Swimming Downstream

Plans for Topics in GlueX Tracking

Mark M. Ito

Jefferson Lab

December 7, 2007

Overview

- Overall goal: least-squares track fitting framework
  - non-uniform magnetic field
  - geometry independent
- swimming
- fitting

choices for swimming

- EGS
- geant3
- geant4
- DTrajectory
- google

Michel's scheme

F. Curtis Michel, "Numerical integration of trajectories in static magnetic fields," http://cnx.org/content/m12765/latest/
Start with Eq. (1) of Ref. 1.

$$\mathbf{F} = e(\mathbf{E} + \mathbf{v} \times \mathbf{B})$$

If $\mathbf{E} = 0$

$$\frac{d\mathbf{p}}{dt} = e(\mathbf{v} \times \mathbf{B})$$

Also

$$\mathbf{p} = \gamma m \mathbf{v}.$$

If we let $k_1 = e\Delta t/\gamma m$ then

$$\Delta \mathbf{v} = k_1(\bar{\mathbf{v}} \times \mathbf{B})$$

where $\Delta \mathbf{v} = \mathbf{v}' - \mathbf{v}$ and $\bar{\mathbf{v}} = (\mathbf{v}' + \mathbf{v})/2$. If we define the vector $\mathbf{k} = k_1 \mathbf{B}/2$, then

$$\mathbf{v}' = \mathbf{v} + [(\mathbf{v}' + \mathbf{v}) \times \mathbf{k}]$$

the x-component of this equation is

$$v'_x = v_x + (v'_y + v_y)k_z - (v'_z + v_z)k_y$$

recovering eq. (24) of Ref. 1. Rewriting this as

$$v'_x - v'_y k_z + v'_z k_y = v_x + v_y k_z - v_z k_y$$

we note that all three components in matrix notation can be written as

$$\begin{pmatrix} 1 & -k_z & k_y \\ k_z & 1 & -k_x \\ -k_y & k_x & 1 \end{pmatrix} \begin{pmatrix} v'_x \\ v'_y \\ v'_z \end{pmatrix} = \begin{pmatrix} 1 & k_z & -k_y \\ -k_z & 1 & k_x \\ k_y & -k_x & 1 \end{pmatrix} \begin{pmatrix} v_x \\ v_y \\ v_z \end{pmatrix}$$

recovering Eq. (25) (without the typo). We can write this as

$$(I + A)\mathbf{v}' = (I - A)\mathbf{v}$$

where the anti-symmetric matrix

$$A = \begin{pmatrix} 0 & -k_z & k_y \\ k_z & 0 & -k_x \\ -k_y & k_x & 0 \end{pmatrix}$$

and I is the identity matrix.

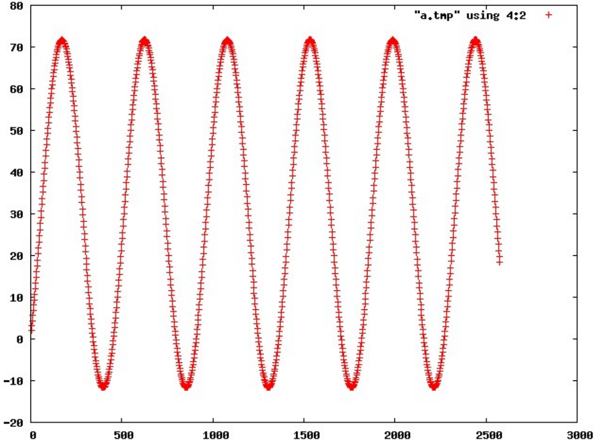
Let $M = (I + A)^{-1}(I - A)$ and $\mathbf{v} = (\mathbf{r}_1 - \mathbf{r}_0)/\Delta t$ and $\mathbf{v}' = (\mathbf{r}_2 - \mathbf{r}_1)/\Delta t$. Then we have

$$\mathbf{r}_2 = \mathbf{r}_1 + M(\mathbf{r}_1 - \mathbf{r}_0)$$

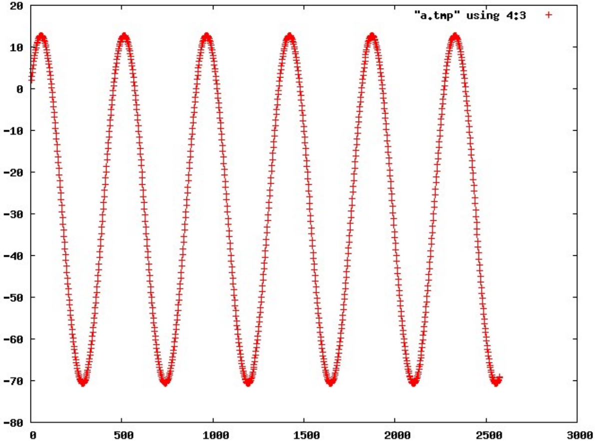
which we can iterate.
comments:

- more accurate than Rutta-Kunge
- simple implementation
- time-of-flight info for free

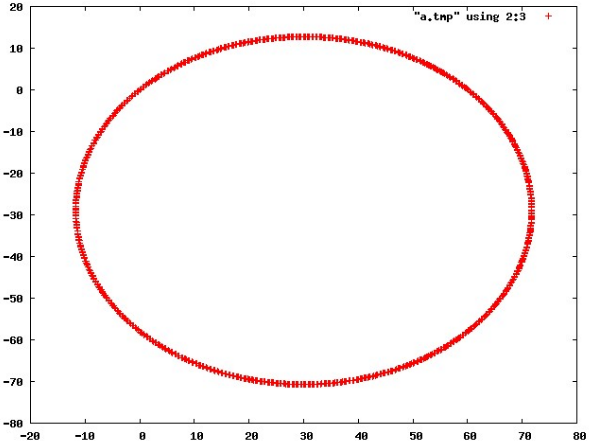
$B = 4$ T, $p = 1$ GeV$/c$, x vs. z



$B = 4$ T, $p = 1$ GeV/c, y vs. z



$B = 4$ T, $p = 1$ GeV$/c$, x vs. y



MyTrajectory C++ class

base class

```
class MyTrajectory {
 public:
  MyTrajectory();
  void swim(HepVector startingPoint,
            double theta, double phi);
  double doca(HepVector& spacePoint);
...
```

derived class

```
class MyTrajectoryHelix : public MyTrajectory {
 public:
  MyTrajectoryHelix(HepVector B);
  void swim(double charge, HepVector startingPoint,
            double p, double theta, double phi);
...
```

swimming future

- B-field map
- comparison with others methods

fitting: distance of closest approach (DOCA) member function

- iterated parabolic interpolation
  - independent of functional form
- point-to-trajectory done
  - good for FDC pseudo-points
- line-to-trajectory to do

10.2 Parabolic Interpolation and Brent's Method 407

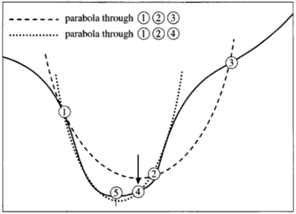


Figure 10.2.1. Convergence to a minimum by inverse parabolic interpolation. A parabola (dashed line) is drawn through the three original points 1,2,3 on the given function (solid line). The function is evaluated at the parabola's minimum, 4, which replaces point 3. A new parabola (dotted line) is drawn through points 1,4,2. The minimum of this parabola is at 5, which is close to the minimum of the function.

fitting: χ^2 minimizer

- GNU Scientific Library (GSL) non-linear least squares fitter
  - method name???
  - weighted or unweighted
  - depends on (weighted) residuals only
  - implemented in C
- C++ wrapper written, being tested
  - ugly work around for C++
  - problem with C callback to C++ member functions
  - ROOT also has wrapper

fitting: to do...put pieces together

- swimmer
- DOCA
- GSL χ^2 fitter