EUROPEAN ORGANIZATION FOR NUCLEAR RESEARCH
CERN – PS DIVISION

CERN/PS 99-002 (OP)
CERN/IT/99/1

TOMOGRAPHIC MEASUREMENTS OF LONGITUDINAL PHASE SPACE DENSITY

S. Hancock, M. Lindroos, E. McIntosh, M. Metcalf

Abstract

Tomography – the reconstruction of a two-dimensional image from a series of its one-dimensional projections – is now a very broad topic with a wealth of algorithms for the reconstruction of both qualitative and quantitative images. One of the simplest algorithms has been modified to take into account the non-linearity of large-amplitude synchrotron motion in a particle accelerator. This permits the accurate reconstruction of longitudinal phase space density from one-dimensional bunch profile data. The algorithm was developed in Mathematica™ in order to exploit the extensive built-in functions and graphics. Subsequently, it has been recoded in Fortran 90 with the aim of reducing the execution time by at least a factor of one hundred. The choice of Fortran 90 was governed by the desire ultimately to exploit parallel architectures, but sequential compilation and execution have already largely yielded the required gain in speed. The use of the method to produce longitudinal phase space plots, animated sequences of the evolution of phase space density and to estimate accelerator parameters is presented. More generally, the new algorithm constitutes an extension of computerized tomography which caters for non-rigid bodies whose projections cannot be measured simultaneously.

Conference on Computational Physics, Granada, September 1998
(Accepted for publication in Computer Physics Communications)

Geneva, Switzerland
19 January 1999

1 INTRODUCTION

The underlying principle of tomography is to combine the information in a sufficiently large number of projections to be able to reconstruct unambiguously the fuller picture with the extra dimension reinstated. Thus, for example, many one-dimensional profiles of x-ray transparency taken from different angles can give doctors an image of a two-dimensional slice through a patient.

The application of tomography to longitudinal phase space in an accelerator becomes obvious once it is realised that a bunch of particles performing synchrotron motion is analogous to a patient rotating in a stationary body scanner. On each turn around the machine, a longitudinal pick-up provides a "snapshot" of the bunch projected at a slightly different angle. It only remains to combine such profiles tomographically to obtain a two-dimensional picture of phase space density [1, 2].

2 RECONSTRUCTION

Back projection is a key process by which the contents of the bins of a one-dimensional histogram are redistributed over the two-dimensional array of cells which comprise the reconstructed image. Given no *a priori* knowledge of the original two-dimensional distribution, the contents of one bin is shared over all the cells that could have contributed to that bin. The back projection of all bins of all profiles yields a first approximation to the original distribution.

Algebraic Reconstruction Techniques (ART) [3] exploit the fact that the coefficients for sharing bins in back projection can also be used to project the contents of cells into those bins. Hence a set of projections can be obtained from the first approximation. Back projection of the bin-by-bin difference between the original set of profiles and this new one yields an improved approximation. Further iterations converge more rapidly if any cell whose contents has become negative is reset to zero.

The problem with conventional ART is that its strategies for estimating the redistribution coefficients are based on straight-line back projection geometry. This implies either rigid, circular motion of the two-dimensional distribution or that its projections be measured simultaneously. An alternative approach is to consider how the contents of one cell gets projected into the bins of a particular profile. By launching a small number of test particles which, initially, are uniformly distributed within the cell, the calculation of coefficients becomes a simple matter of counting how many particles end up in each bin at the particular instant when the profile was measured. Thus, a hybrid algorithm which combines particle tracking with ART allows large-amplitude synchrotron motion to be taken into account since the trajectories of the test particles need not be assumed circular. Although iteration proceeds as before, there is a price to be paid: a large map of (projection) coefficients must first be built and its inverse (for back projection) derived for every profile in the set of measured data. On the other hand, since most of the computational effort is invested in building the maps, it becomes trivial to repeat a calculation with fresh data taken under the same conditions.

The method is very robust. Since the iterations converge to the consensus of the information in the profiles, trigger jitter and noise in the data have only a minor effect. In addition, since the test particles are only tracked for a relatively brief period to build the maps, moderate errors in the accelerator parameters used in the tracking model are not serious.

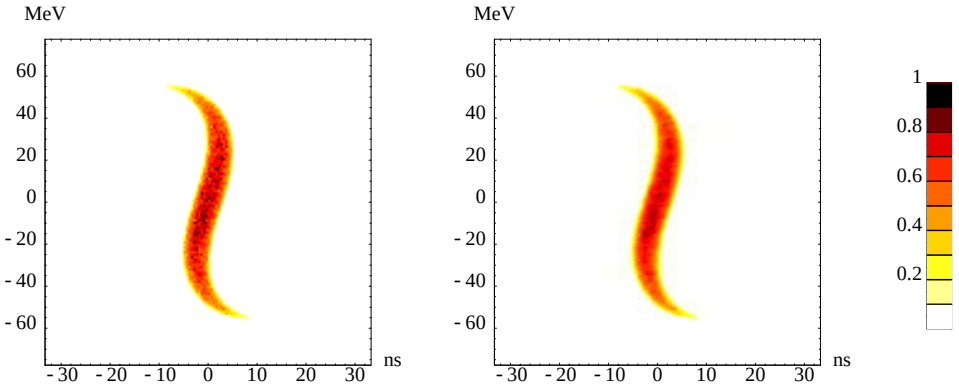

Figure 1: Simulated proton distribution and its reconstruction (on the same density scale).

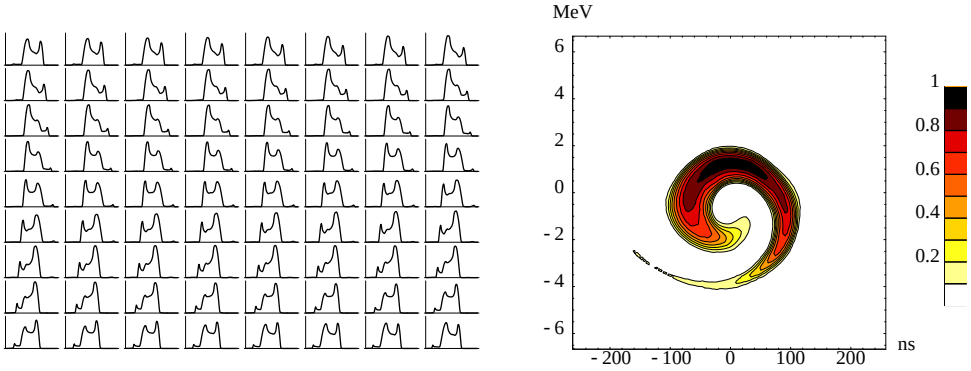

Figure 2: Measured profiles and reconstructed contour plot of a bunch after phase modulation at 1 GeV.

3 SOME RESULTS

The first part of Fig. 1 shows a simulated distribution comprising an unmatched bunch. These particles were tracked over an entire synchrotron period to obtain 72 profiles as the distribution filamented. Taking these profiles as the input dataset, the new algorithm yields the accurate reconstruction of the second part of Fig. 1.

The 72 profiles of Fig. 2 were measured every 48 turns at the CERN Booster during attempts to move empty phase space into the centre of a normal bunch by phase modulation of the rf. Without tomography, it would be difficult to deduce anything from the bunch shape information.

Arbitrarily complex rf systems can be treated. The "mountain range" plot of Fig. 3 shows a series of measurements every 50 turns during bunch splitting in the CERN PS. Splitting is achieved by increasing the second-harmonic voltage component of a dual-harmonic rf system while reducing the fundamental. The data span 10 ms and the details of the two voltage programmes during this period were incorporated in the tracking model of the algorithm to produce the corresponding reconstructed image. Since the origin in time at

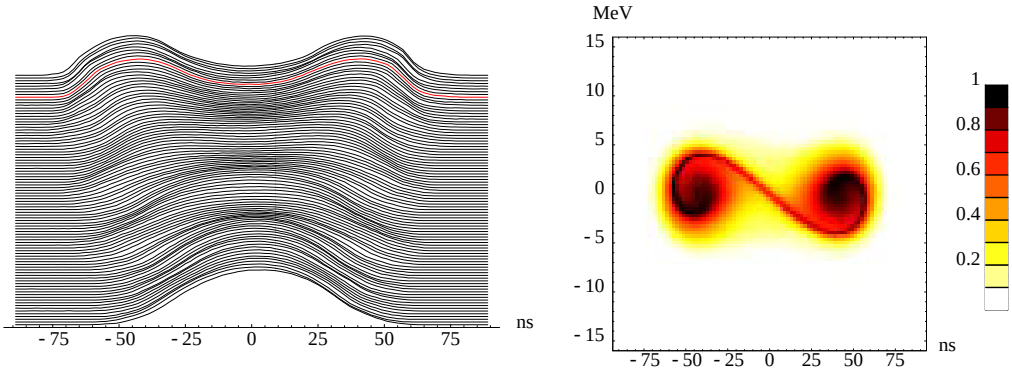

Figure 3: Quasi-adiabatic bunch splitting at 3.5 GeV/c.

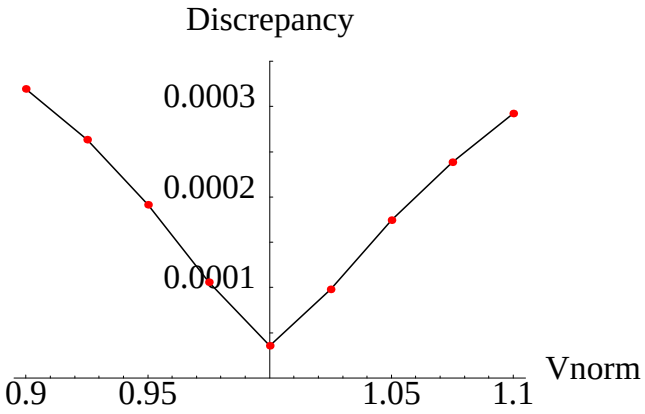

Figure 4: Discrepancy versus normalized rf voltage.

which the test particles are launched can be chosen freely within the measurement interval, the reconstruction can be at any instant during that period. In Fig. 3, the reconstruction is at the time of the 84th profile (out of 92). An animation of the entire process can be made by repeating the calculation for all time slices.

"Discrepancy" [3] expresses in a single figure of merit the residual differences between the projections of a reconstructed distribution and the original profiles. As such it may be used to monitor convergence. It is an obvious conjecture that, in the limit of many iterations, the lowest discrepancy should be obtained when the parameters used in the tracking model match most closely the reality of the machine. Reconstruction of the simulated data of Fig. 1 assuming different rf voltages supports this idea. The discrepancy after one hundred iterations exhibits a sharp minimum at precisely the value of the rf voltage that was used to generate the data in the first place (see Fig. 4). Arguably, this constitutes a voltage measurement.

4 COMPUTER CODE

4.1 Program philosophy

The algorithm was originally developed in Mathematica™, a choice mainly motivated by the rich set of built-in functions for graphics and for the manipulation of arrays. However, the objective was merely to establish a proof of principle. Consequently, with the aim of reducing the execution time by a factor of at least one hundred, the code was translated into Fortran 90/High Performance Fortran (HPF). This was done with the view to use parallel architectures. Mathematical toolkits with the associated definition modules from Numerical Recipes [4] were integrated in the program from the outset to avoid unnecessary duplication of standard numerical routines. The entire issue of providing a user interface for the demanding control room environment of the CERN accelerator complex was separated out as an independent development. This has now been started and will be discussed in Section 4.6. For the development of the numerical part of the code, simple input and output files were used to avoid any possible complications associated with the choice of a particular graphics package. The display of resulting graphs and surfaces has been done with Mathematica™ throughout the development phase.

The fact that the method could be applicable to processes other than synchrotron motion implied a division of the code into generic modules and modules specific to longitudinal beam dynamics. Furthermore, all parameters characterizing the considered process are passed to the program as input at execution time, making the code to a large extent data-driven. In Fig. 5 a schematic view of the structure of the program is presented.

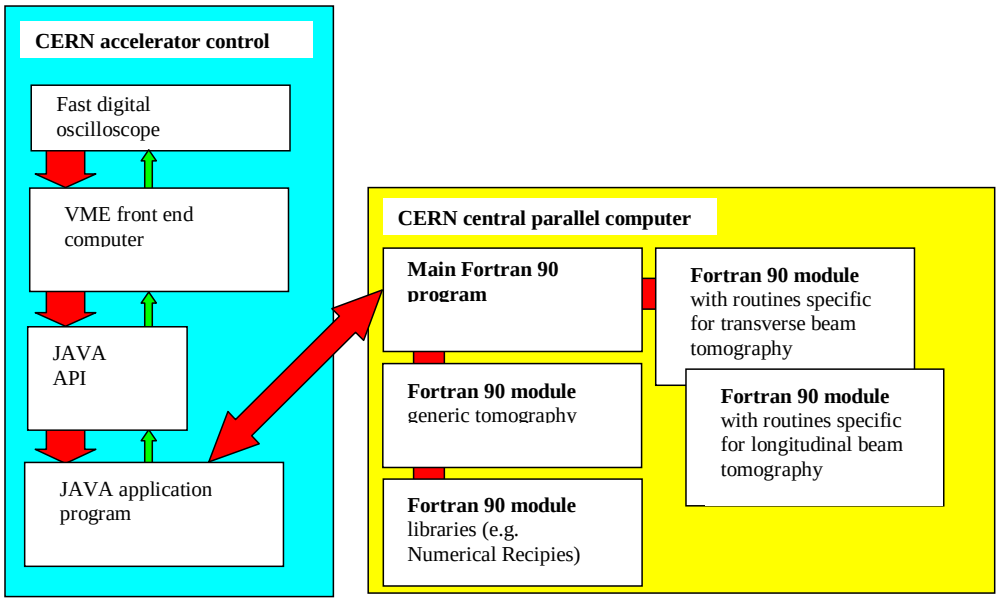

Figure 5: A schematic view of a future on-line tomography system. To the left, the hardware and Graphical User Interface (GUI) configuration is drawn and, to the right, the structure of the present numerical part of the code. The data flow is indicated with broad arrows and the command flow with narrow ones.

4.2 Coding considerations

In the original Mathematica™ code there is extensive use of floating-point operations and many very large and very sparse matrices. In the Fortran 90 version, the number of floating-point operations is reduced by the use of integer representation until the final step in each number manipulation, while a pointer-like re-allocatable vector representation was created to deal with the sparse matrices. The sparse matrices are stored in an array with sufficient depth to contain most of the data. Where additional depth is needed the excess data are stored in a supplementary array and the index to the array element used is stored in the last element of the original matrix. The supplementary array is re-allocatable and this procedure can obviously be repeated any number of times. Consequently, the actual depth of the matrix is only limited by the available memory during execution. The built-in pointer facility of Fortran 90 was avoided because it would make the efficient use of parallel architectures with shared memory impossible.

A fundamental part of the algorithm is the construction of the forward and backward projection maps by tracking a small number of test particles launched from each cell in the two-dimensional image. The individual test particles are tracked without considering any particle-to-particle interaction. Consequently, the launching and tracking of all particles for all cells can be done in parallel, a fact which was taken into consideration from the beginning and is fully exploited in a parallel version.

4.3 Optimization

A first execution time analysis showed that more than 90% of the time was spent in the tracking procedure longtrack, 75% in evaluating sine functions (despite the use of fast libraries). In the case of Fig. 3, for example, longtrack was called 91 times to make 50 updates to some 6×10^5 floating-point numbers representing the test particles. A total approaching 3×10^9 sines have to be evaluated for one such image.

As a first step, the call to the Fortran library SIN function was replaced by an inline procedure using classic polynomial approximations for $cos(x)$ to 4th and 10th order in x with a maximum error of about 10^{-3} and 2×10^{-9}, respectively [6]. Further statement level profiling showed a very significant part of the time was now spent reducing the argument range to $0 \leq x < \pi/2$. The precision of the first approximation was considered inadequate and that of the second unnecessary since it would easily be swamped by errors in the parameters of the accelerator model.

The sine function evaluation was therefore replaced by a look-up procedure over the range -2π to 2π. Tests with a table of 8192 values per quadrant produced satisfactory results and required 512 kBytes, not too much compared with an application of some 16 MBytes. Choosing a power of two for the number of tabulated values enables division to be performed by shifting. The removal of all tests, branches and functions from the procedure allows full pipe-lining or vectorisation on processors with the appropriate architecture.

The influence of each of the approximations outlined above was tested using simulated data generated from a known two-dimensional distribution.

Timing measurements in a production environment turned out to be heavily dependent on the system load. While this effect is still under investigation, dividing the matrices into blocks not only produced a further improvement due to better usage of the cache, but also reduced the timing variations. This technique also removed the need for some large

temporary matrices generated by the compiler.

All the optimizations have somewhat reduced the generality of the code, but the same techniques can be easily applied to new rf voltage programmes. The necessary changes are confined to a single subroutine in the tracking module:

```
SUBROUTINE longtrack(direction,nrep,yp,xp,turnnow)
  USE nrstuff
  IMPLICIT NONE
  REAL(SP), DIMENSION(:), INTENT(INOUT) :: xp,yp
  INTEGER :: i,nrep,direction,turnnow
  INTEGER, PARAMETER :: nblock=100
  INTEGER, PARAMETER :: steps=4*8192
  REAL(SP), PARAMETER :: delta=twopi/steps
  REAL(SP), PARAMETER :: deltareciproc=1.0_SP/delta
  REAL(SP), DIMENSION(-steps+1:steps-1), SAVE :: sine
  REAL(SP), SAVE :: hratiod
  REAL(SP), DIMENSION(nblock) :: dphi,denergy
  REAL(SP), DIMENSION(:), ALLOCATABLE, SAVE :: avrf1,avrf2
  REAL(SP) :: ang1,ang2,s1,s2
  INTEGER :: ind1,ind2
  INTEGER :: t,j,j1,nleft,first
  DATA first/1/
  IF (first.EQ.1) THEN
    first=0
    ALLOCATE(avrf1(1:nrep),avrf2(1:nrep))
    hratiod=hratio*deltareciproc
    sine(0)= 0.0_SP
    DO j=1,steps-1
      sine(j)=SIN((j+0.5_SP)*delta)
      sine(-j)=-sine(j)
    ENDDO
  ENDIF
  t=turnnow
  IF (direction.GT.0) THEN
    DO i=1,nrep
        avrf1(i)=q*(VRF1+VRF1dot*tatturn(t+i))
        avrf2(i)=q*(VRF2+VRF2dot*tatturn(t+i))
    END DO
    DO j1=1,SIZE(xp),nblock
      nleft=SIZE(xp)-j1+1
      IF (nleft.gt.nblock) nleft=nblock
      t=turnnow
      DO j=1,nleft
        dphi(j)=(xp(j1+j-1)+xorigin)*h*omegarev0(t)*&
                 dtbin-phi0(t)
        denergy(j)=(yp(j1+j-1)-yat0)*dEbin
      END DO
      DO i=1,nrep
        t=t+1
        DO j=1,nleft
```

```
        dphi(j)=dphi(j)-c1(t-1)*denergy(j)
        ang1=dphi(j)+phi0(t)
        sine(0)=ang1
        ind1=(deltareciproc*ang1)
        ind1=ind1-(ind1/steps)*steps
        s1=sine(ind1)
        ang2=ang1-phi12
        sine(0)=ang2
        ind2=(hratiod*ang2)
        ind2=ind2-(ind2/steps)*steps
        s2=sine(ind2)
        denergy(j)=denergy(j)+&
        avrf1(i)*s1+avrf2(i)*s2-c2(t)
      END DO
    END DO
    DO j=1,nleft
      xp(j1+j-1)=(dphi(j)+phi0(t))/&
                 (h*omegarev0(t)*dtbin)-xorigin
      yp(j1+j-1)=denergy(j)/dEbin+yat0
    END DO
  END DO
ELSE
  DO i=1,nrep
      avrf1(i)=q*(VRF1+VRF1dot*tatturn(t-i+1))
      avrf2(i)=q*(VRF2+VRF2dot*tatturn(t-i+1))
  END DO
  DO j1=1,SIZE(xp),nblock
    nleft=SIZE(xp)-j1+1
    IF (nleft.gt.nblock) nleft=nblock
    t=turnnow
    DO j=1,nleft
      dphi(j)=(xp(j1+j-1)+xorigin)*h*omegarev0(t)*&
              dtbin-phi0(t)
      denergy(j)=(yp(j1+j-1)-yat0)*dEbin
    END DO
    DO i=1,nrep
      DO j=1,nleft
        ang1=dphi(j)+phi0(t)
        sine(0)=ang1
        ind1=(deltareciproc*ang1)
        ind1=ind1-(ind1/steps)*steps
        s1=sine(ind1)
        ang2=ang1-phi12
        sine(0)=ang2
        ind2=(hratiod*ang2)
        ind2=ind2-(ind2/steps)*steps
        s2=sine(ind2)
        denergy(j)=denergy(j)-&
        avrf1(i)*s1-avrf2(i)*s2+c2(t)
        dphi(j)=dphi(j)+c1(t-1)*denergy(j)
      END DO
```

```
        t=t-1
      END DO
      DO j=1,nleft
        xp(j1+j-1)=(dphi(j)+phi0(t))/&
                   (h*omegarev0(t)*dtbin)-xorigin
        yp(j1+j-1)=denergy(j)/dEbin+yat0
      END DO
    END DO
  END IF
  turnnow=t
END SUBROUTINE longtrack
```

4.4 Timing results for Digital Alpha and IBM Power PC

Timing results for different processors are shown in Tab. 1. Here, PPC refers to a 332 MHz Model 43P-140 IBM Power PC, Alpha to a 500 MHz Digital EV56 Alpha Workstation and Turbo to a 350 MHz Digital EV5 10 CPU Turbolaser. An overall speed-up by a factor of more than two has been obtained. The largest gain was for the Power PC, where there is presumably a greater imbalance between processor and memory speed. There are no platform-specific optimizations other than compiler options.

Table 1: CPU time in seconds for the reconstruction of three well-resolved images using different processors and different versions of the sine function.

Version	Alpha	Turbo	PPC
Original F90 code (F90 fast library SIN)	475	655	1060
Optimized code	228	322	233

4.5 Parallelization

Although the optimizations described above have reduced the CPU time per call to the tracking procedure to the order of one second, the use of multiple processors with HPF directives and the parallel FORALL statement is still under study. A parallel version of the code could provide the same performance at lower cost by using a dual or quadruple CPU PC instead of a high performance workstation.

Since a large part of the execution time is spent in the longtrack subroutine, an HPF/F95 version of longtrack has been developed. First tests with this routine showed a speed-up of two on a dual processor machine. The parallelized code follows:

```
SUBROUTINE longtrack(direction,nrep,yp,xp,turnnow)
  USE nrstuff
  IMPLICIT NONE
  REAL(SP), DIMENSION(:), INTENT(INOUT) :: xp,yp
  REAL(SP), DIMENSION(SIZE(xp)) :: dphi,denergy
!HPF$ distribute dphi(block)
!HPF$ align with dphi :: denergy
integer :: mm
  INTEGER :: i,nrep,direction,turnnow
```

```
  dphi=(xp+xorigin)*omegarev0(turnnow)*h*dtbin-phi0(turnnow)
  denergy=(yp-yat0)*dEbin
  IF (direction.GE.0) THEN
    DO i=1,nrep
      forall(mm=1:size(xp)) dphi(mm)=dphi(mm)-c1(turnnow)*denergy(mm)
      forall(mm=1:size(xp)) denergy(mm)=denergy(mm)+        &
        ( q*((VRF1+VRF1dot*tatturn(turnnow+1))*             &
             SIN(dphi(mm)+phi0(turnnow+1))+                 &
             (VRF2+VRF2dot*tatturn(turnnow+1))*             &
             SIN(hratio*(dphi(mm)+phi0(turnnow+1)-phi12)))- &
          c2(turnnow+1) )
      turnnow=turnnow+1
    END DO
  ELSE
    DO i=1,nrep
      forall(mm=1:size(xp)) denergy(mm)=denergy(mm)-        &
        ( q*((VRF1+VRF1dot*tatturn(turnnow))*               &
             SIN(dphi(mm)+phi0(turnnow))+                   &
             (VRF2+VRF2dot*tatturn(turnnow))*               &
             SIN(hratio*(dphi(mm)+phi0(turnnow)-phi12))) )+ &
        c2(turnnow)
      forall(mm=1:size(xp)) dphi(mm)=dphi(mm)+c1(turnnow-1)*denergy(mm)
      turnnow=turnnow-1
    END DO
  END IF
  xp=(dphi+phi0(turnnow))/(h*omegarev0(turnnow)*dtbin)-xorigin
  yp=denergy/dEbin+yat0
END SUBROUTINE longtrack
```

4.6 User interface

The present trend at CERN is towards a common control environment for all the accelerators with the console software written in Java communicating with the control system through a Java Application Program Interface (API) [5]. The GUI for the tomography program is presently being specified (see Fig. 5). The plan is to write the GUI and the hardware interface in Java with the execution of the numerically demanding part of the code in a parallel environment. The call to the parallel environment could be made using several different mechanisms, but the one presently favoured is through remote job submission where the job is passed as well as the input and output data. The necessary response time, defined as the time from the start of the measurement until the tomographically reconstructed image is visible on the screen, is set by the CERN machine cycle and should ideally be of the order of 15 seconds for a well-resolved image.

4.7 Usage notes

Successful reconstruction requires that the measured data span an interval of the order of at least half a synchrotron period. In addition, normalization requires that each profile encompass the same number of particles. Together, these constraints suggest that no particles exist outside the largest closed phase space trajectory that can be drawn inside the image width. Consequently, in order to reduce computation time, map coefficients are

only derived for those cells of the image that lie within this limiting trajectory at the reconstruction time. The cells of interest can be further restricted to a range of columns between upper and lower limits supplied as input parameters. This effectively declares the profile bins that lie outside this range to be empty at the reconstruction time, although they may well be populated at other times. In the event that the closed trajectory is too restrictive to permit the reconstruction of the entire distribution, a flag may be set to extend the region of interest up to a fixed off-energy limit for all columns in the specified range. If this flag is not set and there is no second-harmonic rf component, the cell height is chosen such that small-amplitude trajectories in the reconstructed image are circular.

Although a mechanism is provided to reconstruct at multiple time slices within the measurement interval and hence produce an animation of phase space motion, the original concept was to build a single "snapshot" of phase space from the information contained in a brief span of data. In this spirit, the particle tracking is very elementary and the only parameter variations considered are constant first-order time derivatives of the dipole magnetic field and rf voltages.

5 CONCLUSIONS

An algorithm has been developed for longitudinal phase space tomography in which the contents of the reconstructed array is effectively rotated instead of inclining profile bins in order to make a projection. This allows a different mapping to be applied to each cell in the array so that rigid, circular motion of the phase space distribution need not be assumed.

Simulated proton data have shown the method to be both accurate and robust. Valuable tomographic measurements have been made during machine experiments. The algorithm is a hybrid one and, consequently, arbitrarily complex rf systems can be catered for by modifying a small part of the code. Likewise the method may be extended to cover other non-rigid bodies whose deformation is governed by a known model.

The recent advances with optimization of the numerical part of the code are very promising. It seems possible that a GUI in Java together with number crunching in parallel structures will yield response times of less than one minute for a single image, making on-line machine optimization with tomography possible.

References

[1] S. Hancock, "A Simple Algorithm for Longitudinal Phase Space Tomography", CERN PS/RF/Note 97-06, 1997

[2] G. Jackson, "A Phase Space Tomography (PST) Monitor for Adjusting Bunch Rotation During Coalescing", Fermi Nat. Acc. Lab. FN-469, 1987

[3] R. Gordon, "A Tutorial on ART", IEEE Trans. Nucl. Sci., NS-21, pp. 78-93, 1974

[4] W. H. Press, S. A. Teukolsky, W. T. Vetterling and B. P. Flannery, "Numerical Recipies in Fortran 90: The Art of Parallel Scientific Computing", Cambridge University Press, 1996

[5] P. Charrue, J. Cuperus, I. Deloose, F. DiMaio, K. Kostro, M. Vanden Eynden, W. Watson, "Java Controls API, Project Definition Report", CERN PS/CO/Note 98-09, 1998

[6] "Rational Approximation of Functions", Los Alamos Scientific Laboratory LA-1943, Los Alamos, N. Mex., 1955