turns}} = 1600 \) turns are required; for the ESS, \( N_{\text{turns}} \sim 1000 \). For reliable results, need \( \sim 5000 \) particles per turn, so \( \gtrsim 5 \times 10^6 \) overall.

One solution is to use “painting” technique with variable charge build-up. Restricts total to \( \sim 10^5 \) particles, yet has \( \sim 5000 \) per turn.
TRACK1D assignment uses Triangular Shaped Cloud (TSC). Line density smoothed with cubic splines to remove statistical effects. Includes corrections to counteract artificial spreading of the beam.

Figure 5-5 The assignment function shape interpretation of charge assignment. The fraction of charge assigned from a particle at position $x$ to a given mesh point is equal to the value of the assignment function $W$ at that point.
Longitudinal Study of CERN-PS

C. Prior & P. Knaus, ~1998

- Off-energy injection
- 11 micro-bunches per bucket
- Tracked for ~10 synchrotron periods
- No beam loss, good distribution
Dual harmonic (h=2/4) injection, trapping and acceleration in ISIS. 150 turns of 70-800 MeV beam

Longitudinal phase space plots for acceleration of $3 \times 10^{13}$ protons in the ISIS synchrotron from 70 MeV to 800 MeV with dual harmonic RF system.

C. Prior, ICANS 1996
Available Codes I

IMPACT  Rob Ryne, Ji Qiang (LBL); mainly a linac code with new MaryLie developments for modelling rings

TRACEWIN  Nicolas Pichoff (CEA); Windows linac code

ORBIT  Jeff Holme, Sarah Cousineau (SNS); ring code developed for SNS; exists in different versions at ORNL, FNAL, BNL

SIMPSONS  Shinji Machida (RAL); used for J-PARC modelling

ACCSIM  Fred Jones (TRIUMF); developed from matrix applications with space-charge kicks

TRACKxD  Chris Prior (RAL); $x = 1, 2, 3$; used for ISIS, ESS, SNS, HIDIF, Neutrino Factory and other modelling

GPT  Bas van der Geer (Pulsar Physics); Commercial code developed initially for high intensity, very short electron bunches.
OPAL  Andreas Adelmann (PSI); major advances; fully 3D; developed mainly for cyclotron studies

BEST  Hong Qin (PPPL); based on $\delta f$-method. Used for investigating two-stream instabilities.

Micromap  Ingo Hofmann, Giuliano Franchetti (GSI); used for modelling FAIR and for comparison with earlier theories.

WARP  David Grote, Alex Friedmann (LLNL); fully 3D, mainly for early stages of acceleration, from ion gun.

PATH  Alessandra Lombardi (CERN); originally written to model the muon beam for the CERN neutrino factory, later developed for protons.

VADOR  Eric Sonnendrucker (Strasbourg); Vlasov solver

PARMILA  Jim Billen (LANL); long established linac code; works in association with Trace3D; also PARMELA.
Goals

Plus:

- Full treatment of field maps
- Off-axis beams
- Full suite of routines for beam diagnostics
- Higher order effects
- Secondary particle effects
  - electron cloud
  - decays (muons)

From: ACCSIM, Fred Jones (TRIUMF)
Components of a typical Beam Modelling Package

- Initialize particles
- Advance positions & momenta a half step using $H_{\text{ext}}$
- Setup and solve Poisson equation
- Charge deposition on grid
- Field solution on grid
- Field interpolation at particle positions
- Advance positions & momenta a half step using $H_{\text{ext}}$
- Advance momenta using $H_{\text{space charge}}$
- (optional) diagnostics

Courtesy of R. Ryne

Monday, 15 April 2013
Components of a typical Beam Modelling Package

- **Initialize particles**
- **Advance positions & momenta a half step using** $H_{ext}$
- **Setup and solve Poisson equation**
  - **Charge deposition on grid**
  - **Field solution on grid**
  - **Field interpolation at particle positions**
  - **Advance momenta using** $H_{space\ charge}$
- **Advance positions & momenta a half step using** $H_{ext}$
- **(optional) diagnostics**

**Synergia**
- **wrapper**
- **IMPACT**
- **mxzyptlk/ beamline**

**Science & Technology Facilities Council**

Monday, 15 April 2013
2.3 Enabling technologies

Software infrastructure for multiphysics modeling is a central part of ComPASS. Our approach is to develop application components using existing mature physics or algorithmic implementations as the core of each component. Here, "component" is defined as a portion of software implementation that can be added or removed from multiple applications. The most serious challenge in developing simulation components is the definition of their interfaces: the components need to be objects that can be used for multiple implementations of different ComPASS simulations. Thus, the interface definition has to be independent of any details from any particular application implementation, including the parallelization (data distribution) schemes. Our approach builds on AST and ComPASS work on the Synergia2 [13] and MaryLie/IMPACT [14] beam dynamics frameworks and is extending to other areas of the project with work such as the TEMP3P electromagnetic simulation framework. Figure 4 shows a schematic of the Synergia2 framework, with the physics components and software infrastructure dependencies.

Use of advanced mathematical techniques, scalable numerical algorithms, and computational tools are also major components of the ComPASS activities. For example, through implementation in VORPAL, SciDAC has supplied the first massively parallel implementation of FDTD electromagnetic computations (see figure 3). While many of the mathematical and computational tools we employ are relatively mature, we need to enhance their capabilities to meet the petascale computational challenges of SciDAC-2. In addition, we need to explore the benefits of employing new techniques and algorithms, and we need to port the new and old implementations to the new petascale capable hardware that is or will be available in the SciDAC-2 era.
How accurate are the codes?

- All codes should have a set of basic tests, preferably with known analytical solutions

- Benchmarking should cover
  - code v. code
  - code v. experiment

- Recent examples
  - Montague resonance tests with CERN PS, \(2Q_h-2Q_v=0\) (ACCSIM, SYNERGIA, MICROMAP, SIMPSONS, IMPACT, ORBIT, SIMBAD)
  - HIPPI linac injector comparison
  - Electron cloud studies (PEHT, PEHTS, QUICKPIC, HEADTAIL)
  - Study of Hofmann resonances at KEK (IMPACT, TraceWin)

See ICFA Beam Dynamics Newsletter 41, December 2006
ORBIT Application: SNS 1.44 MW Injection Space Charge Benchmark and Final Distribution

Benchmark of Space Charge Models

2D Space Charge, Direct Force Model
2D Space Charge, Potential Model
3D Space Charge

Percent of Beam Exceeding Emittance

Horizontal Emittance [mm-mrad]

Transverse Beam Distribution at 1.5 MW

Vertical Coordinate y [mm]

Horizontal Coordinate x [mm]

Science & Technology Facilities Council
10^5 macro particles
peak current 100A
corresponding to 2 MW proton accumulation

SIMBAD/ORBIT (FFT) VS. SIMPSONS
Gaussian distribution, original SNS FODO lattice

Beam Kinetic Energy | 1 GeV
Beam Average Power | 1.0-2.0 MW
Proton Revolution Period | 0.8413 μsec
Ring Circumference | 220.688 m
Number of Turns Injected | 1225
Beam Emittance $\varepsilon_{x,y}$ | 120 πmm-mr
Tunes $v_x / v_y$ | 1 1/290
Max. $\beta_x / max. \beta_y$ | 19.2 / 19.2 m
Dispersion $X_p$ (max/min) | 4.1 / 0.0 m
Scope of Existing Codes with Space Charge

• Results/predictions
  – Beam profile measurements: CERN-PSB, KEK-PS, PSR
  – Injection losses (e.g. mismatch, emittance transfer)
  – Coherent resonances -- intensity limitation
  – Benchmarks with Accsim, Orbit, Simpsons, show long-term (50k turns) RMS matching to high degree of precision

• Study
  – Beam redistribution while preserving RMS matching
  – Intrinsic resonance due to space charge (sensitive to working point, observed independently in most codes)
  – Synchro-betatron effects -- space charge, chromaticity
  – Halo parameters and other amplitude measures
  – Stationary distributions (not widely used?)
  – Still a need for more accessible benchmarks and test cases
Vlasov Solvers

- High intensity beams are usually modeled by the Vlasov equation.
- The distribution function $f(x, v, t)$ is given by
  \[
  \frac{\partial f}{\partial t} + v \cdot \nabla_x f + \frac{q}{m} (E + v \times B) \cdot \nabla_v f = 0,
  \]
generally coupled with the Poisson or Maxwell equations.
- Numerical simulations are mostly performed using PIC method.

Difficult in >2D phase space because of very large number of grid points. Hence very slow.

Use symmetry or conserved quantities on characteristics

Optimise number of grid points for specific simulations

- Important noise in PIC methods especially in poorly populated regions of phase space makes it hard to see phenomena like e.g.
  - particle trapping (strong Landau damping) in plasmas
  - halo formation in beams
- Computers now powerful enough to do realistic physics using a grid in phase space.
- Provides alternative to PIC for code benchmarking.

Proposal to develop 4D-->6D Vlasov solver with Jonathan Smith (Tech-X) and Hartree Centre, Daresbury

Monday, 15 April 2013
Example: Evolution of a Semi–Gaussian beam of 80keV Potassium ions in a uniform focussing channel, Q~-0.25

Results from VADOR code, E. Sonnendrucker, Strasbourg
<table>
<thead>
<tr>
<th>Code</th>
<th>Language</th>
<th>Platform</th>
<th>GUI</th>
<th>Parallel</th>
<th>1D/2D/3D</th>
<th>Particles</th>
<th>linacs/rings</th>
</tr>
</thead>
<tbody>
<tr>
<td>IMPACT</td>
<td>F90</td>
<td>Unix/Linux</td>
<td>no</td>
<td>MPI</td>
<td>3D</td>
<td>&gt; 10^6</td>
<td>linacs</td>
</tr>
<tr>
<td>ML-IMPACT</td>
<td>F90</td>
<td>Unix/Linux/Mac</td>
<td>no</td>
<td>MPI</td>
<td>3D</td>
<td>&gt; 10^6</td>
<td>linacs/rings</td>
</tr>
<tr>
<td>PARMILA</td>
<td>F90</td>
<td>Windows</td>
<td>no</td>
<td>no</td>
<td>2D/3D</td>
<td>10^4-10^5</td>
<td>linacs/transfer lines</td>
</tr>
<tr>
<td>GPT</td>
<td>C, C++</td>
<td>Windows</td>
<td>yes</td>
<td>MPI scans</td>
<td>3D</td>
<td>10^6</td>
<td>linacs/transfer lines</td>
</tr>
<tr>
<td>BEST</td>
<td>F90</td>
<td>Unix/Linux</td>
<td>python/IDL</td>
<td>MPI/OpenMP</td>
<td>3D</td>
<td>&gt; 10^6</td>
<td>linacs/rings</td>
</tr>
<tr>
<td>VADOR</td>
<td>C++</td>
<td>Unix/Linux</td>
<td>no</td>
<td>MPI</td>
<td>2D</td>
<td>n/a</td>
<td>linacs</td>
</tr>
<tr>
<td>SPUNCH</td>
<td>F77</td>
<td>Linux</td>
<td>no</td>
<td></td>
<td>1D</td>
<td>10^4</td>
<td>LEBT</td>
</tr>
<tr>
<td>PATH</td>
<td>F90</td>
<td>Windows</td>
<td>yes</td>
<td>no</td>
<td>3D</td>
<td>10^5</td>
<td>linacs/rings</td>
</tr>
<tr>
<td>TRACEWIN</td>
<td>C++</td>
<td>Windows</td>
<td>yes</td>
<td>no</td>
<td>2D/3D</td>
<td>10^5</td>
<td>linacs</td>
</tr>
<tr>
<td>DYNAC</td>
<td>F77</td>
<td>Linux/Unix/Windows</td>
<td>no</td>
<td>no</td>
<td>2D/3D</td>
<td>10^5</td>
<td>linacs</td>
</tr>
<tr>
<td>Synergia</td>
<td>F90/C++/Python</td>
<td>Unix</td>
<td>no</td>
<td>MPI</td>
<td>3D</td>
<td>&gt; 10^6</td>
<td>linacs/rings</td>
</tr>
<tr>
<td>WARP</td>
<td>Python/F77/F90/C</td>
<td>Linux/Unix/Windows/Mac</td>
<td>Under dev</td>
<td>MPI</td>
<td>3D/rz/xy</td>
<td>up to 10^8</td>
<td>linacs/rings</td>
</tr>
</tbody>
</table>
## Spreadsheet of Space-Charge Codes II

<table>
<thead>
<tr>
<th>Code</th>
<th>Space Charge Solver</th>
<th>Boundaries/Images</th>
<th>Impedances</th>
<th>Field Maps</th>
<th>Integration order</th>
</tr>
</thead>
<tbody>
<tr>
<td>IMPACT</td>
<td>spectral</td>
<td>open/periodic/rectangular/circular</td>
<td>no</td>
<td>yes</td>
<td>2nd order in $z$</td>
</tr>
<tr>
<td>ML-IMPACT</td>
<td>spectral</td>
<td>elliptical/polygon/lossy</td>
<td>yes</td>
<td>no</td>
<td>2nd in $z$</td>
</tr>
<tr>
<td>PARMILA</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td>5th Runge-Kutta</td>
</tr>
<tr>
<td>GPT</td>
<td>3D multigrid</td>
<td>open conductive rect. pipe, cathode</td>
<td>no</td>
<td>2D,3D</td>
<td>5th Runge-Kutta</td>
</tr>
<tr>
<td>BEST</td>
<td>spectral, FD</td>
<td>circular conducting wall</td>
<td>automatic/external</td>
<td>no</td>
<td>user specified</td>
</tr>
<tr>
<td>VADOR</td>
<td>FFT</td>
<td>conductive wall any shape</td>
<td>no</td>
<td>no</td>
<td>2nd</td>
</tr>
<tr>
<td>SPUNCH</td>
<td>exact for disc-shaped particles</td>
<td>circular conducting wall</td>
<td>n/</td>
<td>n/a</td>
<td>1st</td>
</tr>
<tr>
<td>PATH</td>
<td>Schell, pt-to-pt</td>
<td>open</td>
<td>no</td>
<td>yes</td>
<td>?</td>
</tr>
<tr>
<td>TRACEWIN</td>
<td>Scheff/PICNIC/Gaussup</td>
<td>open</td>
<td>no</td>
<td>no</td>
<td>?</td>
</tr>
<tr>
<td>DYNAC</td>
<td>Scheff/Scherm/Hersc</td>
<td>open</td>
<td>no</td>
<td>yes</td>
<td>3rd analytical</td>
</tr>
<tr>
<td>Synergia</td>
<td>spectral (IMPACT)</td>
<td>open/periodic/rectangular/circular</td>
<td>no</td>
<td>yes</td>
<td>2nd order in $z$</td>
</tr>
<tr>
<td>WARP</td>
<td>FFT, Cap matrix,</td>
<td>square/round pipe, internal conductors,</td>
<td>ad hoc</td>
<td>no</td>
<td>2nd order</td>
</tr>
<tr>
<td></td>
<td>multigrid, adaptive</td>
<td>bent pipe, general</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td>mesh, refined MG</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>
## Spreadsheet of Space-Charge Codes III

<table>
<thead>
<tr>
<th>Code</th>
<th>$t$ or $s$ tracking</th>
<th>Graphics</th>
<th>Portability</th>
<th>Source code available?</th>
<th>Manual</th>
<th>Standard test cases</th>
</tr>
</thead>
<tbody>
<tr>
<td>IMPACT</td>
<td>$s$</td>
<td>post proc.</td>
<td>all unix platforms</td>
<td>to collaborators</td>
<td>partial</td>
<td>yes</td>
</tr>
<tr>
<td>ML-IMPACT</td>
<td>$s$</td>
<td>post proc.</td>
<td></td>
<td>to collaborators</td>
<td>partial</td>
<td>yes</td>
</tr>
<tr>
<td>GPT</td>
<td>$t$</td>
<td>built in</td>
<td>portable except user interface</td>
<td>all beamline components</td>
<td>yes</td>
<td>yes</td>
</tr>
<tr>
<td>BEST</td>
<td>$t$</td>
<td>netcdf, IDL</td>
<td>any Linux</td>
<td>yes</td>
<td>no</td>
<td>yes</td>
</tr>
<tr>
<td>VADOR</td>
<td>$s$</td>
<td>GNUplot, openDX</td>
<td>any Linux</td>
<td>yes</td>
<td>almost</td>
<td>yes</td>
</tr>
<tr>
<td>SPUNCH</td>
<td>$s$</td>
<td>built in</td>
<td>fully portable</td>
<td>yes</td>
<td>no</td>
<td>yes</td>
</tr>
<tr>
<td>PATH</td>
<td>$s$</td>
<td>built in</td>
<td>any Windows</td>
<td>yes</td>
<td>yes</td>
<td>?</td>
</tr>
<tr>
<td>TRACEWIN</td>
<td>$s$</td>
<td>built in</td>
<td>any Windows</td>
<td>no</td>
<td>yes</td>
<td>?</td>
</tr>
<tr>
<td>DYNAC</td>
<td>$t, s$</td>
<td>GNUplot</td>
<td>fully portable</td>
<td>yes</td>
<td>yes</td>
<td>yes</td>
</tr>
<tr>
<td>Synergia</td>
<td>$s$</td>
<td>post proc.</td>
<td>all Linux</td>
<td>yes</td>
<td>in progress</td>
<td>not yet</td>
</tr>
<tr>
<td>WARP</td>
<td>$t, s$</td>
<td>PyGist 2D, OpenDX 3D</td>
<td>portable</td>
<td>yes</td>
<td>online</td>
<td>yes</td>
</tr>
</tbody>
</table>
SAVE EVERYTHING:

Even if you think the job is finished, don’t for one moment think you will never need the data or the code again.